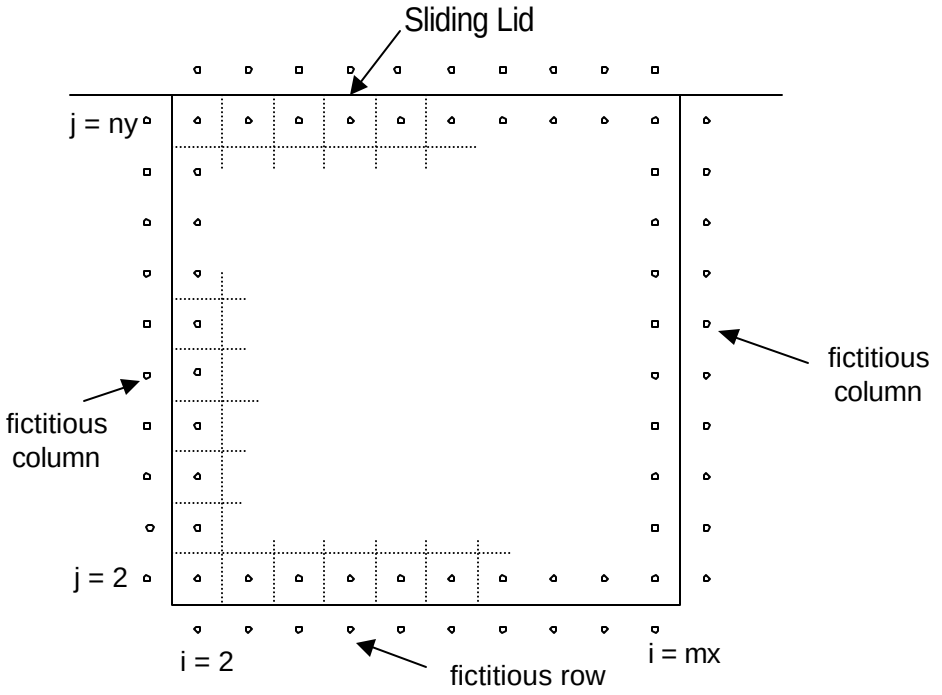
Stokes Flow in a Driven Cavity Using Primitive Variables

Introduction

In this project we will compute the flow of a viscous fluid in a rectangular cavity - the so-called driven cavity problem - using a primitive variable algorithm. By primitive variables we mean pressure and the velocity components, i.e., quantities that appear in the usual forms of the governing equations and may be directly measured in the laboratory. The source of the motion in this problem is a moving upper lid that drags the fluid along (see figure below). The motion will be assumed to be slow enough (as measured by the Reynolds number) that the non-linear (advection) terms may be neglected. Alternatively this problem may be solved in terms of the vorticity and streamfunction equations, and the steady-state form of the latter may also be combined to yield a single biharmonic equation (see Sec. 3.8 of Chow, 1979). The biharmonic equation may be solved using the same iterative methods as commonly used for Poisson equations; however, because the biharmonic equation is fourth order, a nine-point stencil is typically used. Convergence using iterative methods may be very slow. The primitive variable method used here is typical of algorithms for low speed, incompressible flows and is basically that developed by Harlow and Welch in the mid 1960's. Many commercial CFD packages employ schemes that are derived from the method implemented here.

Background

The computational region is shown below. All boundaries of the rectangular region are considered to be "no-slip", and the sliding upper lid drags the viscous fluid along with it, while the other three walls retard the motion.



