Primer on Scripting Comsol with Matlab

Written by Matt Aasted, Class of 2008
Last edited: March 2, 2007

Introduction

When optimizing the geometry of a shape in Comsol, you will frequently repeat the same steps over and over again with slight variations. The goal of this paper is to give an example of how to make Matlab do the footwork for you, and digest the results.

This document explains how to repeat simple operations in Comsol in a way which can be controlled by Matlab. It does not attempt to comprehensively document the possibilities of interaction between the two programs -- for more information on the interface between them, use the help documentation in Comsol.

General Theory

Comsol is actually implemented in Matlab as a toolbox. Therefore, Comsol is able to output exactly what it having Matlab do as a series of commands in an m file (a Matlab script). Each major action (usually involving hitting ok and a statement at the bottom of the main window) performed in Comsol maps to a statement written in the m file, in the order they are performed. For example, if you hit the "Refine Mesh" button, it corresponds to a line in the m file. Be careful to avoid extraneous actions to prevent the m file from becoming too complicated to work with.

For this example, I will be making a rectangle out of Iron, applying a temperature to one side and varying its dimensions. Feel free to change the geometry and materials to fit your application, but I would recommend getting a feel for the rectangle first since it is simple.

Instructions

1: Open Comsol (Not with Matlab -- for some reason, setting the subdomain didn't work for me when I tried this with Matlab, though you may have different result)
In the Model Navigator window, choose Application Modes -> COMSOL Multiphysics -> Conduction -> Steady State Analysis. Hit Ok.

The program will open a new window.

2: Draw your geometry

For this example, I will draw a rectangle using the upper rectangle tool. It will be 1 by 0.6 starting at (-.6,-.4) and going to (.4,.2). Coordinates are of the form (x,y).

3: Set subdomain settings.
Click Physics -> Subdomain Settings... and choose "1" in the Subdomain Selection box. In the Physics tab, hit the "Load..." button and in the window which opens choose Library 1 -> Iron. Hit Ok, and hit Ok again in the Subdomain Setting window.

4: Boundary settings.
For this example, we will set the left hand side to being 400 K and all other sides to having convective cooling to the environment.

a. Click Physics -> Boundary settings. Under boundary selection, select 1. You can verify that this is the left hand side of the box by noticing that when you select it, the corresponding boundary turns red in the background window. Under the Coefficients tab, choose the boundary condition Temperature and in the T_0 parameter box (it's the only other white box) enter 400.

b. Under boundary selection, choose 2, 3 and 4 by clicking 2 then holding control while clicking 3 and 4. Under Boundary Condition choose "Heat Flux". This opens up a variety of parameters for setting. q_0 is a constant heat flux value independent of temperature, h is the convection constant, T_inf is the air temperature for convection, Const is a constant for radiative cooling, and T_amb is the temperature for radiative cooling. Since we want a boundary condition of convection, we leave all parameters except for h and T_inf at 0. Set h to 10 in the box next to it, and set T_inf to 300 (a little warmer than room temperature) in its box. Click ok.

5: Mesh
a. Choose Mesh -> Initialize Mesh
b. Now Choose Mesh -> Refine Mesh

6: Solve the problem
Choose Solve -> Solve Problem

It will take a moment and then present you with a colorful graph of your solution.

7: Measure boundary heat flux.
We will now integrate the heat flowing across the temperature boundary and across the convection boundaries.
Choose Postprocessing -> Boundary integration
a. Under boundary selection, click 1.

Under Expressions to integrate in Predefined quantities, choose Normal Heat Flux (the heat flux perpendicular to the boundary). Click Apply. The result will appear in the bottom of the main window.
b. In the same window, select boundaries 2-4 in the way described in 4b. Choose Normal Heat Flux, and click Ok.

8: Save the m-file.
Choose File -> Save. In the window that opens, choose where you want to save the file and what you want to name it. Under Files of Type, choose Model M-file (*.m) and click Ok.

