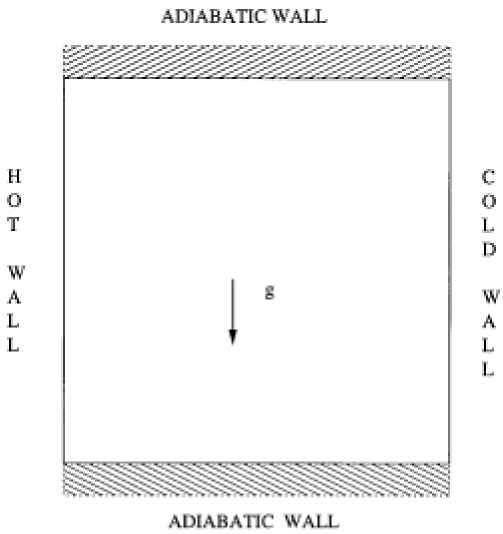
Vorticity-Streamfunction Method for Buoyancy Driven Cavity Probelm

Key words : vorticity-streamfunction method; buoyancy driven cavity flow

1. Problem



The dimensionless governing equations are:

$$\frac{\partial \omega}{\partial t}+u\frac{\partial \omega}{\partial x}+v\frac{\partial \omega}{\partial y}=\Pr(\frac{\partial^2 \omega}{\partial x^2}+\frac{\partial^2 \omega}{\partial y^2})-Ra*\Pr*\frac{\partial T}{\partial x}$$

$$\frac{\partial^2 \psi}{\partial x^2}+\frac{\partial^2 \psi}{\partial x^2}=w$$

$$\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}=\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}$$

where, streamfunction is defined as $u=\frac{\partial \psi}{\partial y}, v=-\frac{\partial \psi}{\partial x}$,

Vorticity (in Left hand cartesian system) is defined as $\nabla^2 \psi = \omega$

Initial boundary condition set as:

$$\omega = \psi = T = u = v = 0$$

$$\psi = 0, \quad T = 1 \qquad x = 0, 0 \le y \le 1$$

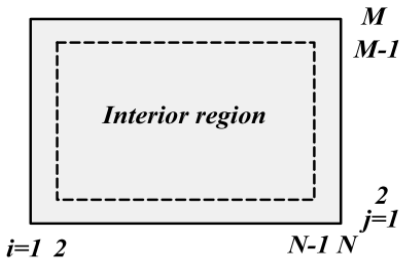
$$\psi = 0, \quad T = 0 \qquad x = 1, 0 \le y \le 1$$

$$\psi = 0, \frac{\partial T}{\partial y} = 0 \qquad y = 0, 0 < x < 1$$

2. Numerical Scheme

2.1 Mesh generation

Uniform mesh is used as



2.2 Discretization

2.2.1 Discretize the vorticity equation at all the interior points, for spatial discretization using central discretization scheme and for time discretization using explicit time advance method (Euler method).

$$\frac{\omega_{i,j}^{n+1} - \omega_{i,j}^{n}}{\Delta t} = \Pr\left(\frac{\omega_{i+1,j}^{n} - 2\omega_{i,j}^{n} + \omega_{i-1,j}^{n}}{\Delta x^2} + \frac{\omega_{i,j+1}^{n} - 2\omega_{i,j}^{n} + \omega_{i,j-1}^{n}}{\Delta y^2}\right) - Ra * \Pr \frac{T_{i+1,j}^{n} - T_{i-1,j}^{n}}{2\Delta x} - \left(u_{i,j}^{n} \frac{\omega_{i+1,j}^{n} - \omega_{i-1,j}^{n}}{2\Delta x} + v_{i,j}^{n} \frac{\omega_{i,j+1}^{n} - \omega_{i,j-1}^{n}}{2\Delta y}\right)$$

2.2.2 Discretize the streamfunction equation using central difference scheme

$$\frac{\psi_{i+1,j}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i-1,j}^{n+1}}{\Delta x^2} + \frac{\psi_{i,j+1}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i,j-1}^{n+1}}{\Delta y^2} = \omega_{i,j}^{n+1}$$

Employing implicit $\Delta - \boldsymbol{form}$ methd, Setting $;\ \psi_{i,j}^{n+1} = \psi_{i,j}^{n} + \Delta\psi_{i,j}^{n+1};$, then get

$$b_W \Delta\psi_{i-1,j}^{n+1} + b_E \Delta\psi_{i+1,j}^{n+1} + b_S \Delta\psi_{i,j-1}^{n+1} + b_N \Delta\psi_{i,j+1}^{n+1} + b_P \Delta\psi_{i,j}^{n+1} = S_{i,j}^{n}$$

