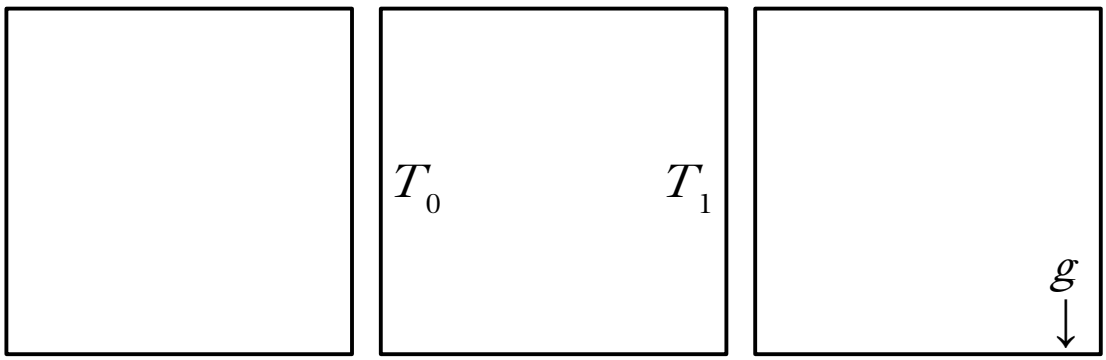
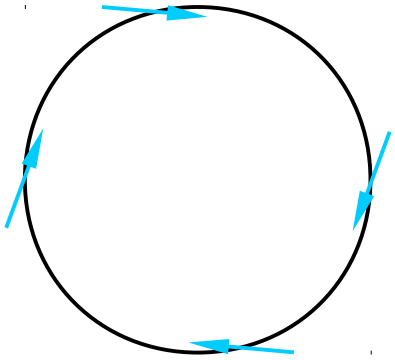
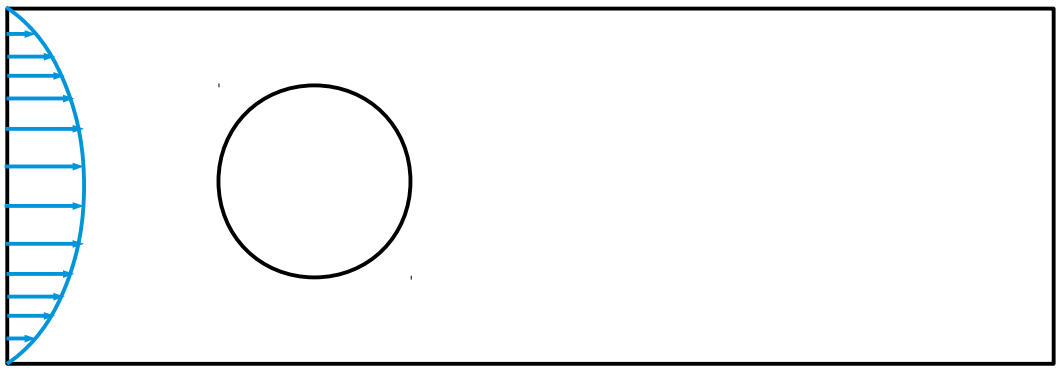
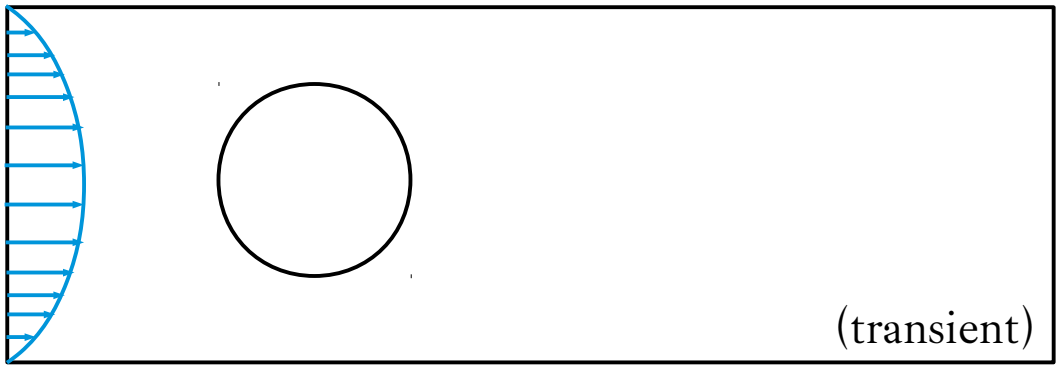
Symmetry Conditions

