Numerical Implementation for 2D Lid-Driven Cavity Flow in Stream Function Formulation

1 Problem Description

Fluid is contained in a square domain with Dirichlet boundary conditions on all sides, with three stationary sides and one moving side (with velocity tangent to the side).

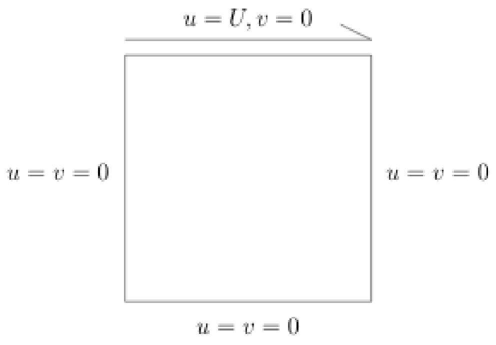

Figure 1 Lid Cavity Flow

Solve the lid driven cavity flow using vorticity-stream function formulation. Apply the finite difference approach for discretization. Plot the velocity vectors, the vorticity contours and the stream function. Explain your methodology in detail and comment on the results.

2 Theory

2.1 Stream Function

The two dimensional incompressible flow continuity equation is given by

$$\nabla . \mathrm{V} = 0$$
$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{1}$$

This equation is satisfied identically by the stream function $\psi(x, y)$ which is defined as

$$\mathrm{V} = \left(\frac{\partial \psi}{\partial y}\right)\hat{\imath} + \left(-\frac{\partial \psi}{\partial x}\right)\hat{\jmath} \tag{2}$$

Such that Equation (1) is becomes

$$\frac{\partial}{\partial x}\left(\frac{\partial \psi}{\partial y}\right) + \frac{\partial v}{\partial y}\left(-\frac{\partial \psi}{\partial x}\right) = 0 \tag{3}$$

Two variables u and v are replaced by a single higher order function ψ. The vorticity ω of the velocity vector u is defined as the curl of V,

$$\omega = \nabla\times V = \begin{vmatrix} i & j & k \\ \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & 0 \\ \dfrac{\partial\psi}{\partial y} & -\dfrac{\partial\psi}{\partial x} & 0 \end{vmatrix} = -\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}\right)\hat{k} = -\nabla^2\psi\hat{k} \qquad (4)$$

The incompressible flow momentum equation is given by

$$\rho\frac{DV}{Dt} = -\vec{\nabla}P + \rho\vec{g} + \mu\nabla^2 V \qquad (5)$$

The body force $\rho\vec{g}$ can be written as the gradient of a scalar potential. As all gradient fields are irrotational, taking the curl of the Equation (6) makes the gradient fields of pressure and body force potential disappear. We obtain

$$\left(\frac{\partial}{\partial t}+u\frac{\partial}{\partial x}+v\frac{\partial}{\partial y}\right)(\nabla\times V) = \nu\nabla^2(\nabla\times V) \qquad (6)$$

where $\nu = \mu/\rho$ is the kinematic viscosity. Utilizing the definitions of Equations (2) and (4)

$$\frac{\partial\omega}{\partial t}+\frac{\partial\psi}{\partial y}\frac{\partial\omega}{\partial x}-\frac{\partial\psi}{\partial x}\frac{\partial\omega}{\partial y} = \nu\left(\frac{\partial^2\omega}{\partial x^2}+\frac{\partial^2\omega}{\partial y^2}\right) \qquad (7)$$

Equations (4) and (7) are the governing equations of the flow. ω and ψ fields are to be solved for.

2.2 Boundary Conditions

The normal velocity is zero at walls, thus

$$V_n = (-)\frac{\partial\psi}{\partial t} = 0 \qquad (8)$$

where n and t are the normal and tangential to wall directions, respectively. As the derivate of ψ along any wall is zero, ψ is constant along any wall. Thus

$$\psi_{wall} = c \qquad (9)$$

where c is an arbitrary constant, which may be set to zero. Moreover, Equation (4) is reduced to

$$\omega_{wall} = -\frac{\partial^2\psi}{\partial n^2} \qquad (10)$$

The tangential boundary velocities are given by

$$\begin{aligned} u_{top} &= \left.\frac{\partial\psi}{\partial y}\right|_{top} = U_{wall} \\ u_{bottom} &= \left.\frac{\partial\psi}{\partial y}\right|_{top} = 0 \\ v_{side} &= -\left.\frac{\partial\psi}{\partial x}\right|_{side} = 0 \end{aligned} \qquad (11)$$