9: Close Comsol

10: Open Comsol with Matlab (in the default setup, it is Start -> Program Files -> Comsol -> Comsol with Matlab)

11: Close Comsol but not Matlab -- it is necessary to open Matlab this way because it sets path settings to use Comsol libraries.

12: In Matlab, choose File -> Open... and navigate to your m-file you saved in Comsol.

13: In the script editor, hit F5 to run it. When it asks you if you want to change your path, choose "Change Current Directory" (the default) and hit ok.

A loading window should appear, and then a graph of temperature. Also, you should now have two variables I1 and I2 which represent the two boundary integrals in the order they were performed (left boundary, then the convection boundaries).

Scripting Details

Now let's look at our script.

Comsol comments fairly well, so you can see each step annotated in the file. The order things are written out in is:

- Some version information and a statement which generates a new world.

- The drawing we did in step 2.

- Step 5's meshing and mesh refining.

- Data from step 1 (the type of problem to be solved) and step 4 (the boundary settings).
For example, bnd.h is the h setting that we applied for convection.
This section is a bit complicated, so I'll return to it in a minute.

- The subdomain settings from step 3.

- "fem.lib = lib;" and Multiphysics
More boilerplate stuff. The script creates an object called fem which is sort of a big object containing everything, and so the assignment makes fem contain the library information on the problem, while the multiphysics section further initializes the fem object.

- Extend mesh: This activates the mesh the rest of the way, and may be a consequence of the mesh refinement.

- Solve Problem: This actually solves the system. Corresponds to step 6 above in our directions.

- Plot Solution: This step is part of solving. You're going to want to comment this section out later because it's obnoxious to have it plot every time it solves the system.

- Integrate (two sections) -- these correspond to our two boundary integrations. Notice that they are assigned to I1 and I2 -- you'll want to assign these to arrays later.

Boundaries in scripts:

In the "Application Mode 1" section, above library materials, there are these lines (if you did the rectangle):

```
clear bnd
bnd.type = {'q','T'};
bnd.h = {10,0};
bnd.T0 = {0,400};
bnd.Tinf = {300,0};
bnd.ind = [2,1,1,1];
appl.bnd = bnd;
```

What is happening here is that Comsol creates an array of boundary types, in this case two. The first boundary is the first index of each of these variables (except bnd.ind), while the second boundary is the second index. You can see that the first boundary is type 'q', has an h of 10, no T0 value (it's not relevant to a heat flow boundary) and a Tinf value of 300 (the air temperature). The second boundary is the constant temperature boundary. bnd.ind assigns each boundary of the drawing in order to one of these two boundary types.

Therefore, if you want to vary h, you would vary the first index value of bnd.h.

Wrap it all in a for loop

So, what can we do with all of this? We can wrap it in a for loop and make it change the geometry.

First, comment out the section "Plot Solution" as they will be very irritating later. Select them and hit ctrl+R to do it quickly.

With our rectangle, the command we're working with is in the Geometry section. Look at the first line:

```
g2=rect2(1,0.6,'base','corner','pos',[-0.6,-0.4]);
```

What this is doing is making a rectangle which is 1 unit long by .6 units high and begins at the point [-.6,-.4]. Let's vary our width for this example. Change the line to as follows:

```
g2=rect2(w,0.6,'base','corner','pos',[-0.6,-0.4]);
```

Now, at the top of the file (above the first comments), enter this:

```
I1 = [ ];
I2 = [ ];

for w = [.1:.1:1]
```

We will be varying w (the width of the rectangle) from .1 by increments of .1 up to 1.

Select the entire rest of the file and hit ctrl+] (or select Text -> Increase Indent).

At the end of the file put this:

```
end
```

Now, inside the integrate sections, change I1 to I1(end+1) and I2 to I2(end+1). This will save each new integral in a new spot in the array.

Hit F5 to save and run the new script. This will take a minute or two to run.

Now, in the matlab main window, plot I1 and I2 -- they should both be about equal in magnitude (by conservation of energy since the system is at steady state and we're measuring flow in and flow out) and should both increase as the width increases.

Congratulations! You've just made Comsol work inside Matlab.