1. Draw a line of symmetry for the following geometries, if possible. (If not, specify "N/A".)
2. Label each geometry with the Fluent boundary condition types you would use in a simulation (make assumptions about inflow/outflow boundaries in such cases).



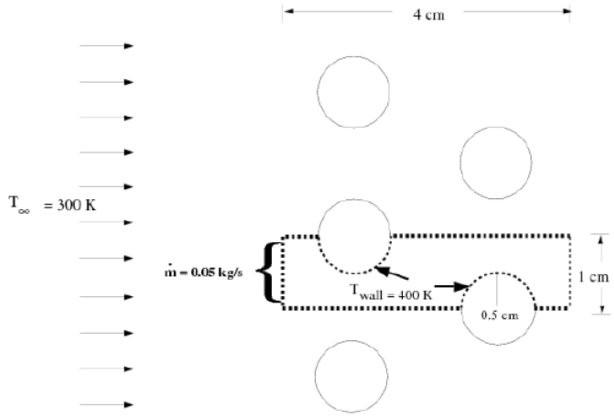






Heat Exchanger Tube Bank (Internal Flow with Symmetry)

Many industrial applications, such as steam generation in a boiler or air cooling in the coil of an air conditioner, can be modeled as two-dimensional periodic heat flow.

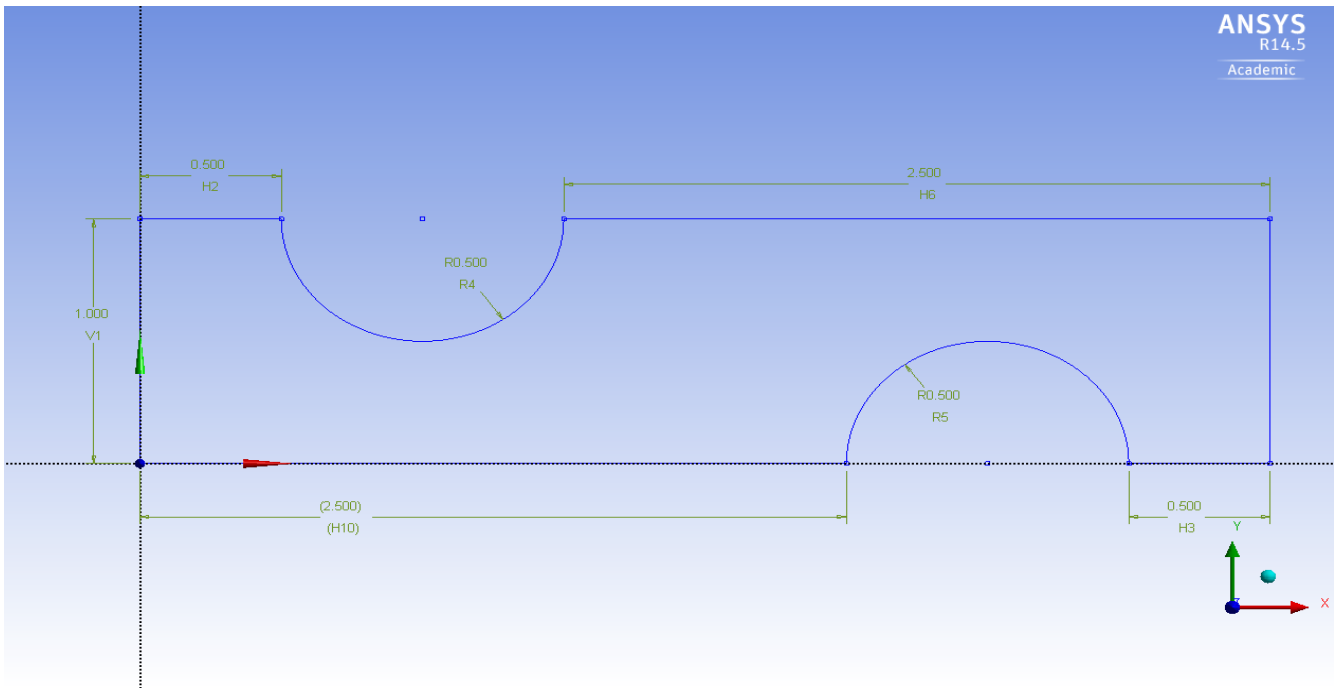


The model represents a bank of tubes containing a flowing fluid at one temperature that is immersed in a second fluid in cross flow at a different temperature. Both fluids are water, and flow is laminar and steady, with Re ~ 100. The model is used to predict the flow and temperature fields that result from convective heat transfer.

ρ = 998.2 kg/m^3
μ = 0.001003 kg/m-s
c_p = 4182 J/kg-K
k = 0.6 W/m-K

Due to symmetry and periodicity of the flow inherent in the tube bank geometry, only a portion of the geometry need be modeled. The resulting mesh consists of a periodic module with symmetry. The inlet boundary will be redefined as a *periodic zone*, and the outflow boundary defined as its *shadow*.

The bank consists of uniformly spaced tubes with a diameter of 1 cm which are staggered across the cross-fluid flow. Their centers are separated by a distance of 2 cm in the x-direction, and 1 cm in the y-direction. The bank has a depth of 1 m and a mass flow rate of 0.05 kg/s is applied at the inlet boundary of the periodic module. The tube wall temperature T_{wall} = 400 K and the bulk temperature of the crossflow water T = 300 K.

