

# Fluent – Secondary flow in a teacup

Author: John M. Cimbala, Penn State University  
Latest revision: 29 January 2008

## Introduction and Instructions:

This learning module contains a procedure to solve for laminar flow inside a cup of water which is swirling around the cup's axis of symmetry. The CFD program Fluent is used as the solver. The solution illustrates secondary flow phenomena. *Note:* This set of instructions assumes that you have already run Gambit, and have generated an appropriate grid. The file "teacup.msh" is presumed to exist on your directory.

## Log on and launch Fluent:

1. Log on to a computer that has access to the Fluent software.
2. Start Fluent, and select 2ddp, the two-dimensional, double precision option. Run. [*Note:* If you are using a Unix or Linux machine, go to the desired directory, and enter the command fluent 2ddp & (the & symbol lets Fluent run in background mode so that the Unix shell is still usable).]
3. After a few seconds, the main *Fluent* window should appear on your screen.

## Read the grid points and geometry of the flow domain:

1. Select File-Read-Case. Browse as necessary to locate the mesh file teacup.msh (or whatever you named it) that was previously created by Gambit. [On Windows machines, the default may be the C:\Temp directory.] Highlight this file (click on it), and OK. Fluent will read in the grid geometry and mesh, and the word "Done" should appear. Some warnings may appear about the axis boundary condition.
2. Check the validity of the grid: Grid-Check. If the grid is valid, no errors should appear. If there are errors, you may have done something wrong in the grid generation, and will have to go back and regenerate the grid.
3. Look at the grid. Display-Grid-Display. A new graphical display window opens up showing the grid. If this window is too big or too small, re-scale it by dragging the edges of the window. It is best if the graphical display window is small enough that both it and the *Fluent* window are visible simultaneously. The *Fluent* window and/or the graphical display window may need to be moved to accomplish this.
4. The graphical display can be zoomed-in or zoomed-out with the *middle* mouse button. If you start on the *lower left* and draw a rectangle with the middle mouse button towards the upper right, the display zooms in on what is included in the rectangle. Meanwhile, the *left* mouse button can be used to drag the image to a new location. If you draw a rectangle *backwards* with the middle mouse button, from right to the left, it zooms out. Zoom in if necessary until the grid is shown nicely in the window. Close the *Grid Display* window; the display itself will remain.

## Define the problem as axisymmetric with swirl:

1. In the main *Fluent* window, Define-Models-Solver. In the *Space* region of the *Solver* window, choose Axisymmetric Swirl. Keep in mind that Fluent automatically assumes that the *x*-axis is the axis of symmetry in any axisymmetric flow problem. OK. The *Solver* window should disappear.
2. Again, Define-Models-Solver to open up the *Solver* window. Use the Help button in the *Solver* window, if available, to read about axisymmetric swirling flows. ***There are some specific recommendations and suggested solution strategies which you should read.*** If you have trouble getting the solution to converge, make some of the recommended changes before continuing – the most important of which is to use the "PRESTO" pressure discretization scheme by Solve-Controls-Solution, pick PRESTO!, and OK. Write down the changes that you made. You may also find that you achieve the best convergence by lowering the under-relaxation factors somewhat (also from Solve-Controls-Solution).

## Generate a line at the mid-plane of the domain:

1. The swirl velocity profile will be examined in detail at three *x* locations, namely at  $x = 0.0$  m (the bottom of the cup),  $0.025$  m (the mid-plane of the cup), and  $0.05$  m (the top of the cup). Lines already exist for the bottom and top of the cup. A line representing the mid-plane of the cup needs to be defined.
2. In the main *Fluent* window, Surface-Line/Rake. Type in the desired starting and ending *x* and *y* locations of the vertical line – a vertical line going from (0.025,0) to (0.025,0.05).
3. The *New Surface Name* should be assigned at this point. It is suggested that this line be called "mid plane" or something equally descriptive of its intended purpose. Create to create the line.
4. To view this newly created line, return to the main *Fluent* window, and Display-Grid-Display. Unselect (by left mouse click) the default interior, and select the newly created line instead. Display. The line should be visible at the appropriate location. If not, create it again more carefully.
5. Close both the *Line/Rake Surface* window and the *Grid Display* window.