Where

$$b_W = \frac{1}{\Delta x^2},$$
$$b_E = b_W,$$
$$b_S = \frac{1}{\Delta y^2},$$
$$b_N = b_S,$$
$$b_P = -2*(b_W + b_S),$$
$$S_{i,j}^n = \omega_{i,j}^{n+1} - \frac{\psi_{i+1,j}^n - 2\psi_{i,j}^n + \psi_{i-1,j}^n}{\Delta x^2} - \frac{\psi_{i,j+1}^n - 2\psi_{i,j}^n + \psi_{i,j-1}^n}{\Delta y^2}.$$

2.2.3 For velocity components,

$$u_{i,j}^{n+1} = \frac{\psi_{i,j+1}^{n+1} - \psi_{i,j-1}^{n+1}}{2\Delta y}, \qquad v_{i,j}^{n+1} = \frac{\psi_{i+1,j}^{n+1} - \psi_{i-1,j}^{n+1}}{-2\Delta x}$$

2.2.4 Energy equation is solved in the same way as vorticity equation:

$$\frac{T_{i,j}^{n+1} - T_{i,j}^n}{\Delta t} = \left(\frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2} + \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2}\right) - \left(u_{i,j}^{n+1}\frac{T_{i+1,j}^n - T_{i-1,j}^n}{2\Delta x} + v_{i,j}^{n+1}\frac{T_{i,j+1}^n - T_{i,j-1}^n}{2\Delta y}\right)$$

2.3 Boundary condition

2.3.1 Implementation of voticity condition using Taylor series expansion with 2nd order approximation:

$$\omega_{1,j}^{n+1} = \frac{3(\psi_{2,j}^n - \psi_{1,j}^n)}{\Delta x^2} - \frac{1}{2}\omega_{2,j}^n, \qquad \omega_{m,j}^{n+1} = \frac{3(\psi_{m-1,j}^n - \psi_{m,j}^n)}{\Delta x^2} - \frac{1}{2}\omega_{m-1,j}^n$$
$$\omega_{i,1}^{n+1} = \frac{3(\psi_{i,2}^n - \psi_{i,1}^n)}{\Delta x^2} - \frac{1}{2}\omega_{i,2}^n, \qquad \omega_{i,m}^{n+1} = \frac{3(\psi_{i,m-1}^n - \psi_{i,m}^n)}{\Delta x^2} - \frac{1}{2}\omega_{i,m-1}^n$$

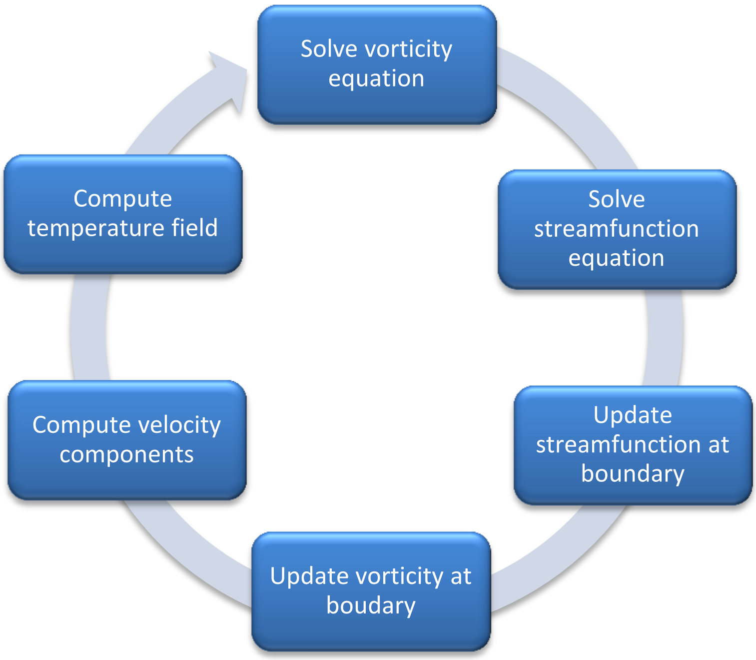
2.3.2 Implementation of streamfunction condition using 2nd order finite difference scheme to approximate the derivative condition:

$$\psi_{2,j} = \frac{1}{4}\psi_{3,j}$$,etc.

2.3.3 Implementation of temperature condition

$$T_{1,j} = 1, T_{m,j} = 0$$
$$T_{i,1} = \frac{4T_{i,2} - T_{i,3}}{3}, T_{i,m} = \frac{4T_{i,m-1} - T_{i,m-2}}{3}$$