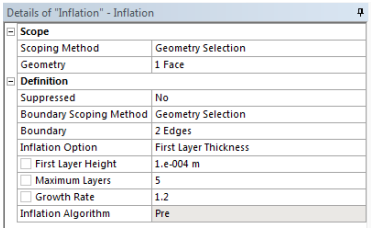


Geometry

- Open ANSYS Design Modeler and select "centimeter" as the length unit.
- Use line and arc drawing tools to make the base sketch.
- Next, use dimension tools to scale the sketch (dimensions shown in the picture).
- Lastly, generate a surface from the sketch.

Mesh

- Open the *Meshing* tool.
  Unfortunately, we cannot use a mapped face meshing for this geometry without partitioning, so we will utilize an unstructured mesh.
- Change the *Max Face Size* to 2.5×10^{-4} m. (Fluent works in meters in the *Meshing* tool.)
- Generate and examine the mesh.
  It looks fine in general. However, we want finer mesh around the tubes to resolve the thermal gradient. Assigning refined grids near certain boundaries is called "Inflation" in Fluent.
- Select the *Mesh* in the *Outline* window and right click on it. *Insert→Inflation.*
  - In the *Details of Inflation*, select the two semicircles as the *Boundary*. (Hold the Ctrl key on the keyboard to select multiple edges.)
  - Change the *Inflation Option* to "First Layer Thickness", and enter "1e-4 m" for the *First Layer Height* and 5 for the *Maximum Layers*.
  - Keep the *Growth Rate* as 1.2.



- Now generate the mesh again.
  Notice the change around the tubes.
- Name the left boundary as "inlet", the right "outlet", the semicircles "tube_wall", top two edges "symmetry_top", and bottom two edges "symmetry_bottom".