Now, there are lots of more clever things you can do than what I have done here. First of all, don't be afraid to change the names of w, I1 and I2 to make them more human readable. Second, you can do more clever loops than just a for loop. A simple example to optimize would be to do a hill climbing algorithm -- see Wikipedia's entry on it for more info, or this Matlab script: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=1339&objectType=file

If you don't want to look at the guts of the file all the time, another clever thing to do is to make the whole thing a function. I've implemented this as a function which returns the two integral and takes the width, height, air temperature, boundary temperature and h value that you want to solve for. I've included this at the end of this document. Run "help function" in Matlab for more information on writing function files

******* The altered version the Comsol m file, for loop included ********

```
% Remember, this won't work without loading "Comsol with Matlab" from the start
menu.
I1 = [];
I2 = [];
for w = [.1:.1:1]

   % COMSOL Multiphysics Model M-file
   % Generated by COMSOL 3.2b (COMSOL 3.2.0.304, $Date: 2006/04/04 14:56:13
$)

   flclear fem

   % COMSOL version
   clear vrsn
   vrsn.name = 'COMSOL 3.2';
   vrsn.ext = 'b';
   vrsn.major = 0;
   vrsn.build = 304;
   vrsn.rcs = '$Name:  $';
   vrsn.date = '$Date: 2006/04/04 14:56:13 $';
   fem.version = vrsn;

   % Geometry
   g2=rect2(w,0.6,'base','corner','pos',[-0.6,-0.4]);
   clear s
   s.objs={g2};
   s.name={'R1'};
   s.tags={'g2'};

   fem.draw=struct('s',s);
   fem.geom=geomcsg(fem);

   % Initialize mesh
   fem.mesh=meshinit(fem);

   % Refine mesh
   fem.mesh=meshrefine(fem, ...
                    'mcase',0, ...
                    'rmethod','regular');

   % (Default values are not included)

   % Application mode 1
   clear appl
   appl.mode.class = 'HeatTransfer';
   appl.assignsuffix = '_ht';
   clear prop
   prop.analysis='static';
   appl.prop = prop;
   clear bnd
   bnd.type = {'q','T'};
   bnd.h = {10,0};
   bnd.T0 = {0,400};
   bnd.Tinf = {300,0};
   bnd.ind = [2,1,1,1];
   appl.bnd = bnd;
   clear equ
   equ.k = 'mat1_k';
   equ.rho = 'mat1_rho';
   equ.C = 'mat1_C';
   equ.ind = [1];
   appl.equ = equ;
   fem.appl{1} = appl;
   fem.frame = {'ref'};
   fem.border = 1;
   fem.outform = 'general';
   fem.units = 'SI';

   % Library materials
   clear lib
   lib.mat{1}.name='Iron';
   lib.mat{1}.varname='mat1';
   lib.mat{1}.variables.sigma='1.12e7';
   lib.mat{1}.variables.mur='4000';
   lib.mat{1}.variables.k='76.2';
   lib.mat{1}.variables.epsilonr='1';
   lib.mat{1}.variables.rho='7870';
   lib.mat{1}.variables.C='440';
   lib.mat{1}.variables.nu='0.29';
   lib.mat{1}.variables.alpha='12.2e-6';
   lib.mat{1}.variables.E='200e9';


   fem.lib = lib;

   % Multiphysics
   fem=multiphysics(fem);

   % Extend mesh
   fem.xmesh=meshextend(fem);

   % Solve problem
   fem.sol=femnlin(fem, ...
                'solcomp',{'T'}, ...
                'outcomp',{'T'});

   % Save current fem structure for restart purposes
   fem0=fem;

   % % Plot solution
   % postplot(fem, ...
   %          'tridata',{'T','cont','internal'}, ...
   %          'trimap','jet(1024)', ...
   %          'title','Surface: Temperature [K]', ...
   %          'axis',[-1.53406326034063,1.53406326034063,-1,1,-1,1]);

   % Integrate
```

```
   I1(end+1)=postint(fem,'nflux_ht', ...
             'dl',[1], ...
             'edim',1);

   % Integrate
   I2(end+1)=postint(fem,'nflux_ht', ...
             'dl',[2,3,4], ...
             'edim',1);
end
```