2.3 Finite Difference Approximation

The finite difference method is a numerical procedure which solves a partial differential equation (PDE) by discretizing the continuous physical domain into a discrete finite difference grid, approximating the individual exact partial derivatives in the PDE by algebraic finite difference approximations (FDAs), substituting the FDAs in to the PDE obtain an algebraic finite difference equation (FDE), and solving the resulting algebraic finite difference equations (FDEs) for the dependent variable. The solution domain must be covered by a two-dimensional grid of lines, called the finite difference grid. The intersections of these grid lines are the grid points at which the finite difference solution to the partial differential equation is to be obtained. The subscripts i and j are used to denote the physical grid lines corresponding to constant values of x and y, respectively, such that

$$\begin{aligned} x_i &= (i-1)\Delta x \\ y_j &= (j-1)\Delta y \end{aligned} \tag{12}$$

Thus, grid point (i,j) corresponds to location (x_i, y_j) in the solution domain. The total number of x grid lines is denoted by imax, and the total number of y grid lines is denoted by jmax. For a grid aspect ratio of 1, $\Delta y = \Delta x = h$. The superscript n is used to denote the time step.

Finite difference approximations (FDAs) of the individual exact partial derivatives appearing in the partial differential equation must be developed. This is accomplished by writing Taylor series for the dependent variable at several neighboring grid points using grid point (i,j) as the base point, and combining these Taylor series to solve for the desired partial derivatives.

The exact spatial and temporal partial derivatives in Equation (7) are approximated by the 2nd and 1st order centered difference approximations, respectively, to yield:

$$\begin{aligned} \frac{\omega_{i,j}^{n+1} - \omega_{i,j}^{n}}{\Delta t} + \left(\frac{\psi_{i,j+1}^{n} - \psi_{i,j-1}^{n}}{2h}\right)\left(\frac{\omega_{i+1,j}^{n} - \omega_{i-1,j}^{n}}{2h}\right) \\ - \left(\frac{\psi_{i+1,j}^{n} - \psi_{i-1,j}^{n}}{2h}\right)\left(\frac{\omega_{i,j+1}^{n} - \omega_{i,j-1}^{n}}{2h}\right) \\ = \nu\left(\frac{\omega_{i+1,j}^{n} - 2\omega_{i,j}^{n} + \omega_{i-1,j}^{n}}{h^2} + \frac{\omega_{i,j+1}^{n} - 2\omega_{i,j}^{n} + \omega_{i,j-1}^{n}}{h^2}\right) \end{aligned} \tag{13}$$

Rearrange to obtain an explicit formula for $\omega_{i,j}^{n+1}$

$$\begin{aligned} \omega_{i,j}^{n+1} = \omega_{i,j}^{n} + \Delta t\Bigg[\left(\frac{\psi_{i+1,j}^{n} - \psi_{i-1,j}^{n}}{2h}\right)\left(\frac{\omega_{i,j+1}^{n} - \omega_{i,j-1}^{n}}{2h}\right) \\ - \left(\frac{\psi_{i,j+1}^{n} - \psi_{i,j-1}^{n}}{2h}\right)\left(\frac{\omega_{i+1,j}^{n} - \omega_{i-1,j}^{n}}{2h}\right) \\ + \nu\left(\frac{\omega_{i+1,j}^{n} + \omega_{i-1,j}^{n} + \omega_{i,j+1}^{n} + \omega_{i,j-1}^{n} - 4\omega_{i,j}^{n}}{h^2}\right)\Bigg] \end{aligned} \tag{14}$$

Similarly, Equation (4) yields

$$\omega_{i,j} = \frac{4\psi_{i,j} - \psi_{i+1,j} - \psi_{i-1,j} - \psi_{i,j+1} - \psi_{i,j-1}}{h^2} \tag{15}$$

Rearrange to obtain an expression for $\psi_{i,j}$

$$\psi_{i,j} = \frac{1}{4}\left(\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} + h^2\omega_{i,j}\right) \tag{16}$$

The boundary conditions along the top wall nodes are

$$\begin{gathered} u = \frac{\partial \psi_{i,jmax}}{\partial y} = U_{wall} \\ \omega_{i,jmax} = -\frac{\partial^2 \psi_{i,jmax}}{\partial y^2} \end{gathered} \tag{17}$$