### Define the fluid as liquid water:

1. The default fluid is air, which must be changed to water. In the main *Fluent* window, Define-Materials-Fluent Database. Select water-liquid from the *Fluid Materials* drop down list. Copy.
2. Write down the density and viscosity of liquid water, which may be needed later to calculate Reynolds numbers, etc. Close.
3. Also Close the *Materials* window. *Caution*: This has added liquid water into the list of fluids available as boundary conditions, but has not actually changed the fluid from air to water. This needs to be done when specifying the boundary conditions.

### Define the boundary conditions:

1. Now the boundary conditions need to be specified. In Gambit, the boundary conditions were declared, but actual values for wall velocities, etc. were never defined. This must be done in Fluent. In the main *Fluent* window, Define-Boundary Conditions, and a new *Boundary Conditions* window will pop up. Select fluid. Set. Choose water-liquid from the list of material names. OK.
2. Select outer wall (the top-most edge of the computational domain). The boundary condition is a wall, but Fluent's default is a stationary wall. We want this wall to be rotating about the *x*-axis. Set. In the *Wall* window that pops up, check the box for moving wall, and select Absolute and Rotational. Enter the angular velocity as 5 radian/second.
3. Repeat for the wall called top (the right-most edge of the computational domain). In this problem, we are setting the angular velocity of both the outer wall and top walls to the same value. This will impart a bulk swirling motion to the water in the cup. [You can also use the Copy command here if you want.]
4. Select the wall called bottom. The proper boundary condition on this wall is the default – a stationary wall, so that the secondary flow can be established. [Nothing needs to be done to this wall boundary condition.]
5. Nothing needs to be done to the axis boundary condition, so Close the *Boundary Conditions* window.

### Set up some parameters and initialize:

1. In the main *Fluent* window, Solve-Monitors-Residual. In the *Residual Monitors* window that pops up, turn on Plot in the *Options* portion of the window. The Print option should already be on by default. Here, Print refers to text printed in the main *Fluent* window, and Plot causes the code to plot the residuals on the screen while the code is iterating.
2. Since there are four differential equations to be solved in an axisymmetric incompressible laminar flow problem with swirl, there are four residuals to be monitored for convergence: continuity, *x*-momentum, *y*-momentum, and *z*-momentum (swirl). The default convergence criteria are 0.001 for all of these. Experience has shown that this value is generally not low enough for proper convergence. Click on each 0.001 individually, and change it to 0.000001 (or 1.E-06).
3. When all the residuals are set, OK to exit the *Residual Monitors* window.
4. In the main *Fluent* window, Solve-Initialize-Initialize. The default initial values of velocity and gage pressure are all zero. The velocities need to have non-zero values, or else the code will not iterate properly. Set the velocities to some small value like 0.1 m/s. Init and Close.
5. At this point, and every so often, it is wise to save your work. In the main *Fluent* window, File-Write-Case & Data. If not on by default, turn on the option to Write Binary Files (to save disk space). To save even more disk space, the files can be compressed by adding a “gz” at the end of the file name. In the *Select File* window that pops up, name the file “teacup.cas.gz” (or something equally descriptive). Note that “Case & Data” refers to both the *case* (the grid plus all boundary conditions and other specified parameters) and the *data* (the velocity and pressure fields calculated by the code). The code actually writes out *two* files, teacup.cas.gz and teacup.dat.gz.
6. Click OK to write the file onto your directory.