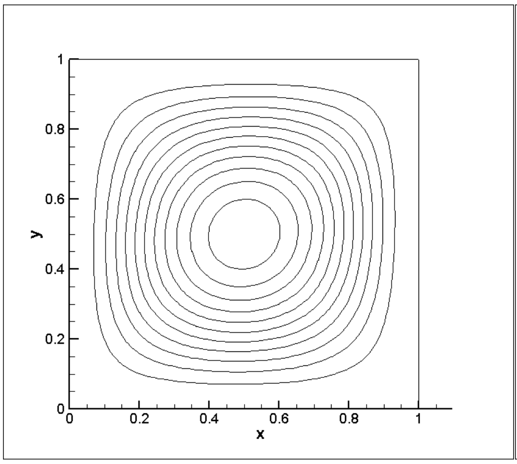
2.4 Flow chat

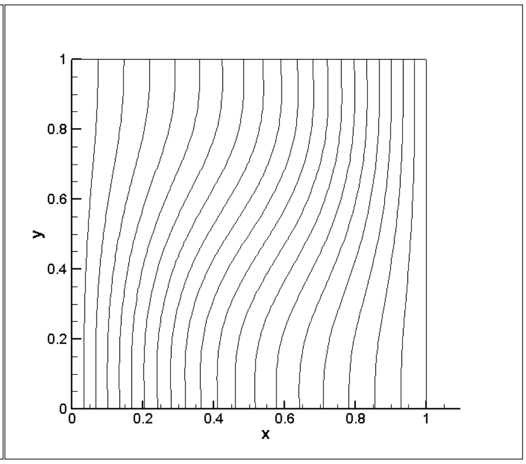


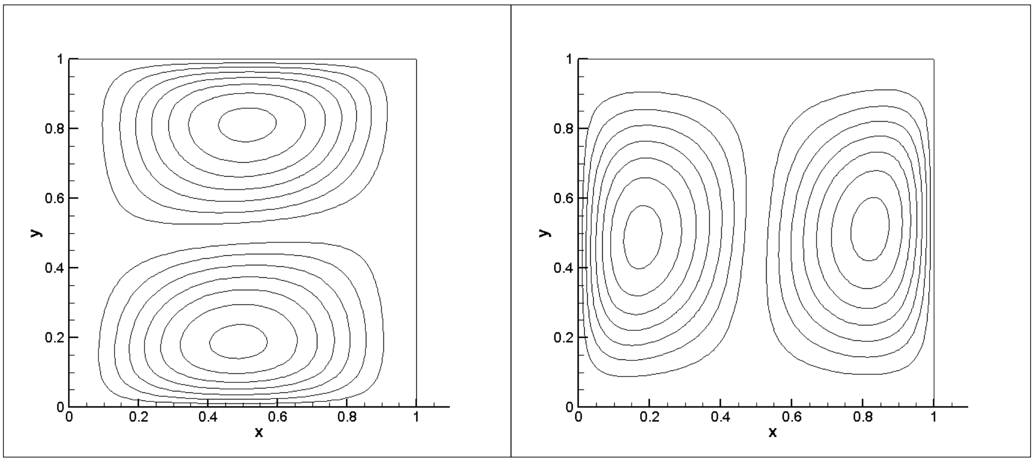
3. Results

3.1 Flow patterns (psi, T, U, V)

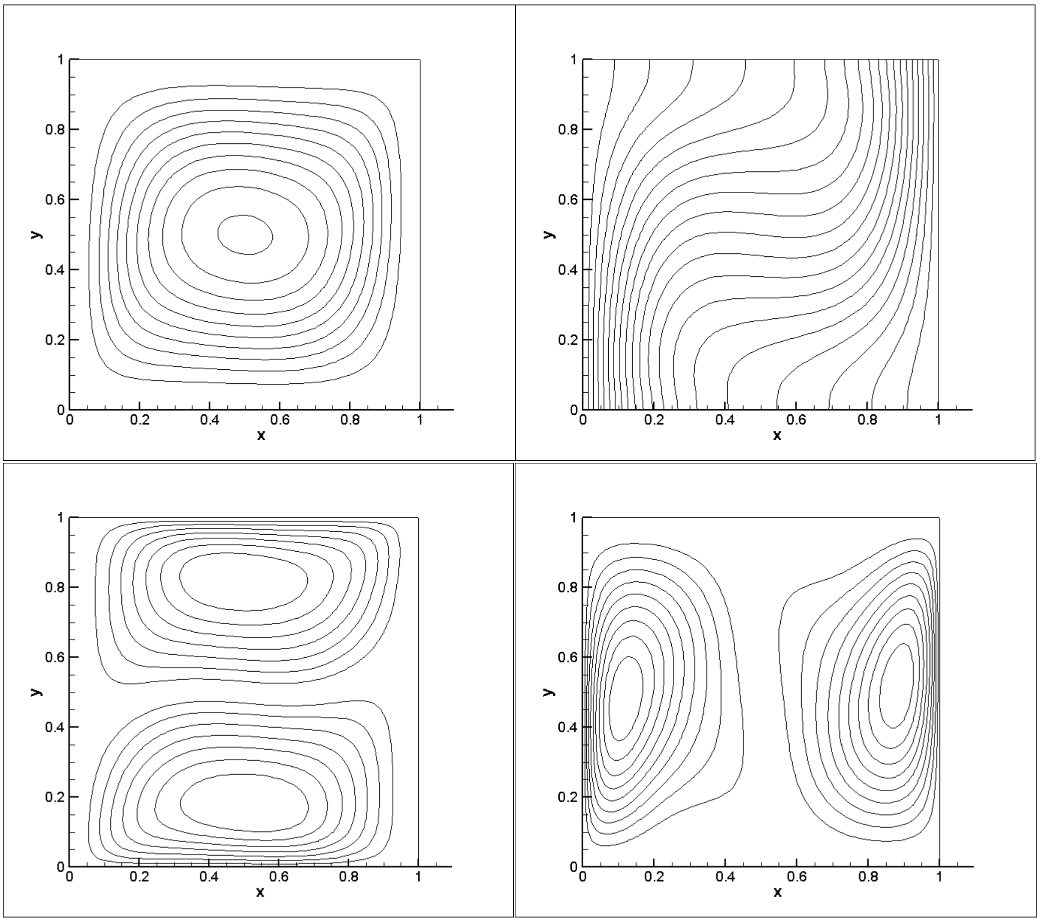
Ra=1e3,



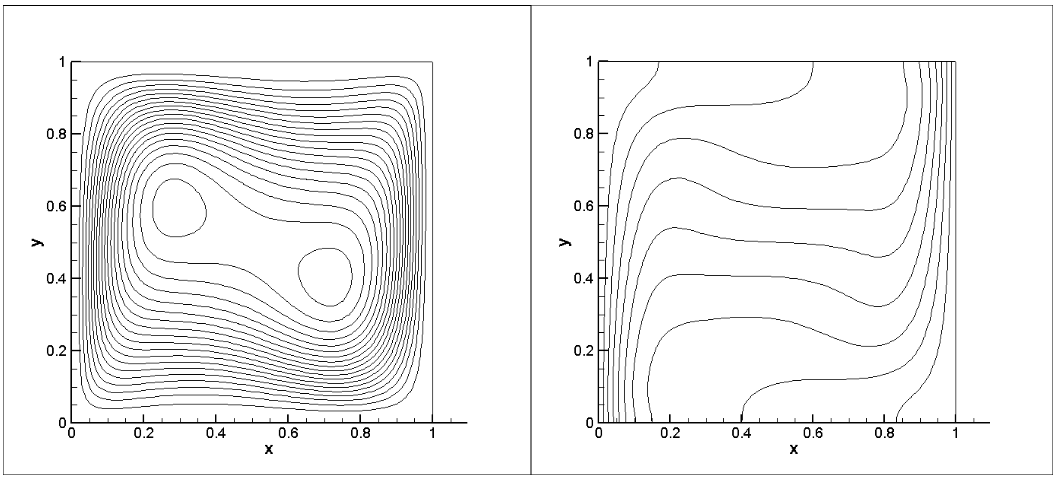


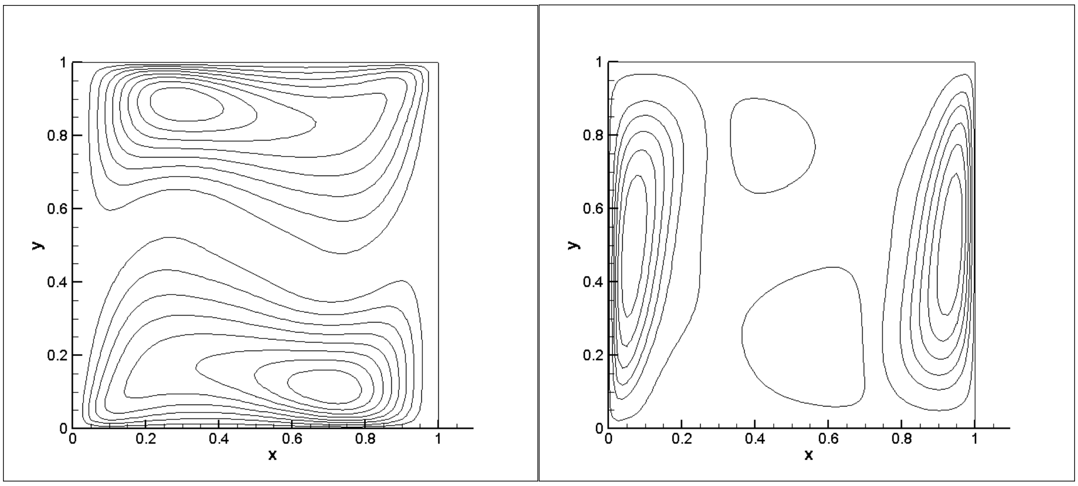


Ra=1e4,

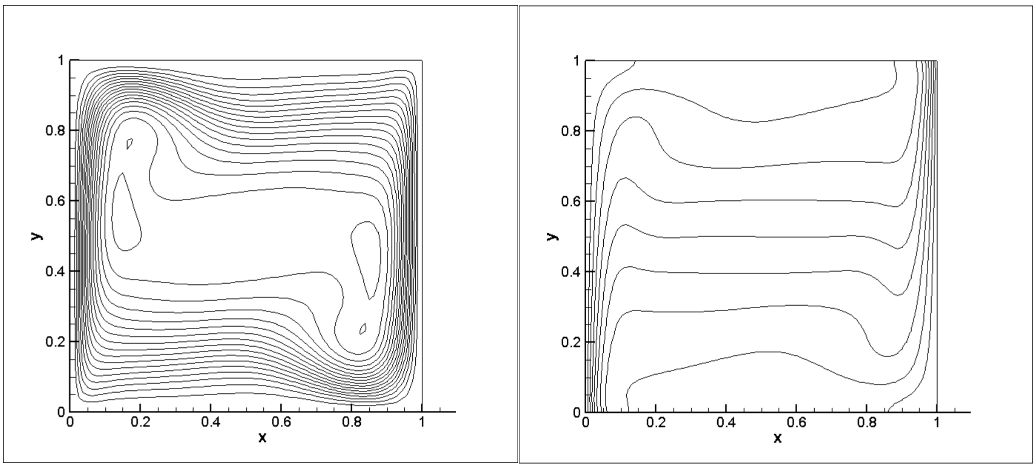


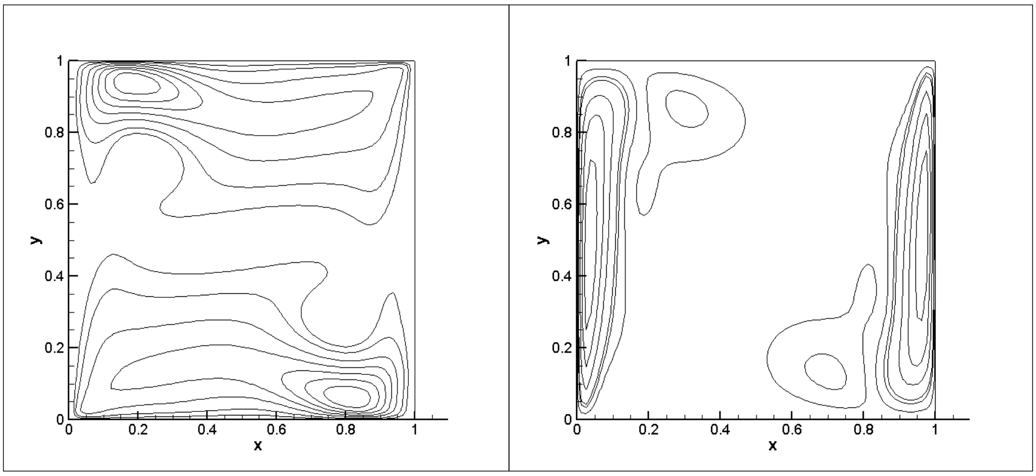
Ra=1e5,





Ra=1e6,





3.2 Validation:

Mesh 81*81

<table>
<tr><td>Ra</td><td>|Psi_mid|</td><td colspan="2">U_max
y</td><td colspan="2">V_max
x</td><td colspan="2">Nu_max
y</td><td colspan="2">Nu_min
y</td></tr>