In this project we retain the original time-dependent, two-dimensional governing equations:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\nabla^2 u, \tag{1}$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\nabla^2 v, \tag{2}$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{3}$$

Note that the equation of state for an incompressible fluid reduces to simply ρ = constant. Velocities, time, pressure and lengths are non-dimensionalized as $u'=\frac{u}{U_o}$; $t'=\frac{U_o t}{L}$; $p'=\frac{p}{\rho U_o^2}$; and $x'=\frac{x}{L}$, respectively. Then after dropping the primes, Equation 1, for example, becomes:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{\partial p}{\partial x}+\frac{1}{\mathrm{Re}}\left[\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right] \tag{4}$$

Neglecting the non-linear terms, the equations we want to solve are:

$$\frac{\partial u}{\partial t}=-\frac{\partial p}{\partial x}+\frac{1}{\mathrm{Re}}\nabla^2 u, \tag{5}$$

$$\frac{\partial v}{\partial t}=-\frac{\partial p}{\partial y}+\frac{1}{\mathrm{Re}}\nabla^2 v, \tag{6}$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{7}$$

These last three equations have three unknowns: u, v and p. Since they include derivatives that might be used to project out in time, it appears that Equation 5 and 6 may be used as predictive equations for u and v, respectively. Unfortunately, there is no comparable equation for p and furthermore the equation of state, which for a compressible fluid relates the pressure and density, is of little value here. To compensate we propose to use the continuity equation (7) instead. The placement of variables used in the following discussion is shown in Figure 2 below.

Essentially the pressure is defined at the center of primary control volumes and the velocities are defined at the edges, centered in their own control volumes. It would be more descriptive to name the velocities appearing in the sketch as $u_{i+\frac{1}{2},j}$ and $v_{i,j+\frac{1}{2}}$, but computers don't like fractional indices. This grid arrangement is known as a staggered mesh.

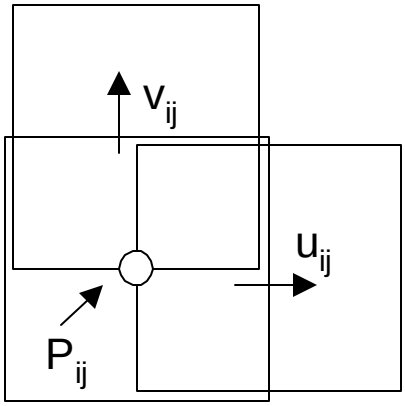


Figure 2. Staggered Mesh

With the prime indicating the advanced time level from here on, we require continuity at that time level, i.e.,

$$u'_{i-1,j}\Delta y - u'_{i,j}\Delta y + v'_{i,j-1}\Delta x - v'_{i,j}\Delta x = 0 \tag{8}$$

(Divide through by $\Delta x \Delta y$ and Equation 8 is a finite-difference approximation to Equation 7.) The staggered mesh has an obvious advantage here; the velocities are defined exactly at the faces of the primary volumes where they are needed and no averaging is used. Furthermore the volumetric flux leaving one cell is identical to the volumetric flux entering its next-door neighbor. Patankar (1980) discusses the advantages of this staggered arrangement in terms of the absence of "waviness" in the computed pressure and velocity fields.

Recognizing that their own control volumes are shifted a half grid space to the right of the primary control volumes and a half grid space up, respectively, the momentum equations (5,6) are differenced as:

$$\frac{u'_{i,j} - u_{i,j}}{\Delta t} = -\frac{P'_{i+1,j} - P'_{i,j}}{\Delta x} + RM_{i,j} \tag{9}$$

$$\frac{v'_{i,j} - v_{i,j}}{\Delta t} = -\frac{P'_{i,j+1} - P'_{i,j}}{\Delta y} + QM_{i,j} \tag{10}$$

Here the RM and QM are finite-difference approximations to the viscous terms (plus the convection terms if present) and are located coincidentally with the u and v velocities, respectively. Now divide the continuity equation at the new time level (Equation 8) by Δt and stick these and analogous forms at (i-1,j) and (i,j-1) in:

$$\Delta y\left[\frac{u_{i-1,j}}{\Delta t} - \frac{P'_{i,j} - P'_{i-1,j}}{\Delta x} + RM_{i-1,j}\right] - \Delta y\left[\frac{u_{i,j}}{\Delta t} - \frac{P'_{i+1,j} - P'_{i,j}}{\Delta x} + RM_{i,j}\right] \tag{11}$$

$$\Delta x\left[\frac{v_{i-1,j}}{\Delta t} - \frac{P'_{i,j} - P'_{i,j-1}}{\Delta y} + QM_{ij-1}\right] - \Delta x\left[\frac{v_{i,j}}{\Delta t} - \frac{P'_{i,j+1} - P'_{i,j}}{\Delta y} + QM_{i,j}\right]$$