Setup

- Open Fluent with double precision.
- *Models*: Energy
- *Materials*: water-liquid
- *Cell Zone Conditions*: change "surface_body" to "water-liquid"
- *Boundary Conditions*:
  - Set symmetric boundaries to type "symmetry"
  - Fix "tube_wall" temperature at 400 K.
  - Next, we will make the inlet and outlet boundaries periodic.
    - Find out their IDs (next to the boundary type).
    - Go to the *text user interface* and type in "mesh/modify-zones/make-periodic".
    - Enter the inlet ID for "Periodic zone", and hit enter.
    - Enter the outlet ID for "Shadow zone" and type "no" for "Rotational periodic"; twice type "yes".
      Note that a new Periodic Conditions... *button appears.*
    - Click on it to open the *Periodic Conditions* window.
    - Choose *Specify Mass Flow*, and enter 0.05 for the mass flow rate. Keep the rest of the settings as their default.

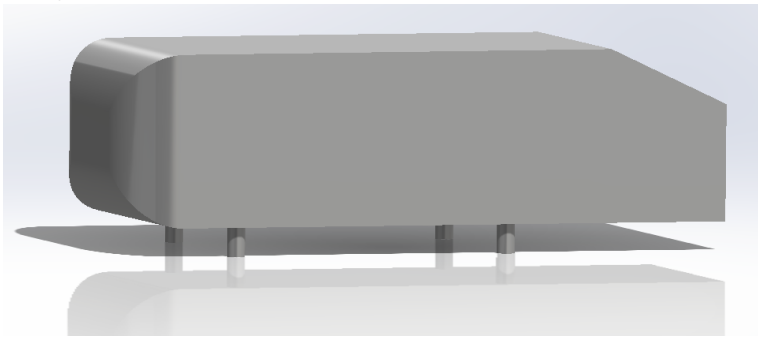
```
/mesh/modify-zones> make-periodic
Periodic zone [()] 7
Shadow zone [()] 8
Rotational periodic? (if no, translational) [yes] no
Create periodic zones? [yes] yes
Auto detect translation vector? [yes] yes

  computed translation deltas: 0.040000 -0.000000
  all 40 faces matched for zones 7 and 8.

  zone 8 deleted

  created periodic zones.
```

Solution Controls: change *Under-Relaxation Factors* of *Energy* to 0.9.
Initialization: standard
Run Calculation: 1000 iterations
Note how the residuals decrease.
Postprocessing

- In Fluent, you can display mirror images across a symmetric boundary.
  - Go to *Graphics and Animations* and click the *Views...* Views... button.
  - Select "symmetry_top" and "symmetry_bottom" in the *Mirror Planes* list, and *Apply*.
- Make a contour plot of temperature, of pressure, and of velocity.
- Plot the *y*-velocity and the temperature along the horizontal centerline.

Ahmed Car Body (External Flow with Symmetry)

The Ahmed car body is a standard validating case used by many aerodynamicists to test CFD models for flows past road vehicles. The model serves as a good benchmark, since a lot of data is available on fluid flow visualizations and lift and drag coefficients.



We will solve for the forces acting on an Ahmed car body traveling at a speed of 40 m/s.

Geometry

- Download the geometry for the car body from `go.illinois.edu/me498cf-ahmed`.
- Open the ANSYS Workbench and create a *Geometry* module. Import the Ahmed Car to it. *Probably the most challenging new feature of this exercise is that we have to create the surrounding fluid in the negative space around the car body. We will setup the problem in a similar way to a wind tunnel, where the object of interest is at rest and the fluid around it is moving.*
- Start the *Design Modeler*. Click on *Generate*.
  - In *Details of Import* in the *Operations* section, select *Add Material*. Go to *Tools* and select *Freeze*. Click on *Generate* again.
    This is essential when we create a body of fluid around our car body so that the 2 bodies do not merge. We will take the advantage of symmetry in our problem which lies at 0.2 m from the yz plane.
- Go to *Create* and then *New plane*. Create a plane at 0.2 m from the *yz* plane. Create another plane from *xz* at a distance of –0.195 m. The use of this plane will be justified later.
- Go to *Tools* and select *Enclosure*. We will stick with the box shape for our surround control volume.
  - In *Number of Planes* select 2 which implies that we have 2 symmetric planes. Assign "Plane 1" and "Plane 2" created in the above step as our symmetry planes
  - The *Cushion* helps us in setting the dimensions of our control volume. Enter the cushion lengths as shown. These may change substantially from one problem to another. It is important to check the results very carefully and examine if all the flow effects have been adequately captured.