To obtain the top wall boundary condition FDA, consider the stream function backward Taylor series expansion

$$\psi_{i,jmax-1} = \psi_{i,jmax} - \frac{\partial \psi_{i,jmax}}{\partial y}\, h + \frac{\partial^2 \psi_{i,jmax}}{\partial y^2}\frac{h^2}{2} + \cdots \tag{18}$$

Substitute Equation (17) into (18) to obtain

$$\psi_{i,jmax-1} = \psi_{i,jmax} - hU_{wall} - \frac{h^2}{2}\omega_{i,jmax} + \cdots \tag{19}$$

Ignore higher order terms and rearrange to obtain

$$\omega_{i,jmax} = \frac{2}{h^2}\left(\psi_{i,jmax} - \psi_{i,jmax-1}\right) - \frac{2}{h}U_{wall} \tag{20}$$

Similarly, the bottom and side wall boundary conditions FDA are given by

$$\begin{gathered} \omega_{i,1} = \frac{2}{h^2}\left(\psi_{i,1} - \psi_{i,2}\right) \\ \omega_{1,j} = \frac{2}{h^2}\left(\psi_{1,j} - \psi_{2,j}\right) \\ \omega_{imax,j} = \frac{2}{h^2}\left(\psi_{imax,j} - \psi_{imax-1,j}\right) \end{gathered} \tag{21}$$

3 Algorithm

An algorithm for computing the temporal evolution of incompressible, two-dimensional flow using vorticity-stream function formulation is given as follows:

1) Initiate the stream function
2) Compute the boundary nodes vorticity using Equations (20) and (21)
3) Compute the interior nodes vorticity using Equation (15)
4) Compute the interior nodes new time step vorticity using Equation (14)
5) Use an iterative scheme for elliptic equations to solve Equation (16) for the interior nodes new time step stream function
6) Loop back to (2)

4 Iterative Methods

The application of Equation (16) on all grid points results in a system of a general form

$$A\psi = \omega \tag{22}$$

Direct solutions of Equation (22) require excessive computational effort and computer memory as the number of grid points increases. In such cases, iterative methods should be employed.

4.1 Jacobi Method

An initial approximation $\psi^{(0)}$ is made to start the process. The superscript in parentheses denotes the iteration number, with zero denoting the initial solution vector. The initial solution vector $\psi^{(0)}$ is substituted into Equation (16) to yield the first improved solution vector $\psi^{(1)}$.This procedure is repeated (i.e., iterated) until some convergence criterion is satisfied. The Jacobi algorithm for the general iteration step (k) is:

$$\psi_{i,j}^{(k+1)} = \frac{1}{4}\left(\psi_{i+1,j}^{(k)} + \psi_{i-1,j}^{(k)} + \psi_{i,j+1}^{(k)} + \psi_{i,j-1}^{(k)} + h^2\omega_{i,j}\right) \tag{23}$$

In the Jacobi method, all values of $\psi_{i,j}^{(k+1)}$ are based on $\psi_{i,j}^{(k)}$.

4.2 Gauss-Seidel Method

The Gauss-Seidel method is similar to the Jacobi method except that the most recently computed values of all $\psi_{i,j}^{(k)}$ are used in all computations. In brief, as better values of $\psi_{i,j}$ are obtained, use them immediately. The Gauss-Seidel algorithm is obtained from the Jacobi algorithm, Equation (23) by using the new values of terms of smaller indices (assuming the sweeps through the nodes proceed from $i, j = 1$ to end). Thus,

$$\psi_{i,j}^{(k+1)} = \frac{1}{4}\left(\psi_{i+1,j}^{(k)} + \psi_{i-1,j}^{(k+1)} + \psi_{i,j+1}^{(k)} + \psi_{i,j-1}^{(k+1)} + h^2\omega_{i,j}\right) \tag{24}$$

An equivalent, but more convenient, form of Equation (24) can be obtained by adding and subtracting $\psi_{i,j}^{(k)}$ from its right-hand side to yield

$$\psi_{i,j}^{(k+1)} = \psi_{i,j}^{(k)} + \frac{1}{4}\left(\psi_{i+1,j}^{(k)} + \psi_{i-1,j}^{(k+1)} + \psi_{i,j+1}^{(k)} + \psi_{i,j-1}^{(k+1)} + h^2\omega_{i,j}\right) - \psi_{i,j}^{(k)} \tag{25}$$