<tr><td>1e3</td><td>1.178</td><td>3.650</td><td>0.812</td><td>3.713</td><td>0.175</td><td>1.513</td><td>0.087</td><td>0.688</td><td>1</td></tr>
<tr><td>1e4</td><td>5.093</td><td>16.260</td><td>0.825</td><td>19.744</td><td>0.112</td><td>3.580</td><td>0.137</td><td>0.581</td><td>1</td></tr>
<tr><td>1e5</td><td>9.222</td><td>35.762</td><td>0.862</td><td>69.641</td><td>0.062</td><td>8.008</td><td>0.075</td><td>0.716</td><td>1</td></tr>
<tr><td>1e6</td><td>16.915</td><td>67.802</td><td>0.850</td><td>232.826</td><td>0.0375</td><td>18.013</td><td>0.0375</td><td>0.950</td><td>1</td></tr>
</table>

Reference:

Natural Convection of Air in a Square Cavity : A Benchmark Numerical Solution

Mesh 41*41

<table>
<tr><td>Ra</td><td>|Psi_mid|</td><td colspan="2">U_max
y</td><td colspan="2">V_max
x</td><td colspan="2">Nu_max
y</td><td colspan="2">Nu_min
y</td></tr>
<tr><td>1e3</td><td>1.174</td><td>3.634</td><td>0.813</td><td>3.679</td><td>0.179</td><td>1.501</td><td>0.087</td><td>0.694</td><td>1</td></tr>
<tr><td>1e4</td><td>5.098</td><td>16.182</td><td>0.823</td><td>19.509</td><td>0.120</td><td>3.545</td><td>0.149</td><td>0.592</td><td>1</td></tr>
<tr><td>1e5</td><td>9.234</td><td>35.07</td><td>0.855</td><td>66.73</td><td>0.068</td><td>7.905</td><td>0.095</td><td>0.755</td><td>1</td></tr>
<tr><td>1e6</td><td>17.15</td><td>67.49</td><td>0.854</td><td>206.32</td><td>0.0423</td><td>17.947</td><td>0.0675</td><td>1.015</td><td>0.984</td></tr>
</table>