Details View

Details of Enclosure4	
Enclosure	Enclosure4
Shape	Box
Number of Planes	2
Symmetry Plane 1	Plane4
Symmetry Plane 2	Plane5
Model Type	Full Model
Cushion	Non-Uniform
FD1, Cushion +X value (>0)	3 m
FD2, Cushion +Y value (>0)	3 m
FD3, Cushion +Z value (>0)	5 m
FD4, Cushion -X value (>0)	7.5 m
FD5, Cushion -Y value (>0)	7.5 m
FD6, Cushion -Z value (>0)	7.5 m
Target Bodies	All Bodies

- Rename the new solid body generated to "air".
  It is now important to subtract out Ahmed body from the fluid body such that we are only left with the fluid volume where the differential equations will be solved.

- Go to *Create* and *Boolean*. In *Operation* select *Subtract*. Enter "air" as our *Target body* and "Ahmed body" as *Tool body*.
  We have now ended up with just Air without the solid part of our Ahmed body. All the surfaces left act as stationary walls.
  Mesh
- Open the *Meshing* tool.
- Try the default mesh, just to see what Fluent generates.
- Let us now go through ANSYS meshing menu one by one.
  - Let us start with *Sizing*:
    - Select "Proximity and Curvature" in *Advanced Size Function*.
      This will refine our mesh based on curvature and proximity.
    - The *Relevance Center* is a tool which changes the minimum size of the mesh in the nature of its selection. You can adjust its importance from *Relevance*. (We can also set all the sizing parameters manually.)
    - Select "Active assembly" in *Initial seed size* since we have only one part in our geometry; and we set *Smoothing* to "High".
    - Leave *Span angle center* as "Fine".
    - Keep the *Curvature Normal Angle* at 12; it does not much affect this geometry.
    - In order to increase the number of cells between the body and the floor, we can increase the *Number of Cells across Gap*; leave it at 5.
    - The *Minimum size* is that of the meshing element. In order to get the idea of the size, you can generate a mesh and check if the size value is too big or small. We will keep it at "1 mm".
    - *Max face size* and *Max size* set the maximum possible size of the element which will be present at far away boundaries. For now we will keep both of them at "250 mm".
    - *Growth rate* affects how mesh grows from small elements to larger elements. We will leave it at 1.2.
- Click on *Generate Mesh* to see the updated version.
  This mesh appears far better than the default, but still wastes many elements across the domain. Thus we will limit the size of elements near the surface of the body.
- Right click on *Mesh* and then insert *Sizing*. We can select from "Body", "Face" or "Edge" sizing. We will select the two legs and limit the *Element Size* to "2 mm"; start a new *Sizing* and limit the mesh size on all the surfaces of our car body to "10 mm".
  We will now add an inflation layer, so that we can capture the boundary layer and its effect on our body. The inflation layer starts from the surface and then dissipates out into the main fluid. We will use prism elements.
- Fluent generally creates inflation layers intelligently. From the *Details of Mesh* menu, expand *Inflation*. In *Use Automatic Inflation*, select "Program Controlled". Click *Update*.
- Create some named selections for convenience:
  - Select the *Face Selection Tool* and then, from *Select Mode*, choose "Box Select". Zoom into the Ahmed body and capture all the surfaces associated with it. Right click and select *Create Named Selection*. Enter the name you wish. Click on the named selection created. In the details, select *Include* for "Program Controlled Inflation".

