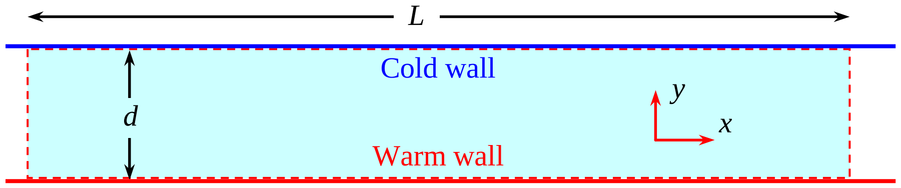
Gambit – Generation of grid for the Bénard problem

Author: John M. Cimbala, Penn State University
Latest revision: 12 February 2008

Introduction and Instructions:
In this document is a procedure that enables you to generate a two-dimensional, structured rectangular grid that will be used with the CFD program Fluent to solve for the thermal instability between two plates – the Bénard problem.

Overall description of the problem and grid:

1. The computational domain for this problem is very simple – just a rectangle, as sketched below.



2. In the sketch, *x* is in the horizontal direction and *y* is in the vertical direction. Gravity acts downward (in the negative *y* direction).
3. The bottom wall is hotter than the top wall, so that a thermal instability is possible, depending on the Rayleigh number.
4. For all cases the distance *d* between the two walls is set to 2.00 mm (0.00200 m). However, domain length, *L* varies with each student. [Each student is assigned a different value of length *L*.]
5. In other words, each student is assigned a different aspect ratio for his/her computational domain, where aspect ratio is defined as AR = *L*/*d*, according to the following table:

Student PSU E-mail	**Assigned AR = *L*/*d***
aem277	7.0
axa962	7.2
bkj116	7.4
dxs961	7.6
edr141	7.8
gdl121	8.0
ggb110	8.2
gom106	8.4
gun100	8.6
jdt195	8.8
jsh293	9.0
krs289	9.2
kxk952	9.4
kzc124	9.6
msy117	9.8
Prof. Cimbala	10.0
mts244	10.2
nrk134	10.4
nwm108	10.6
prs153	10.8
pxk918	11.0
pxo132	11.2
spf136	11.4
suw154	11.6
sxk388	11.8
szj121	12.0
szs205	12.2
vbk104	12.4
vcp109	12.6
vri102	12.8
wxl184	13.0

Log on and start Gambit

1. Log onto any Windows or Linux computer that has a valid license for the Ansys-Fluent family of programs – Gambit (grid generator) and Fluent (CFD solver).
2. Start Gambit. *Note*: If Gambit does not open properly, you may not have the proper path set up in your user files. If this is the case, check with the system administrator.
3. In Gambit, when the mouse cursor is placed over a button, a short description of the function of that button appears in the lower right window under *Description*. This is useful for understanding the commands.

Create some data points (vertices) for the desired geometry:

1. First, create a vertex at the lower-left corner of the computational domain. In *Geometry*, Vertex Command Button. Under *Vertex*, right mouse click the top left icon called Create Vertex and then From Coordinates. (In this learning module from now on, this type of command which requires a right click of the mouse is preceded by "R-". Here, for example, R-Create Vertex-From Coordinates).
2. In *Create Real Vertex-Global*, enter 0, -d/2, and 0 for x, y, and z coordinates respectively, where d is the distance between the upper and lower channel walls in meters. Type a label for this vertex if you wish, and Apply. A vertex is created, as indicated by a small plus sign at this location.
3. Vertices need to be created at the other three corners of the domain. Create vertices with coordinates (0,d/2,0), (L,d/2,0), and (L,-d/2,0), where L is your assigned horizontal length of the computational domain.
4. Zoom in to see the vertices more clearly. The easiest way to do this is Fit to Window in the *Global Control* area on the lower right of the Gambit screen.
5. In the main Gambit window near the upper left, File-Save. This will save your work so far. It is a good idea to do this every so often, especially after a major task is completed.