Source Codes:

```
!!!    This program sloves Buoyancy Driven Cavity Flow problem using Vorticity-Streamfunction Methods
!!!    This work is licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License.
!!!    Ao Xu, Profiles: <http://www.linkedin.com/pub/ao-xu/30/a72/a29>
```

```
!!!              Adiabatic Wall
!!!           |---------------|
!!!           |               |
!!!           |       |       |
!!!     Hot   |       |       | Cold
!!!     Wall  |       | g     | Wall
!!!           |       v       |
!!!           |               |
!!!           |---------------|
!!!              Adiabatic Wall

!!!     u(i,j), v(i,j)-------------velocity function
!!!     psi(i,j)-------------------stream function
!!!     vor(i,j)-------------------vorticity function
!!!     Rpsi(i,j)------------------psi^{n+1}_{i,j} - psi^{n}_{i,j}
!!!     RVOR(i,j)------------------(vor^{n+1}_{i,j}-vor^{n}_{i,j})/dt
!!!     T(i,j))--------------------Temperature field
!!!     RT(i,j)--------------------(T^{n+1}_{i,j}-T^{n}_{i,j})/dt

        program main
        implicit none
        integer, parameter :: N=81, M=81
        integer :: i, j, itc, itc_max, k
        real(8) :: dx, dy, Pr, Ra, dt, eps, error
        real(8) :: X(N), Y(M), u(N,M), v(N,M), vor(N,M), RVOR(N,M), psi(N,M),
Rpsi(N,M), T(N,M)

!!! input initial data
        Pr = 0.71d0
        Ra = 1e6
        dx = 1.0d0/(N-1)
        dy = 1.0d0/(M-1)
        dt = 1e-6
        eps = 1e-8
        itc = 0
        itc_max = 1e8
        error = 100.0d0
        k = 0

!!! set up initial flow field
        call initial(N,M,dx,dy,X,Y,u,v,psi,vor,RVOR,Rpsi,T)

        do while((error.GT.eps).AND.(itc.LT.itc_max))
!!! solve vorticity equation
```

```
            call solvor(N,M,dx,dy,Pr,Ra,dt,u,v,vor,RVOR,T)

!!! solve Streamfunction equation
            call solpsi(N,M,dx,dy,vor,psi,Rpsi)

!!! updates the values of sream function at boundary points
            call bcpsi(N,M,dy,psi)

!!! updates the boundary condition for vorticity
            call bcvor(N,M,dx,dy,vor,psi)

!!! compute velocity components u and v
            call caluv(N,M,dx,dy,psi,u,v)

!!! compute temperature field
            call calt(N,M,dt,dx,dy,u,v,T)

!!! check convergence
            call convergence(N,M,dt,RVOR,Rpsi,error,itc)

!!! output preliminary results
            if (MOD(itc,1000000).EQ.0) then
                k = k+1
                call output(N,M,X,Y,u,v,psi,T,k)
            endif

        enddo

!!! output data file
        call output(N,M,X,Y,u,v,psi,T,k)

        open(unit=03,file='results.txt',status='unknown')

        write(03,*)
        write(03,*)
'*************************************************************'
        write(03,*) 'This program sloves Buoyancy Driven Cavity Flow problem'
        write(03,*) 'using Vorticity-Streamfunction Methods'
        write(03,*) 'N =',N,',          M =',M
        write(03,*) 'Pr=',Pr
        write(03,*) 'Ra=',Ra
        write(03,*) 'dt =', dt
        write(03,*) 'eps =',eps
        write(03,*) 'itc =',itc
```

```
        write(03,*) 'Developing time=',dt*itc,'s'
        write(03,*)
'**********************************************************'
        write(03,*)

!!! validate results with reference
        call validation(N,M,dx,dy,u,v,psi,T)

        close(03)

        stop
        end program main


!!! set up initial flow field
        subroutine initial(N,M,dx,dy,X,Y,u,v,psi,vor,RVOR,Rpsi,T)
        implicit none
        integer :: i, j, N, M
        real(8) :: dx, dy