Equation (25) is generally written in the form

$$\psi_{i,j}^{(k+1)} = \psi_{i,j}^{(k)} + R_{i,j}^{(k)} \tag{26}$$

$$R_{i}^{(k)} = \frac{1}{4}\left(\psi_{i+1,j}^{(k)} + \psi_{i-1,j}^{(k+1)} + \psi_{i,j+1}^{(k)} + \psi_{i,j-1}^{(k+1)} + h^2\omega_{i,j}\right) - \psi_{i,j}^{(k)} \tag{27}$$

where the term $R_{i,j}^{(k)}$ is called the residual.

Iterative methods do not yield the exact solution of the system equation directly. They approach the exact solution asymptotically as the number of iterations increases. When the number of iterations increases without bound, the iterative solution yields the exact solution of the system equation, within the round-off limit of the computer. However, in most practical problems, such extreme precision is not warranted. Consequently, the iterative process is usually terminated when some form of convergence criterion has been achieved.

$$\left[\sum_{i,j}^{end}\left(\frac{\psi_{i,j}^{(k+1)}-\psi_{i,j}^{(k)}}{\psi_{i,j}^{(k)}}\right)^2\right]^{1/2}=\left[\sum_{i,j}^{end}\left(\frac{R_{i,j}^{(k)}}{\psi_{i,j}^{(k)}}\right)^2\right]^{1/2}<\varepsilon \tag{28}$$

where ε is the convergence tolerance.

4.3 Successive Over-Relaxation Method

Iterative methods are frequently referred to as relaxation methods, since the iterative procedure can be viewed as relaxing $\psi^{(0)}$ to the exact value ψ. Southwell observed that in many cases the changes in ψ_i from iteration to iteration were always in the same directions. Consequently, over-correcting (i.e., over-relaxing) the values of ψ_i by the right amount accelerates convergence. The Gauss-Seidel method, Equation (24), becomes over-relaxation simply by over-relaxing the residual, by the over-relaxation factor β. Thus,

$$\begin{aligned}\psi_{i,j}^{(k+1)} &= \psi_{i,j}^{(k)}+\beta R_{i,j}^{(k)}\\ &=(1-\beta)\psi_{i,j}^{(k)}+\frac{\beta}{4}\left(\psi_{i+1,j}^{(k)}+\psi_{i-1,j}^{(k+1)}+\psi_{i,j+1}^{(k)}+\psi_{i,j-1}^{(k+1)}+h^2\omega_{i,j}\right)\end{aligned} \tag{29}$$

When Equation (29) is applied repetitively, it is called the successive-over-relaxation (SOR) method.

When $\beta = 1.0$, the SOR method reduces to the Gauss-Seidel method. When $\beta < 1$ the system of equations is under-relaxed. Under-relaxation is appropriate when the Gauss-Seidel algorithm causes the solution vector to overshoot and move farther away from the exact solution. The iterative method diverges if $\beta > 2$. The relaxation factor does not change the final solution since it multiplies the residual, which is zero when the final solution is reached. The major difficulty with the over-relaxation method is the determination of the best value for the over-relaxation factor, β. Unfortunately, there is not a good general method for determining the optimum over-relaxation factor, β_{opt}. The optimumv alue of the over-relaxation factor β_{opt} depends on the size of the system of equations (i.e., the number of equations) and the nature of the equations (i.e., strength of the diagonal dominance, the structure of the coefficient matrix, etc.). As general rule, larger values of β_{opt} are associated with larger systems of equations. In general, one must resort to numerical experimentation to determine β_{opt}. The maximum rate of convergence is achieved when β_{opt} lies between 1 and 2.

A MATLAB implementation of the algorithm, shown in the Appendix, uses the Gauss-Seidel method for the time march iterations and the successive over-relaxation for the elliptic equation iterations.

5 Results

The script is used to solve a lid cavity flow. All parameters can be found labelled in the script. The final vorticity and stream function distribution for this problem is shown in Figure 2.

