Stream Function-Vorticity CFD Solver
MAE 6263

Charles O'Neill

April 1, 2002

Abstract

A finite difference CFD solver was developed for transient, two-dimensional Cartesian viscous flows. Flow parameters are solved through iterating stream function and time marching vorticity. The governing equations and discrete representations for Cartesian, two dimensional flow are reviewed. Two internal flows through an expansion box are solved and discussed. The stream function-vorticity CFD program works correctly. Conclusions regarding the solution technique and general CFD are made.

1 Introduction

The objective is to solve 2D Cartesian transient viscous flows. The governing equations for continuity and momentum are reviewed[1] and finite difference forms are developed. Next, the developed FORTRAN CFD program is discussed. Then, the geometry and fluid properties for the internal flow problem are shown. Results are given and discussed. Finally, conclusions are made.

2 Governing Equations

The fluid flow is solved for a transient, constant-density Cartesian coordinate system based on the stream function and vorticity approach. Continuity and momentum are inherently imbedded in the stream function and vorticity. Finally, the flow relationships are transformed to discrete representations for input into a computer routine.

2.1 Continuity

Continuity for a 2D Cartesian coordinate system is given as,

$$\frac{du}{dx} + \frac{dv}{dy} = 0$$

The stream function definitions $u = d\Psi/dy$ and $v = -d\Psi/dx$ can be imbedded into the continuity equation with vorticity,$\omega = \frac{dv}{dx} - \frac{du}{dy}$,to form a simple Ψ vs ω expression.

$$\nabla\Psi = -\omega$$

This expression must be made discrete for use in the computer program. Applying the appropriate 2nd order finite differentials and solving for Ψ at the center point,P, yields,

$$\Psi_p = A_E\Psi_E + A_W\Psi_W + A_N\Psi_N + A_S\Psi_S + A_\omega\Psi_\omega$$

where the influence coefficients are given by

$$A_{E,W} = \frac{1}{(\Delta x)^2} \cdot \left(\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2}\right)^{-1}$$

$$A_{N,S} = \frac{1}{(\Delta y)^2} \cdot \left(\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2}\right)^{-1}$$

$$A_\omega = \left(\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2}\right)^{-1}$$

This discrete form of the continuity equation is ready for Gauss-Seidel iteration in the computer program.

2.2 Momentum

Similarly, the expression for vorticity is substituted into the governing momentum equations. Thus, the momentum relationship in terms of ω and velocity is,

$$\frac{d\omega}{dt} + \frac{d(u\omega)}{dx} + \frac{d(u\omega)}{dy} = \nu\nabla^2\omega$$

As before, this expression is discretized by the appropriate finite differentials. The discrete expression must account for the convection terms and perform the appropriate forward time-march in vorticity. Generally, the discrete governing equation (solved at the center point P) is,

$$\omega_P = \omega_{P_{old}} + \Delta T\left(-\mathrm{Adv}_x - \mathrm{Adv}_y + \mathrm{Visc}\right)$$

where the terms $Adv_{x,y}$ are the appropriate upstream differenced expressions and $Visc$ expresses the viscous terms. This expression for ω is time marched forward.

$$Adv_x = \begin{array}{ll} ((u\omega)_E - (u\omega)_W)/(2\Delta x) & u_P = 0 \\ ((u\omega)_P - (u\omega)_W)/(\Delta x) & u_P > 0 \\ ((u\omega)_E - (u\omega)_P)/(\Delta x) & u_P < 0 \end{array}$$

$$Adv_y = \begin{array}{ll} ((u\omega)_N - (u\omega)_S)/(2\Delta y) & v_P = 0 \\ ((u\omega)_P - (u\omega)_S)/(\Delta y) & v_P > 0 \\ ((u\omega)_N - (u\omega)_P)/(\Delta y) & v_P < 0 \end{array}$$

$$Visc = \begin{array}{l} (\omega_N - 2\omega_P + \omega_S)\cdot\nu/(\Delta y)^2 \\ +(\omega_E - 2\omega_P + \omega_W)\cdot\nu/(\Delta x)^2 \end{array}$$

2.3 Boundary Conditions

There are two types of boundary conditions needed. First, symmetry and adiabatic boundary conditions are needed for symmetrical and outlet edges. Second, boundary conditions on the wall are needed.

Symmetry boundary condition are created by creating fictious points just outside the flow domain. These fictious points mirror the fluid properties just inside the flow domain. Thus at a symmetry boundary, the relationship between the neighboring properties normal to the symmetry plane are

$$\zeta_{i+1} = \zeta_{i-1}\mathrm{atpoint}i$$

. This is equivalent to the 2 point second derivative at i being equal to zero.

The second boundary condition concerns vorticity at the walls. At a solid boundary, the update expression for ω is,

$$\omega_p = 2(\Psi_p - \Psi_i)/\delta^2$$

where δ and i are in the direction (inward) normal to the wall.

3 Computer Code

A FORTRAN computer code was written to solve the governing equations developed above. The complete code is given in the Appendix. The overall solution strategy combines iterating stream function and time-marching vorticity. The general steps of the program are given below.