******* Implemented as a function. Save as rectangleHeatFlow.m to use *******

```
function [I1, I2] = rectangleHeatFlow(width,height,Tinf,T0,h)
% Calculates the heat flow through the boundaries an iron rectangle with specified
properties.
% Requires running "Comsol with Matlab" to work -- needs path information
%
% [I1, I2] = rectangleHeatFlow(width,height,T_inf,T_0,h)
%
% width: Width of the rectangle
% height: Height of the rectangle
% Tinf: The air temperature for convection
% T0: The constant temperature for the left boundary
% h: The convection coefficient
%
% I1: The integral of the heat flowing through the constant temperature
% boundary
% I2: The integral of the heat flowing through the convection boundaries
%
% Written by Matt Aasted
% Last edited 2/18/07


   % COMSOL Multiphysics Model M-file
   % Generated by COMSOL 3.2b (COMSOL 3.2.0.304, $Date: 2006/04/04 14:56:13
$)

   flclear fem

   % COMSOL version
   clear vrsn
   vrsn.name = 'COMSOL 3.2';
vrsn.ext = 'b';
vrsn.major = 0;
vrsn.build = 304;
vrsn.rcs = '$Name:  $';
vrsn.date = '$Date: 2006/04/04 14:56:13 $';
fem.version = vrsn;

% Geometry
g2=rect2(width, height,'base','corner','pos',[-0.6,-0.4]);
clear s
s.objs={g2};
s.name={'R1'};
s.tags={'g2'};

fem.draw=struct('s',s);
fem.geom=geomcsg(fem);

% Initialize mesh
fem.mesh=meshinit(fem);

% Refine mesh
fem.mesh=meshrefine(fem, ...
                  'mcase',0, ...
                  'rmethod','regular');

% (Default values are not included)

% Application mode 1
clear appl
appl.mode.class = 'HeatTransfer';
appl.assignsuffix = '_ht';
clear prop
prop.analysis='static';
appl.prop = prop;
clear bnd
bnd.type = {'q','T'};
bnd.h = {h,0};
bnd.T0 = {0,T0};
bnd.Tinf = {Tinf,0};
bnd.ind = [2,1,1,1];
appl.bnd = bnd;
clear equ
equ.k = 'mat1_k';
equ.rho = 'mat1_rho';
equ.C = 'mat1_C';
equ.ind = [1];
appl.equ = equ;
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
fem.outform = 'general';
fem.units = 'SI';

% Library materials
clear lib
lib.mat{1}.name='Iron';
lib.mat{1}.varname='mat1';
lib.mat{1}.variables.sigma='1.12e7';
lib.mat{1}.variables.mur='4000';
lib.mat{1}.variables.k='76.2';
lib.mat{1}.variables.epsilonr='1';
lib.mat{1}.variables.rho='7870';
lib.mat{1}.variables.C='440';
lib.mat{1}.variables.nu='0.29';
lib.mat{1}.variables.alpha='12.2e-6';
lib.mat{1}.variables.E='200e9';


fem.lib = lib;

% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem);

% Solve problem
fem.sol=femnlin(fem, ...
          'solcomp',{'T'}, ...
          'outcomp',{'T'});

% Save current fem structure for restart purposes
fem0=fem;

% % Plot solution
% postplot(fem, ...
%        'tridata',{'T','cont','internal'}, ...
%        'trimap','jet(1024)', ...
```

```
    %          'title','Surface: Temperature [K]', ...
    %          'axis',[-1.53406326034063,1.53406326034063,-1,1,-1,1]);

    % Integrate
    I1=postint(fem,'nflux_ht', ...
            'dl',[1], ...
            'edim',1);

    % Integrate
    I2=postint(fem,'nflux_ht', ...
            'dl',[2,3,4], ...
            'edim',1);
end
```