MATLAB®/PDEToolHints

Note - for best results, print this page. It is meant to be used as a handy reference when getting started on MATLAB®. Please ignore any sarcastic comments.

Preface

In this document, we (the instructors) are trying to give you (the students) some simple instructions for getting started with the partial differential-equation (PDE) toolbox in MATLAB®. PDEs and their solutions are applicable to many engineering problems, including heat conduction. There are several "pieces" to any PDE problem, and there are a few more to solving them numerically. These are:

- defining the domain of the problem in space (i.e. the geometry)
- defining the time-dependence of the problem, or lack thereof if steady state
- defining the initial and/or boundary conditions
- setting up the proper PDE for solution
- choosing the "coefficients" or physical properties of the system
- defining the "mesh" or set of sub-domains on which to numerically solve the PDE

Luckily for you, the PDE toolbox in MATLAB® has all of these functions laid out nice and neatly for you. Not only that, but you can also save everything you do to a MATLAB® script file so that you won't have to repeat all the steps every time.

The good stuff

Getting started

To get started solving a problem, start MATLAB® (I choose to do this from the menu bar across the top of the screen). Once there, you can type "pdetool" at the prompt. In a few seconds, MATLAB® will open up a graphical user-interface (GUI) for you to work in.
You will notice that there is an empty grid in front of you. If you want to change any characteristics of the grid, such as the min. or max. values or increments, or the number of gridlines or the "snap" characteristics of the drawing tools, go to the Options heading on the toolbox menu. From there, things are pretty self-explanatory for anyone born after 1960.

Geometry
To define the geometry on which you want to solve your PDE, you can simply go to the Draw menu. Again, there are some simple shapes to choose from, and youngsters such as yourselves should have no problem figuring out how to make circles and squares on the grid.

For more complex geometries, such as a square with a hole in the middle, you can draw a square and a circle positioned as you desire. You'll notice in the Set formula window that MATLAB® defines the domain as "square + circle" - you can simply change that to "square - circle" in order to create the desired hole.

Boundary conditions
To set the conditions on the boundary of your geometry, go to the Boundary menu. Before specifying a boundary condition, you must inform MATLAB® which boundary you want by clicking on it with the mouse. Boundaries turn black when selected, and you can select multiple boundaries if they are all to have the same boundary condition.

There are basically two types of boundary conditions that we deal with - constant temperature and constant heat flux (including zero heat flux, known as adiabatic). The constant temperature condition is a "Dirichlet condition" and the constant heat flux condition is a "Neumann condition." In MATLAB®, the variable "u" represents temperature for our purposes. For a Dirichlet condition, you should set the coefficient "h" equal to unity and the coefficient "r" to whatever constant temperature

you desire. For a Neumann condition, you should set the coefficient "q" to zero and the coefficient "g" to the negative of the desired heat flux (or zero if adiabatic). The coefficient "c" is the thermal conductivity of the material and will be specified by you later.

PDE specification
To define the governing equation, now go to PDE on the menu bar. Selecting PDE specification, you will see a variety of choices. For our purposes, an elliptic equation describes describes a steady-state problem while a parabolic equation describes a transient problem. Forget about hyperbolic and eigenmodes.

If the problem is steady-state, you can specify the thermal conductivity of the medium by giving "c" a value; "a" should be set to zero, and "f" is representative of any internal heat generation. If the problem is transient, the coefficient "d" represents the product of the density and specific heat of the medium.

Initial conditions
I'm jumping ahead of the GUI menu here, but it makes sense to discuss how to set initial conditions for the problem if it is transient. You'll go to the Solve menu and choose Parameters. Here, you can specify the time domain (duration and increments for solution), and the variable "u(t0)" represents the temperature on the entire geometry at time = 0, which is up to you to specify (i.e. the initial condition).

The mesh
The next step is to define the mesh - basically, divide the geometry into discrete and compuationally manageable chunks, much like you divided the time domain into little bits for computation. MATLAB® does this automatically when you select the Mesh menu, and you can further Refine the mesh. The mesh should be refined to the point that a regular array of mesh elements covers the irregularities in the geometry ("resolution") but not to the point that there are so many elements that you get marginally better answers for increased solution time ("point of diminishing returns"). You can also Jiggle the mesh to knock the triangular elements into better arrangement.

When you have completed the mesh, you should Export the mesh to MATLAB®, so that you can post-process your results. Unfortunately, the exporting process is one command that MATLAB® does not save to a script file, so you will have to do this manually every time you run a problem.

The solution
To solve the problem - guess how? You got it! Go to the Solve menu. Wow, you MIT engineers are pretty sharp. Again, the selection of Parameters allows you to specify characteristics of the time domain for unsteady problems. After MATLAB® gives you a pretty plot of the temperature solution, you can also export this solution for post-processing. Again, the export step is not saved in your script file, so you'll have to do it manually every time.

Extras
Under the Plot menu, you can change the type of plot you get on the GUI. For example, when you go to Parameters, a color plot of "u" is the default - this is a color plot of the temperature profile. A contour plot of "u" gives you isotherms, a plot of "-c*grad(u)" gives you heat flux, etc.

Post-processing
For your convenience, Professor Lienhard has created a sample MATLAB® script file called bredq.m in the MATLAB directory in the course locker. It will make some plots for you, and there are enough comments that you should be able to modify it to suit your individual needs.

Repeating your steps
If you have created a script file of your session in the toolbox, you should also be able to edit that fairly easily. Being MIT students, you should be able to figure out which command lines correspond to which functions and you can edit the file to match the needs of the problem you're interested in solving. Professor Lienhard has also created a sample problem called aheat.m which is also in "MATLAB®" directory in the course locker. Any script file can be run by simply typing the file name

(without the ".m" extension) at the prompt in MATLAB®. You can copy this into your directory and modify it for the problems you'd like to solve.

Post-mortem

You now have pretty much everything you need to start solving heat-conduction problems on MATLAB®. Again, there are sample files in the course locker and the various commands in these are fairly self-explanatory. If not, MATLAB® itself has a superb "help" function built into the program - how do you think we learned to do all this stuff?

Text author: *Peter Noymer*