1. **Generate Mesh:** The program begins by creating the finite difference mesh. The matrices for $\Psi, \omega, \omega_{old}, u$ and v are initialized and filled with zeros. Next, the grid intervals and sizes in the x and y direction are created. Finally, the Ψ influence coefficients are calculated.

2. **Problem Properties:** The viscosity, inlet velocity and solution time properties are set. The output files for stream function and vorticity are created.

3. **Initial Conditions:** The velocity and stream function initial conditions are calculated for the four walls. The program determines the inlet stream function values by integrating the inlet velocity.

4. **Advance Time:** This step begins the main time-marching routine. Time is advanced through a DO loop.

5. **Internal ω:** An ω sweep is made over the internal nodes. The appropriate influence coefficients for the convection and viscosity are calculated. Next, the vorticity values to the west of the east wall are copied to create an east symmetry plane. Finally, the time-marched internal values of vorticity are calculated according to the influence coefficients and the old vorticity values.

6. **Iterate Ψ:** The stream function values are iterated to conform to the new values of vorticity. This step enforces continuity. A residual checking routine stops the iterations if the stream function field is sufficiently converged.

7. **Find Velocities:** The u and v velocities are found from the local values of stream function. Corner values of velocity are not needed. However, these corner velocities are set to match neighboring points so the final solution graphs are smoother.

8. **Vorticity Boundary Values:** Boundary values of vorticity are calculated from wall and just-internal stream function values. Nonphysical vorticity values appear near sharp convex corners when two velocities are not equal. Roache and Mueller[2] discuss this problem.

9. **Repeat Time March:** The time march is repeated as specified in the *Advance Time* DO loop.

10. **Output Results:** Intermediate and final values of Ψ, ω and velocities are output. A special final WRITE statement is needed to properly view the results in MATLAB.

4 Problem Geometry

The problem consists of a two dimensional backstep internal flow. The flow transitions from a 0.25 meter duct to a 0.50 meter duct. A symmetry plane exists at the centerline of the duct, thus computations are performed only on one half. The geometrical mesh as input into the CFD solver is shown in Figure 1. The flow domain consists of a 0.25 meter high and

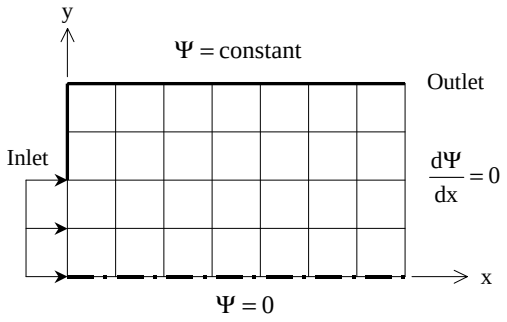

Figure 1: Solution Grid

0.35 meter length rectangle. Fluid enters on the left. The outlet boundary condition, $d\Psi/dx = 0$, approximates no-property-changes in outgoing fluid.

5 Results and Discussion

Two testcases are presented and discussed: a steady state and a transient case. Both were solved with the computer program and solution technique given above.

5.1 Steady State

A steady state flow solution was attempted based on the above flow geometry. The domain was divided into a 5 by 7 grid. The Reynolds number (based on step height) was 125000. Figure 2 plots the vorticity residuals versus time. The residuals converge at approximately 1.4 seconds. The problem's velocity vectors and Ψ contours for the converged steady state solution are shown in Figure 3. Vorticity contours for the problem are given in Figure 4. The vorticity contours directly off the north wall are due to a forming boundary layer. Similarly, the vorticity behind the step (west wall) corresponds to flow expanding around the corner.

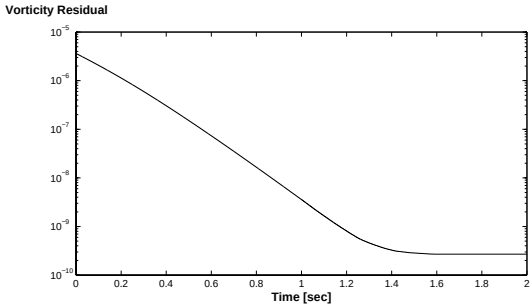

Figure 2: Vorticity Residuals vs. Time

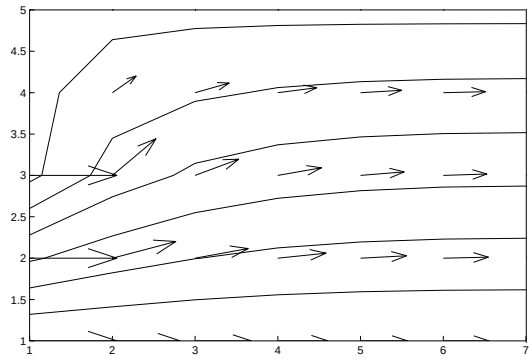

Figure 3: Velocities and Ψ Contours at Steady State

From intuition, the flow solver appears to solve the governing equations correctly. While the flow domain is rather coarse, the solver still managed to