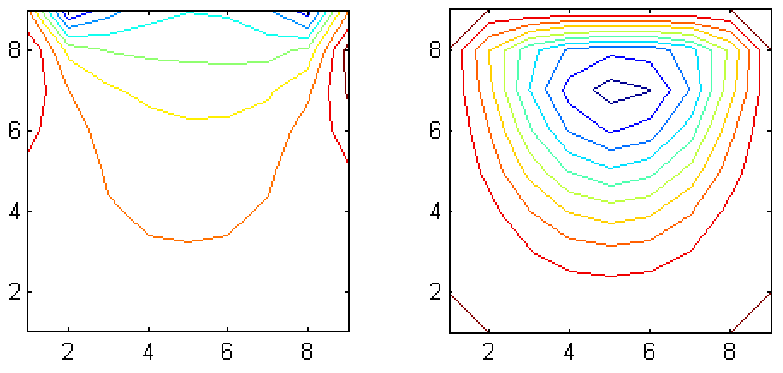

Figure 2 Left: Vorticity Stream Distribution, Right: Stream Function Distribution

6 Discussion

While the vorticity-stream function approach has seen considerable use for 2D incompressible flows, this formulation is not easily extendible to 3D. Moreover, velocity and pressure are not directly computed quantities. Boundary conditions related to vorticity are complicated. Thus, the vorticity-stream function formulation is limited to 2D and can be used effectively for simple problems.

7 Appendix: Lid Cavity Flow Vorticity-Stream Function Formulation MATLAB Script

```
clf;clear;clc;
% Problem Paramters
nu=0.1;                                                 % Kinematic viscosity
ul=1;                                                   % Lid velocity
% Grid Paramters
n=9;                                                    % imax=jmax=n
h=1/(n-1);                                              % Grid step size
% Time March Paramters
mts=60;                                                 % Maximum number of time steps
dt=.02;                                                 % Time step size
cts=0;                                                  % Current time step
% SOR Paramters
b=1.5;                                                  % Over-relaxation factor
mk=100;                                                 % Maximum number of SOR iterations
% Declare and Initialize
[v,s]=deal(zeros(n,n));                                 % Vorticity and stream function
% Time March Loop
while cts<mts
    % Compute the boundary nodes vorticity
    for i=2:n-1
        v(i,1)=2*(s(i,1)-s(i,2))/(h*h);                 % vorticity on bottom wall
        v(i,n)=2*(s(i,n)-s(i,n-1))/(h*h)-2*ul/h;        % vorticity on top wall
        v(1,i)=2*(s(1,i)-s(2,i))/(h*h);                 % vorticity on left wall
        v(n,i)=2*(s(1,i)-s(n-1,i))/(h*h);          % vorticity on right wall
    end
    % Compute the interior nodes vorticity
    for i=2:n-1
        for j=2:n-1
            v(i,j)=(4*s(i,j)-s(i+1,j)-s(i-1,j)-s(i,j+1)-s(i,j-1))/(h*h);
        end
    end
    % Compute the interior nodes new time step vorticity
    for i=2:n-1
        for j=2:n-1
            v(i,j)=v(i,j)+dt*(((((s(i+1,j)-s(i-1,j))*(v(i,j+1)-v(i,j-1)))/(4*h*h))...
                -(((v(i+1,j)-v(i-1,j))*(s(i,j+1)-s(i,j-1)))/(4*h*h))...
                +(v(i+1,j)+v(i-1,j)+v(i,j+1)+v(i,j-1)-4*v(i,j))*(nu/(h*h)));
        end
    end
    % Compute the interior nodes new time step stream function
    ck=0;                                          % Current iteration number
    while ck<mk
        for i=2:n-1
            for j=2:n-1
                s(i,j)=(1-b)*s(i,j)+(s(i+1,j)+s(i-1,j)+s(i,j+1)+s(i,j-
1)+v(i,j)*(h*h))*(b/4);
            end
        end
        ck=ck+1;
    end
    cts=cts+1;                                     % Increment current time
    % Time March Plot
    subplot(121), contour(rot90(fliplr(v))), axis('square');            % Vorticity
    subplot(122), contour(rot90(fliplr(s))), axis('square');            % Stream function
    pause(0.01)
end
```

8 Bibliography

1) Frank M. White, Fluid Mechanics, McGraw-Hill, 1999.
2) Richard W. Hamming, Numerical Methods for Scientists and Engineers, McGraw-Hill, 1973.
3) Link Accessed 14/12/2018
4) Link Accessed 14/12/2018
5) Link Accessed 14/12/2018
6) Link Accessed 14/12/2018