        real(8) :: X(N), Y(M), u(N,M), v(N,M), psi(N,M), vor(N,M), RVOR(N,M),
Rpsi(N,M), T(N,M)

        do i=1,N
            X(i) = (i-1)*dx
        enddo
        do j=1,M
            Y(j) = (j-1)*dy
        enddo

        do i=1,N
            do j=1,M
                u(i,j) = 0.0d0
                v(i,j) = 0.0d0!!!     u(i,j), v(i,j)------------velocity function
                psi(i,j) = 0.0d0!!!     psi(i,j)--------------------stream function
                Rpsi(i,j) = 0.0d0!!!     Rpsi(i,j)--------------psi^{n+1}_{i,j} - psi^{n}_{i,j}
                vor(i,j) = 0.0d0!!!     vor(i,j)------------------vorticity function
                RVOR(i,j) = 0.0d0!!!
RVOR(i,j)-------------(vor^{n+1}_{i,j}-vor^{n}_{i,j})/dt
                T(i,j) = 0.0d0!!!      T(i,j))--------------Temperature field
            enddo
        enddo

        do j =1,M
            T(1,j) = 1.0d0   !Left side hot wall
```

```
        enddo

        return
        end subroutine initial


!!! solve vorticity equation
        subroutine solvor(N,M,dx,dy,Pr,Ra,dt,u,v,vor,RVOR,T)
        implicit none
        integer :: i, j, N, M
        real(8) :: dx, dy, dt, Pr, Ra, dvorx2, dvory2, dvorx1, dvory1
        real(8) :: vor(N,M), u(N,M), v(N,M), RVOR(N,M), T(N,M)

        ! FTCS Sheme
        do i=2,N-1
            do j=2,M-1
                dvorx2 = (vor(i+1,j)-2.0d0*vor(i,j)+vor(i-1,j))/dx/dx
                dvory2 = (vor(i,j+1)-2.0d0*vor(i,j)+vor(i,j-1))/dy/dy
                dvorx1 = (u(i+1,j)*vor(i+1,j)-u(i-1,j)*vor(i-1,j))/2.0d0/dx
                dvory1 = (v(i,j+1)*vor(i,j+1)-v(i,j-1)*vor(i,j-1))/2.0d0/dy
                RVOR(i,j) =
(dvorx2+dvory2)*Pr-Ra*Pr*(T(i+1,j)-T(i-1,j))/2.0d0/dx-dvorx1-dvory1
                vor(i,j) = vor(i,j)+dt*RVOR(i,j)
            enddo
        enddo

        return
        end subroutine solvor


!!! solve Streamfunction equation
        subroutine solpsi(N,M,dx,dy,vor,psi,Rpsi)
        implicit none
        integer :: i, j ,N, M
        real(8) :: alpha, dx, dy, aw, as, ap
        real(8) :: vor(N,M), psi(N,M), Rpsi(N,M), S(N,M)

        aw = 1.0d0/dx/dx
        as = 1.0d0/dy/dy
        ap = -2.0d0*(as+aw)

        do i=3,N-2
            do j=3,M-2
                S(i,j) =
```

```
vor(i,j)-(psi(i+1,j)-2.0d0*psi(i,j)+psi(i-1,j))/dx/dx-(psi(i,j+1)-2.0d0*psi(i,j)
+psi(i,j-1))/dy/dy
            enddo
        enddo

        do j=1,M
            Rpsi(1,j) = 0.0d0
            Rpsi(2,j) = 0.0d0
            Rpsi(N,j) = 0.0d0
            Rpsi(N-1,j) = 0.0d0
        enddo
        do i=1,N
            Rpsi(i,1) = 0.0d0
            Rpsi(i,2) = 0.0d0
            Rpsi(i,M) = 0.0d0
            Rpsi(i,M-1) = 0.0d0
        enddo

        alpha = 1.5d0        !alpha is ralaxtion factor

        do i=3,N-2
            do j=3,M-2
                Rpsi(i,j)=(S(i,j)-aw*Rpsi(i-1,j)-as*Rpsi(i,j-1))/ap
                psi(i,j) = psi(i,j)+alpha*Rpsi(i,j)
            enddo
        enddo

        return
        end subroutine solpsi


!!! updates the values of sream function at boundary points
        subroutine bcpsi(N,M,dy,psi)
        implicit none
        integer :: i, j, N, M
        real(8) :: dy
        real(8) :: psi(N,M)

        do j=2,M-1
            psi(2,j) = 0.25d0*psi(3,j)
            psi(N-1,j) = 0.25d0*psi(N-2,j)
        enddo
        do i=2,N-1
            psi(i,2) = 0.25d0*psi(i,3)
```

```
        psi(i,M-1) = 0.25d0*psi(i,M-2)
    enddo


    return
    end subroutine bcpsi


!!! updates the boundary condition for vorticity
    subroutine bcvor(N,M,dx,dy,vor,psi)
    implicit none