### Iterate towards a solution:

1. In the main *Fluent* window, Solve-Iterate to open up the *Iterate* window. Change Number of Iterations to about 500, change Reporting Interval to 10, and Iterate. The main screen list the residuals after every ten iterations, while the graphics display window plots the residuals as a function of iteration number. The residuals may rise at first, but should slowly start to fall, and should eventually level out. However, this problem does not converge well, so don't expect residuals down to  $10^{-6}$ .
2. The velocity vectors can be visualized at any time. In the main *Fluent* window, Display-Vectors-Display. The graphical display window should show the velocity vectors. Zoom in with the middle mouse, as described previously, to view the velocity field in more detail if desired.

3. By default, the velocity vector display shows all three components ( $u$ ,  $v$ , and  $w$ ) of the velocity vector. For this problem, the swirl velocity is coming out of the screen at the user. Since the swirl component is large compared to the secondary flow components, the vector arrows can become distorted. Also, arrow head size is based on a fraction of the total vector length. So, in the *Velocity Vectors* window, Vector Options. Turn off the Z Component of the velocity vector. Apply and Close the *Vector Options* window.
4. Finally, adjust the *Scale* of the vectors appropriately so that the secondary flow in the boundary layer along the bottom wall of the cup is clearly visible (remember, this is the left edge of the domain). Display.
5. Iterate some more. (To restart the iteration, either find the *Iterate* window, which is probably hidden under some other windows at this point, or click again on Solve-Iterate to re-open the *Iterate* window.) *Caution*: This problem has much difficulty converging! You may need to try some of the suggestions given in the Help panels.
6. Grid adaptation is also useful in regions with poor resolution, but should be used only after the bulk of the flow has been set up. A convenient way to adapt is based on the secondary flow velocity gradient. Adapt-Gradient. Select the *Method* to be Gradient. Choose gradients of Velocity and choose Radial Velocity. Compute. The minimum and maximum gradients are displayed.
7. As a general rule of thumb, set the refine threshold to about 1/10 of the maximum gradient. Enter this value in the box called *Refine Threshold*. Generally, the coarsen threshold should be set to about 4 orders of magnitude smaller than the refine threshold.
8. Before actually adapting, Mark. The main *Fluent* window should indicate how many cells are identified for coarsening or refining. Note that cells from the original mesh can not be coarsened, even though they may be marked as such. Only cells previously refined can subsequently be coarsened.
9. Fluent provides a mechanism for users to see the location of cells targeted for coarsening or refining. In the *Gradient Adaption* window, Manage-Display.
10. When ready to actually adapt the grid, Adapt. Yes, hanging node mode is okay.
11. If desired, the grid can be displayed. Zooming in will show locations where the grid has been refined.
12. Iterate some more. The residuals will jump up at first after adaptation, but will eventually start to decrease again (hopefully!).
13. You may have to play around with under-relaxation factors, grid adaptation, etc. to get a reasonably converged and physically meaningful solution. You may need a few thousand iterations to converge. As mentioned above, the help pages give some hints for solving axisymmetric swirl problems.

#### Save your calculations:

1. In the main *Fluent* window, File-Write-Case & Data. OK. It is okay to overwrite the files, so OK.

#### Turn on gravity:

1. Gravity does not affect the solution in any way, but it is important for the particle tracking (discussed below). In the main *Fluent* window, Define-Operating Conditions.
2. Turn on Gravity, and enter  $-9.807 \text{ m/s}^2$  as the gravity component in the  $x$  direction. Leave the  $y$ -component as zero. (Remember, in this problem, the  $-x$  direction is “down” towards the bottom of our tea cup.)
3. The other parameters here are okay by default, so OK.
4. Run a few iterations to convince yourself that gravity has no effect on the flow solution.