Create edges from these vertices to define the computational domain:

1. Edges need to be created to define the domain. Under *Geometry*, Edge Command Button. (*Note*: If the option is already open, clicking it again will make the options disappear! If this happens, click it again.)
2. Under *Edge*, R-Create Edge-Straight. The straight edge option should be the default.
3. Select (shift + left mouse click) the two bottom-most vertices, type in a label ("bottom" is suggested), and Apply. A yellow line should appear indicating successful creation of this edge.
4. Similarly, create a straight edge called "top".
5. Carefully create two other straight edges called "left" and "right". *Note*: When creating the left and right edges, be sure to choose the bottom vertex first, and then the top vertex, so that the edge direction is *up* for both of these edges. This is important because these edges will be defined with periodic boundary conditions, and they must therefore be oriented identically. Failure to follow these instructions will result in disaster.
6. Close the *Create Straight Edge* window.

Generate a face defining the computational domain:

1. Under *Geometry*, Face Command Button-R-Form Face-Wireframe.
2. *Note*: I like to select edges in mathematically positive counterclockwise order. Select the four edges just created. These edges outline a closed face.
3. In *Create Face From Wireframe*, type in a label for this face if desired ("domain" is suggested), and Apply. If all went well, a pretty blue outline of the face should appear on the screen; this is a face, which is now ready to be meshed.

Link the right and left edges so they can be defined as periodic:

1. We want the left and right edges to be periodic, which means that whatever fluid flows *out of* or *along* the right edge goes *into* or *along* the left edge. To do this properly, these two edges must be linked. In *Operation*, Mesh Command Button-Edge Command Button.
2. In *Edge*, R-Link/Unlink Edges-Link Edge Meshes.
3. In *Link Edge Meshes*, select the left and right edges. The arrows that appear on the selected edges must be in the *same direction* (up) for this to work properly, as mentioned previously.Verify that the Periodic option is turned on. Apply. Close.

Specify the boundary types on all edges:

1. In order for the mesh to be properly transferred to Fluent, the edges must be assigned boundary types, such as wall, axis, etc. In *Operation*, Zones Command Button-Specify Boundary Types Command Button.
2. In the *Specify Boundary Types* window, change *Entity* to Edges (which may or may not be the default). In this problem, which is 2-D, boundary conditions are applied to edges rather than to faces.
3. Select the top-most edge of the computational domain. Change its *Type* to Wall, if necessary (that is usually the default) and type in the name “top” or “top_wall” or “upper_wall” or something equally descriptive. Apply. Turn on Show Labels to see that the boundary type was actually specified.
4. Similarly, make the bottom-most edge a wall called “bottom” or “bottom_wall” or “lower_wall”.
5. Select both the left and right edges, and assign their boundary type simultaneously as *Periodic*, with the label “periodic”.
6. Close the *Specify Boundary Types* window, and save your work.

Define node points along the edges:

1. In *Operation*, Mesh Command Button-Edge Command Button-Mesh Edges. *Mesh Edges* should be the default window that opens; if not, Mesh Edges.
2. Select the two horizontal edges (top and bottom), and in *Mesh Edges*, change the *Spacing* option from Interval Size to Interval Count. Enter the desired number of node points (some value between 100 and 150 is recommended here) as the *Interval Count*.
3. No clustering or bunching of nodes is required in this problem, so keep *Ratio* (in the *Grading* section of the *Mesh Edges* window) as 1.0. Apply. Blue circles should appear at each created node point along these two edges.
4. Now select the two vertical edges, and apply nodes to each of these (between 20 and 30 nodes are recommended).
5. Now all the necessary edges have been assigned nodes; Close the *Mesh Edges* window. Save your work.

Generate the mesh on the face:

1. Under *Mesh*, Face Command Button. The default window that pops up should be *Mesh Faces*. If not, Mesh Faces.
2. Select the face by shift clicking on one of its edges. *Elements* should be Quad by default; if not, change it. Also change *Type* to Map if necessary. The *Spacing* options will be ignored since nodes have already been defined on the appropriate edges of this face.
3. Generate the mesh by Apply. If all goes well, a structured mesh should appear.
4. You can now Close the *Mesh Faces* window.

Write out the mesh in the format used by Fluent, and then exit Gambit:

1. In the main Gambit window, File-Export-Mesh. (The default file name can be changed at this point if desired.) Check the option to export a 2-D mesh, and Accept.
2. When the Transcript (at lower left) informs you that the mesh is done, File-Exit-Yes.
3. The mesh file should now be ready for use by Fluent.