  - In a similar way, select the face of "Inlet" and create it as a named selection of "Velocity Inlet".
  - The opposite face should be named as "Pressure Outlet".
  - Name the symmetry plane as "Symmetry".
  - The top and side faces can be named as "Walls" or "Symmetry".
  - Call the bottom surface "Road".
- The default *Inflation Option* is "Smoothing", but we will select another method, "First Aspect Ratio". As referenced by the PDF file, we will let the default values for this method stay. In *View Advanced Options*, you can play around with different values, but for our purposes, we will only change *Smoothing Iterations* from 5 to 10.
- Update the mesh.

Problem Setup

- Return to the Workbench (save!) and transfer the data to a new Fluent module. Double precision, as usual.
- When you open the mesh, it should be color-coded by automatically recognized boundary zones: blue for inlet, red for outlet, white for wall, yellow for symmetry.
- First, check the mesh for consistency.
  - Click on *Check* to ensure that there is no error, and check the *Dimensions*. The minimum volume should always be positive or else Fluent will not solve your setup.
  - Click on *Scale* to match your dimensions and change units if you wish.
  - Click on *Report Quality* to see the status of *Orthogonality* and *Aspect ratio*.
- We will use a *Pressure based* solver since our flow is primarily incompressible.
- We will select the *Absolute Velocity Formulation* since we do not have any rotating components.
- The problem is steady-state.
- We are going to turn on a turbulence model for more realism, although we havent discussed those yet:
  - In *Models*, use *Viscous-Realizable k–ε* with *Non-equilibrium Wall Functions* which performs well with adverse pressure gradients.
- Double-check *Material Conditions* and *Cell Zone Conditions* to make sure they make sense.
- Moving onto *Boundary Conditions*, you can see that Fluent has already applied Boundary Conditions due to the names we had given previously. "Ahmed body" is assigned a no-slip wall boundary condition.
- We assign top wall and side wall as symmetric surfaces which only implies that they are stationary walls with no viscous effects. Due to the size of our domain, they should not have any effect on the Fluid Flow across the car body.
- Specify 5% *Intensity* and *Viscosity ratio* as 10 for *Pressure Outlet* and 1 % Intensity for *Velocity Inlet*.
- Specify a velocity of 40 m/s at the *Velocity Inlet* in the $-z$ direction.
- We will keep the road as a stationary wall, but you can change it to a moving wall to simulate real-world travel conditions.
- We do not require a dynamic mesh.
- For *Solution Controls*, the *Flow Courant Number* should be kept below 100 for skewed meshes—those which consist of tetrahedral elements—and *Relaxation Factors* should be

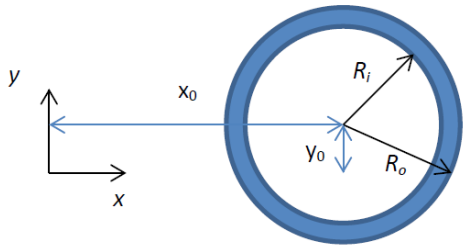
below 0.5 for *Momentum* and *Pressure*. Change the *Turbulent Viscosity Under-Relaxation* factor to 0.95.