#### Analyze the flow:

1. In the main *Fluent* window, Display-Vectors-Display. Zoom in on the boundary layer near the bottom wall (left edge of the domain). Is there a secondary flow? Adjust the scale factor as needed to make the velocity vectors clearly visible.
2. If the vector plot is not real clear due to grid adaption near the wall, you may wish to create some lines along which to plot the velocity vectors. Follow the line creation directions above, but create a couple lines at constant  $y$  locations (I suggest lines at  $y = 0.01$ ,  $0.015$ , and  $0.02$  m), and then plot the velocity vectors along these lines only. This will result in a nicer-looking plot.
3. Add your name, the date, and a title (Teacup Problem) to the bottom of the plot. This is done simply by clicking in the space below the default title at the bottom left of the plot. A cursor should appear, into which you can type some text.
4. When the vector plot is to your liking, generate a hardcopy of the vector plot. In the main *Fluent* window, File-Hardcopy. Change *Coloring* to Monochrome if you have only a black-and-white printer. If you have access to a color printer, select Color. Select the desired format. Generally, TIFF or JPEG work best on Windows machines, but PostScript is often better for Unix or Linux machines. Save. Name the file

“teacup\_velocity\_vectors.ext” or something equally descriptive, where the file extension *ext* should automatically be inserted. **OK** and **Close**.

5. In the main *Fluent* window, select **Plot-XY Plot**. A window called *Solution XY Plot* should open up. In this window, at the lower right corner, select (highlight) the surface we created called mid plane. Also select the surfaces called bottom and top.
6. In the upper left corner of the window, turn **off** Position on X Axis, and turn **on** Position on Y Axis. This will make the y-axis the y position on the plot, as desired.
7. In the upper middle part of that window, set Plot Direction to X = 0 and Y = 1. This will make the y-coordinate position appear on the y-axis, as desired for a standard velocity profile plot.
8. The upper right part of the window selects the variable to be plotted. The *Y Axis Function* is set automatically to Position, and should be left alone. For the *X Axis Function*, select Velocity and Swirl Velocity. **Plot**.
9. The swirl velocity should be zero at the bottom wall, and linear at the top wall, since that is how we defined those boundary conditions. What does the swirl velocity profile look like on the mid plane?
10. Save a plot of the swirl velocity profiles. In the main *Fluent* window, **File-Hardcopy**. It is again suggested that *Coloring* be changed to Monochrome unless you have access to a color printer. Name the file “teacup\_swirl\_velocity.ext” or something equally appropriate.
11. Play with some of the other graphical features, like **Display-Contours-Display**. Both pressure contours and velocity magnitude contours can be viewed. You can also look at the streamlines by choosing Velocity and Stream Function.

### **Inject some particles and track them:**

1. In the main *Fluent* window, **Define-Injections-Create**. In the *Set Injection Properties* window, a number of parameters describing the particle to be injected need to be entered.
2. First, a name (*Injection Name*) can be assigned to each particle. The *Injection Type* recommended is single, which is the default.
3. A *Material* must be selected from the database. *Fluent* does not have an option for tea leaves, so pick something heavy to ensure it sinks to the bottom.
4. Under *Point Properties*, choose an *x* and *y* location. I suggest injecting from around  $x = 0.005$  m and  $y = 0.015$  m. Other injection locations can be tried as well.
5. Scroll down under *Point Properties*, and choose a particle diameter. The default is a really tiny particle that will simply follow the flow. Set a larger particle diameter, so that the particle should sink to the bottom and then move towards the axis, just as in the physical experiment.
6. When all parameters are set, **OK**. This creates the injection point.
7. To actually track the particle path, **Display-Particle Tracks**. Select the injection point just created under *Release From Injections*, and **Display**. A line is drawn showing the particle path.
8. The Pulse option can also be used instead - it makes a kind of animation of the particle moving through the fluid.
9. Play with particle diameter and initial location to try to simulate the tea leaves in the experiment. You can create any number of particles and track their paths separately or simultaneously. Do the particles fall to the bottom and then get swept towards the axis as in the physical experiment? Note that if the particles are too small, they will just follow the flow, and if they are too big, they will just drop to the bottom like bricks. If they are the “proper” size, they should drop down slowly, and then move along the bottom wall towards the center of the cup (the axis).

### **Save your calculations and exit Fluent:**

1. In the main *Fluent* window, **File-Write-Case & Data**. **OK**. It is okay to overwrite the files, so **OK** again.
2. Exit *Fluent* by **File-Exit**.