Dividing through Equation 11 by $\Delta x \Delta y$ and using d_x^2 and d_y^2 to indicate the second central difference operators, we get:

$$d_x^2 P' + d_y^2 P' = \frac{D}{\Delta t} + S_p \tag{12}$$

a discretized Poisson equation for pressure at the advanced time level. Here

$$D \equiv \frac{u_{i,j} - u_{i-1,j}}{\Delta x} + \frac{v_{i,j} - v_{i,j-1}}{\Delta y} \tag{13}$$

which represents the divergence at the current time level. This term ought to be zero, but may not be due to incomplete convergence. It must be retained (Harlow and Welch, 1965)! The source term S_p represents other terms including diffusion, convection where applicable, body forces, etc.

Implementation

With the general background discussed above we still have a number of matters to consider in the implementation. These include:

1. What sequence of operations to use?

2. What method to use on the pressure Poisson equation (and which form to use, Equation 11 or 12)?

3. What boundary conditions for the pressure Poisson equation? They seem to be all Neumann?!

4. Since the diffusion terms are treated explicitly, is there a time step limitation and how is it computed?

5. Is the algorithm stable overall?

6. What boundary conditions are appropriate for the u and v equations?

While the pressure equation can be put into the neat form of Equation 12, it is much easier to apply the boundary conditions on pressure using Equation 11. For instance, for all cells along the left hand boundary, there is no flux through the left hand face. For cells along the lower boundary, there is no flow through their lower face. Thus the whole first bracketed term must be eliminated for cells along the left boundary; the whole third bracketed term must be eliminated for cells on the bottom boundary, etc. This way the normal five-point stencil for pressure automatically takes a four-point form along the boundaries and a three-point form in the corners. Moreover, with clever programming one set of nested LOOP's can be made to cover both internal and boundary volumes in the pressure iteration. This is much easier than to try to figure out boundary conditions for Equation 12, but the two must be equivalent.

There are several ways in which to implement the above. The most straightforward is to set up four vectors to use as "masks." These masks will take on values of 1.0 or 0.0 depending on the position in the grid. For instance, when i = 2, the western mask will have a value of 0.0, thus eliminating all the terms within the first bracketed term for cells in the first real column (which corresponds to i = 2 in the figure). For all

other values of i within the container, the mask would have a value of 1.0. Another equivalent approach, again using i = 2 as an example, would take advantage of the fact that $u_{i-1,j}, P'_{i-1,j}$ and $RM_{i-1,j}$ are either actually 0.0 or have that value because they were initialized to 0.0 and have not been computed. The only term left to take care of is the $P'_{i,j}$. When gathering up and solving for $P'_{i,j}$, it must be remembered that this one doesn't count, and thus the coefficient changes from the usual 1/4 for internal points to 1/3 along edges and 1/2 in corners.

You will most likely be solving the Poisson pressure equation by iteration. Because the source term in the Poisson equation will be changed at the next timestep, there is no need to fully converge the pressure solution at every timestep (assuming you have retained the divergence term as recommended). Thus you ought to limit the maximum number of sweeps per time step to some small number, say 10, and to include a convergence test so that as overall convergence is approached, the number of Poisson sweeps used at each timestep is decreased. Printing out the number of Poisson sweeps required at each timestep is a good way to monitor overall convergence. The last few timesteps should only require 1 or 2 sweeps, since at each timestep you are using better and better pressure values as your first guess.