- In the *Limits*, increase the *Maximum Turb. Viscosity Ratio* by 100 times so as to avoid warnings during iterations.
- We will select *Hybrid Initialization* which basically solves a one equation model to initialize our solution from the boundary conditions specified and is computationally useful.
- Run the simulation for 600 iterations.
  We want our residuals going down especially Continuity otherwise it means that our solution is diverging. It is highly possible for a 3D problem to fail due to poor quality of mesh. Whenever this happens, you will have to improve your mesh by increasing the number of elements or carrying out mesh refinement.
  Postprocessing
- Make a velocity contour plot projected onto the symmetry plane.
  - Also create a new plane where we want to view the velocity contours. In the pop-up window of contours, select *New Surface* and select "Plane" from the drop-down menu. We will create a plane which is shifted from the symmetry plane by 20 cm; hence, enter 0.2 in *x0* and 0 in other boxes for *Points*. The *Direction Vector* is the x-axis. Select the new "Plane" ("Plane-11" in this case) from the *Surfaces* box and click *Display*.
- Display pressure contours on the surfaces of the Ahmed body to find out which surfaces are creating the most drag.
- Let us now try to display path lines to captures some vortices in the wake of the car body.
  - Select *Pathlines* from the *Graphics* menu. Select *Symmetry* from the *Release from Surfaces* box. Enter 50 in the *Path Skip* box. (The lower this number, the more pathlines you see.) Color by "Velocity Magnitude".
- Similarly, display Vectors' to find out the direction of velocity vectors. Double click on 'Vectors' from the 'Graphics' menu. Enter '30' in 'Scale' box and '0' in 'Skip' box. Select the 'Symmetry' plane in 'Surfaces'.
- We can now see the direction of velocity vectors in the wake of the body.
-
- To verify our simulation, we will now check the y^+ contours on the car body. These should be low in value to indicate a high-quality mesh.
  - In *Contours*, select *Turbulence* and then select "Wall Yplus". Select "Ahmed body" in *Surfaces* and unclick "Global range". Then click *Display*.
- Plot the coefficient of pressure along the yz-plane.
  - Go to *Plots* in the *Results and Animation* section. In the *Y Axis Function* select *Pressure* and then "Pressure Coefficient".
  - Create a new surface. Select *Iso-Surface* from the *New Surface* menu. Select *Mesh* in surface of *Constant* drop down menu and then select "X-Coordinate". Pick "Ahmed body" from the *From Surface* box. Click *Create*.
  - In the *Plot Direction* enter 1 for Z, 0 otherwise. Select the x-coordinate plot name which you gave in the *Surfaces* box and then click on *Plot*.

  Conclusion
- Submit your plots to `me498admin@mechse.illinois.edu` with subject "`flec04 lab`".
  You may retain this hard copy.

Taylor–Couette Flow (User-Defined Function Implementation)

Consider a two-dimensional journal bearing layed out as at right. The bearing contains a fluid between two cylinders (circles in 2D) which rotate at different angular rates relative to one another. You are to calculate the stresses acting on the outer ring. In order to effect this model, we will employ a user-defined function for the boundary condition.



Geometry

- R_i = 6 mm; R_o = 8 mm; the shaft rotates with angular frequency ω = 1250 RPM. Create this geometry, ensuring that the circles are concentric with the common center *not* at the origin of the coordinate system. Record the value of the offset distances x_0, y_0.
- In *Design Modeler*, select "mm" as the base units.
- Activate *Auto Constraints* to co-center the circles more easily.
- Use the radius dimension tool to supply the proper dimensions, and the length/distance dimension tool to specify x_0, y_0.

Mesh

- *Mapped Face Mesh*→*Max Face Size* "0.2 mm" [*Meshing* expects length in meters.]
- *Create Named Selections*→"inner", "outer"

Setup

- Double-precision (as usual)
- Check the mesh.
- Accept default settings in *General* and *Models*.

Materials

- *Create/Edit*; create a material called "grease" with the following properties:
  - ρ = 880 kg/m^3; μ = 2 kg/m/s
- *Change/Create*; overwrite "air" if prompted.

Boundary Conditions

- Ensure that "outer" represents a stationary wall.
- Examine the options for "inner". Change the type to a moving wall with rotational components. (We can specify this directly, but will use a UDF for example's sake.)
- Open a text editor and examine the file `inner.c` (download this from the course web site). Commentary is available on the last page of this lab.
- In the main menu on the top (Fluent), execute *Define*→*User-Defined*→*Functions*→*Interpreted*, and browse for your `inner.c` file.
- Click the *Interpret* button, and look at the compiler output for any errors in your code. If everything works properly, when you edit the inner wall boundary conditions to define a moving wall by *Components*, you will have options in the pulldown boxes for `inner_vx` and `inner_vy`.