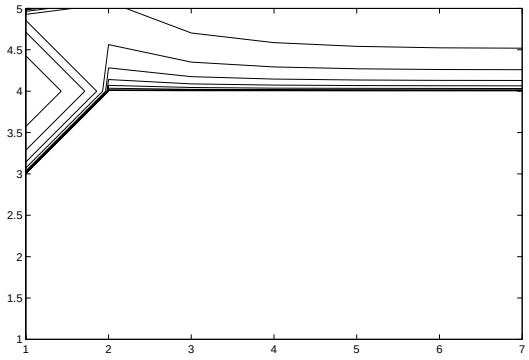

Figure 4: Vorticity Contours at Steady State

converge to a steady state solution. Decreasing the grid sizes would improve resolution at the expense of solution time. From Roache and Mueller[2], it is expected that the time step for stability approximately decreases with the square of grid size. Better solutions will take vastly longer.

5.2 Transient

A transient case was next attempted. While the steady case was a converged transient case, a new testcase was needed to test convection of a completely swirling flow where vorticity would clearly convect downstream.

This "transient" testcase required an increase in grid resolution to 10 by 14. The fluid properties were changed to a Reynolds number based on step size of 12.5. Figure 5 shows the vorticity residuals. The flow solution converges; however, the solution is clearly more complex because the residuals are jagged. The transient velocities and vorticity time

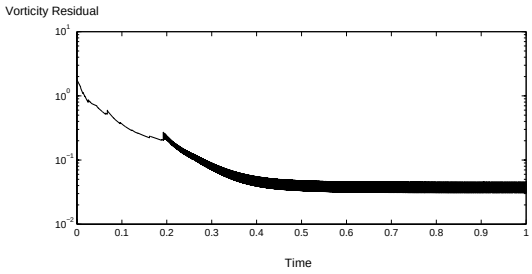

Figure 5: Transient Vorticity Residuals

slices are shown in figure 6. The initial flow at t=0 quickly forms a vortex behind the step. The vorticity visibly convects downstream. Figure 7 shows the steady state velocities.

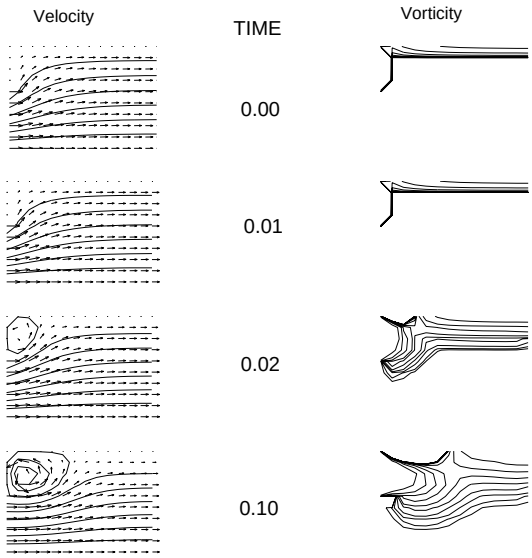

Figure 6: Transient Velocity and Vorticity

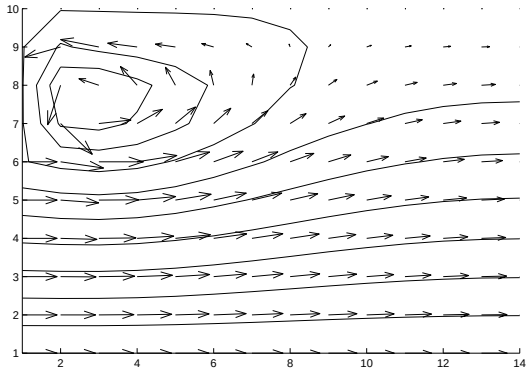

Figure 7: Converged Velocity and Vorticity

This testcase demonstrated the transient abilities of the flow solver. The flow solution appears reasonable and follows the expected flow physics.

6 Conclusions

A transient 2D stream function-vorticity flow solver was successfully implemented. A FORTRAN code is used to iteratively solve for stream function and

vorticity. The flow solver appears to solve transient 2D viscous flows correctly.

The stream function combines both flow velocities which simplifies the overall flow solution and creates an intuitive output function. However, the method requires coupling stream function and vorticity through velocities. Thus, the method doesn't eliminate finding velocities.

The current implementation does have problems. As described in Roache and Mueller[2], corners and edges are often difficult to properly define when working with discrete expressions. Also, there are often too many nodes in low gradient areas. These "extra" nodes waste computer effort.

Overall, the stream function-vorticity solution technique works and is moderately simple to implement. Two testcases based on a backstep expansion were analyzed with the developed computer code.The code was used to determine an "easy" steady state problem and a more difficult "swirling" transient problem

References

[1] Lilley, D. G., *Computational Fluid Dynamics. Vol I-III*, Stillwater, OK, 1992.

[2] Roache, P. J. and Mueller, T. J, "Numerical Solutions of Laminar Separated Flows," *AIAA Journal*, Vol. 8, March 1970, pp. 530-537.