Boundary conditions are also needed for the velocity equations. See the diagram of the mesh. Suppose, just for illustration, that the left wall is a rigid, slip boundary. (It is to be taken as no-slip, in fact.) Go back to the derivation of the Poisson equation. No throughflow at the left face of a cell means $u'_{i-1,j} = 0$, i.e., $u'_{1,j} = 0$ here, so as discussed above, terms resulting from this disappear from the Poisson equation at i = 2. If we wanted to create a "slip" boundary along the left side, we would set $v_{1,j} = v_{2,j}$. Here $v_{1,j}$ is a fictitious cell outside the computing region, the sole purpose of which is to facilitate imposition of boundary conditions. These values will go into the RM and QM calculated at cells adjacent to the walls. Now, suppose the right side is rigid, no slip. Then $u_{mx,j} = 0$ so terms resulting from it disappear from the Poisson equation at i = mx and $v_{mx+1,j} = -v_{mx,j}$ to make it "no slip." Along the bottom and top boundaries the v = 0.0 boundary condition may be applied directly. The u velocity, which is defined a half space in from the boundary itself, must be reflected with a sign change to satisfy the no slip boundary condition at the bottom. At the top the fictitious values must be computed so that the mean of the real and fictitious values at each value of i is equal to the lid velocity. Assuming you have dimensioned the u and v arrays both (mx+1,ny+1), you should recognize that u(mx+1,j) and v(i,ny+1) will never be used. It is critically important that you get all the DO LOOP ranges right in this problem! It is strongly suggested that you get a piece of graph paper and make yourself a large map of your grid similar to the first figure!

The suggested sequence is as follows:

1. Set up grid and initialize. Use MX = NY = 11 for debugging and use parameter statements so that this can be changed easily later.

2. Compute allowable timestep. (Hint, it is limited by the explicit differencing of the diffusion term in Equation 9 and 10 and is fixed for the whole calculation. If we had included the convection terms, the allowable timestep would be changing as the velocities build up.)

3. Compute the explicit terms (RM, QM). Here again, the key is getting the DO LOOP ranges correct! If you chose not to use the masks for the pressure Poisson equation as discussed above, it is critical that you only compute RM and QM at points where you are also are computing u and v.

4. Compute the right hand side of the Poisson equation (divergence and source terms).

5. Set up and solve the Poisson equation for pressure. Any iterative method (Gauss-Seidel, SOR, etc.) may be used. Obviously you will use the last values from the previous timestep as the first guess.

6. Update velocities using the new pressure and the RM and QM from step (3).

7. Take care of boundary conditions on u and v. All surfaces are to be "no-slip"; the upper boundary is moving at a uniform velocity of unity. The "reflection" you do to implement the no-slip boundary will need to be altered along the top boundary.

8. Repeat steps 3-7 until convergence. A pass through steps 3-7 constitutes a timestep. At each timestep you should print out several variables including the number of Poisson sweeps required.

9. If you have the capability, plot velocity vectors. You will have to average because the u and v velocities are not defined at the same place. A convenient way to do it is to position your resultant velocity at the same point as pressure. Then you can average the horizontal velocity to either side and the vertical velocity above and below to get the resultant that you will plot. If you don't have velocity vector capability, see Item 1 under "Extras" below.

Extras

1. Integrate either your u or v velocity field to find streamfunction, plot and discuss. Whether you integrate u or v, it is convenient to define the streamfunction at the corners of the primary cells.

2. Test your results against Stokes' theorem; i.e., calculate the circulation for your region and compare with the integral of vorticity over the area.

3. Make a contour plot of pressure and discuss.

4. Compute the vorticity ($z = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$) at all points, make a contour plot and discuss.

Verification

You should compute the divergence for each cell and ensure that it is nearly zero at the end. If you encounter difficulty along the way, you may want to compute the divergence at intermediate steps. This may help you localize a programming error.

A plot of velocity vectors by itself is not that good a means to check the validity of your solution. It will show general flow patterns, ensure that velocities go to zero at the boundaries, etc. A plot of streamfunction as suggested above can be very helpful in checking that mass is conserved everywhere.

References

1. Chow, C.Y., *An Introduction to Computational Fluid Mechanics*, Wiley, New York (1979).

2. Fletcher, C.A.J., *Computational Techniques for Fluid Dynamics*, Vol II, Chap. 17, Springer-Verlag, Berlin (1991).

3. Harlow and Welch, *Physics of Fluids*, 8, 2182-2189, 1965.

4. Patankar, S.V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere (1980).

StokesPrim.doc
4/9/02