Initialization, Run Calculation

- 200 iterations

Postprocessing

- Create a vector plot colored by velocity magnitude and do a common-sense check: do the velocities look right for the described conditions of this problem? If your vectors are not all tangential to the inner radius, then there is a good chance that you have not correctly shifted your coordinates in the UDF.
- The Taylor–Couette flow problem has an analytical solution for the tangential velocity profile:

$$u_\theta(r) = -u_i \frac{R_i}{R_o^2 - R_i^2} r + u_i \frac{R_i R_o^2}{R_o^2 - R_i^2} \frac{1}{r}$$

  where r is the radial coordinate, and u_i is the velocity of the inner bearing cylinder. Create an x–y plot of the tangential velocity across the grease film and compare with the analytical solution. Is this a good simulation?

The User-Defined Function `inner.c`

The Fluent UDF language is C, but a highly macro-ized version in order to better support the Fluent API. In order to define the x and y velocity components of the inner bearing race, we will use the macro `DEFINE_PROFILE`.

The first line of your file must be `#include "udf.h"` to get access to the Fluent programming interface. (The basics of the `DEFINE_PROFILE` macro are defined as follows (from the manual):

`DEFINE_PROFILE(name,t,i)`

Argument Type	**Description**
`symbol name`	UDF name
`Thread *t`	Pointer to thread on which boundary condition is to be applied.
`int i`	Index identifying the variable that is to be defined. i is set when you hook the UDF with a variable in a boundary conditions dialog box through the graphical user interface. This index is subsequently passed to your UDF by the ANSYS FLUENT solver so that your function knows which variable to operate on.

Return Type
`void`

Thus there are three required arguments to `DEFINE_PROFILE`: `name`, `t`, and `i`. We are responsible to supply `name`; `t` and `i` are handled by Fluent.

In order to define our inner race velocity, the UDF will look like this:

```
DEFINE_PROFILE(inner_vx, thread, index)
{
real x[ND_ND]; // position vector, from FLUENT
real x0, y0; // center of circle coordinates
real R, omega; // inner radius and angular velocity
face_t f; // fluent face object
x0 = {the x-coordinate of the center of your circles}
y0 = {the y-coordinate of the center of your circles}
R = {your inner radius, in meters}
omega = {your angular velocity, in rpm}
begin_f_loop(f, thread)
{
F_CENTROID(x, f, thread); // this fills the position vector
F_PROFILE(f, thread, index) = {a function we will define}
}
end_f_loop(f, thread)
}
```

Some of this we ignore for the time being, like `face_t f`. (Note that the variable type is `real`; this becomes either `float`s or `double`s depending on whether you run Fluent in single or double precision.) `begin_f_loop` and `end_f_loop` define a loop over cell faces within the specified geometry for the boundary conditions; Fluent tracks the specific cells in the mesh are assigned this boundary condition.

The call to `F_CENTROID` populates the x vector with the coordinates of the current face centroid, so after the function call the value of `x[0]` is the x coordinate and `x[1]` is the y coordinate. The right side of the call to `F_PROFILE` defines the x component of the velocity, which will be a user-specified function of coordinates. This macro will compute the x velocity of the inner race. You will have to define a very similar second macro for the y velocity. (The name of that macro will change to `inner_vy` and the function for `F_PROFILE` will also change accordingly.) The velocity components can be computed as:

$$v_x = -R\omega \sin\theta$$
$$v_y = R\omega \cos\theta$$

So we need to calculate theta. From the geometry, we can compute:

$$\theta = \operatorname{atan2}(x[1] - y_0, x[0] - x_0)$$

where we use the C standard library call `atan2(y, x)`; the coordinates are shifted to account for the center of rotation not being at the origin of the simulation domain. In addition, we need to convert the angular velocity from RPM to rad/s: $\omega = \omega * (2 * \pi / 60)$

In the `inner_vx` function, the x velocity is:

```
F_PROFILE(f, thread, index) = -R * omega * sin(theta)
```

And in the `inner_vy` function, the y velocity is:

```
F_PROFILE(f, thread, index) = R * omega * cos(theta)
```