    integer :: i, j, N, M
    real(8) :: dx, dy
    real(8) :: vor(N,M), psi(N,M)

    ! 2nd order approximation
    do j=1,M
        vor(1,j) = 3.0d0*psi(2,j)/dx/dx-0.5d0*vor(2,j)
        vor(N,j) = 3.0d0*psi(N-1,j)/dx/dx-0.5d0*vor(N-1,j)
    enddo
    do i=1,N
        vor(i,1) = 3.0d0*psi(i,2)/dy/dy-0.5d0*vor(i,2)
        vor(i,M) = 3.0d0*psi(i,M-1)/dy/dy-0.5d0*vor(i,M-1)
    enddo

    return
    end subroutine bcvor


!!! compute velocity components u and v
    subroutine caluv(N,M,dx,dy,psi,u,v)
    implicit none
    integer :: i, j, N, M
    real(8) :: dx, dy
    real(8) :: psi(N,M), u(N,M), v(N,M)

    !physical boundary condition
    do i=1,N
        u(i,1) = 0.0d0
        v(i,1) = 0.0d0
        u(i,M) = 0.0d0
        v(i,M) = 0.0d0
    enddo
    do j=1,M
```

```
        u(1,j) = 0.0d0
        u(N,j) = 0.0d0
        v(1,j) = 0.0d0
        v(N,j) = 0.0d0
    enddo

    do i=2,N-1
        do j=2,M-1
            u(i,j) = 0.5d0*(psi(i,j+1)-psi(i,j-1))/dy
            v(i,j) = -0.5d0*(psi(i+1,j)-psi(i-1,j))/dx
        enddo
    enddo

    return
    end subroutine caluv


!!! compute temperature field
    subroutine calt(N,M,dt,dx,dy,u,v,T)
    implicit none
    integer :: i, j, N, M
    real(8) :: dx, dy, dt, dTx2, dTy2, dTx1, dTy1
    real(8) :: T(N,M), u(N,M), v(N,M)

   ! Interior points using FTCS Sheme
    do i=2,N-1
        do j=2,M-1
            dTx2 = (T(i+1,j)-2*T(i,j)+T(i-1,j))/dx/dx
            dTy2 = (T(i,j+1)-2*T(i,j)+T(i,j-1))/dy/dy
            dTx1 = (u(i+1,j)*T(i+1,j)-u(i-1,j)*T(i-1,j))/2/dx
            dTy1 = (v(i,j+1)*T(i,j+1)-v(i,j-1)*T(i,j-1))/2/dy
            T(i,j) = T(i,j)+dt*(dTx2+dTy2-dTx1-dTy1)
        enddo
    enddo

   !Left and right side boundary(Dirichlet B.C.)
    do j=1,M
        T(1,j) = 1.0d0
        T(N,j) = 0.0d0
    enddo

   !Top and bottom side boundary(Neumann B.C.)
    do i=1,N
        T(i,1) = (4.0d0*T(i,2)-T(i,3))/3.0d0
```

```
            T(i,M) = (4.0d0*T(i,M-1)-T(i,M-2))/3.0d0
        enddo

        return
        end subroutine calt

!!! check convergence
        subroutine convergence(N,M,dt,RVOR,Rpsi,error,itc)
        implicit none
        integer :: N, M, i, j, itc
        real(8) :: RVOR(N,M), Rpsi(N,M)
        real(8) :: dt, error, errvor, errpsi

        itc = itc+1
        errvor = 0.0d0
        errpsi = 0.0d0

        do i=1,N
            do j=1,M
                if(ABS(RVOR(i,j))*dt.GT.errvor) errvor = ABS(RVOR(i,j))*dt
                if(ABS(Rpsi(i,j)).GT.errpsi) errpsi = ABS(Rpsi(i,j))
            enddo
        enddo

        error = MAX(errvor,errpsi)
        if(itc.EQ.1) error = 100.0d0

        open(unit=01,file='error.dat',status='unknown',position='append')

        if (MOD(itc,2000).EQ.0) then
            write(01,*) itc,' ',error
        endif

        close(01)

        return
        end subroutine convergence


!!! validate results with reference
        subroutine validation(N,M,dx,dy,u,v,psi,T)
        implicit none
        integer :: N, M, i, j
        integer :: mid_x, mid_y, temp, temp_max, temp_min
```

```
        real(8) :: dx, dy
        real(8) :: psi_mid, u_max, v_max, u_max_loc , v_max_loc, Nu_max,
Nu_min, Nu_max_loc, Nu_min_loc
        real(8) :: u(N,M), v(N,M), psi(N,M), T(N,M)
        real(8) :: Nu(N)

        mid_x = INT(N/2)
        mid_y = INT(M/2)
        psi_mid = psi(mid_x,mid_y)

        u_max = 0.0d0
        v_max = 0.0d0
        Nu_max = 0.0d0
        Nu_min = 100.0d0
        temp = 0
        temp_max = 0
        temp_min = 0

        do j=1,M
            if(u(mid_x,j).GT.u_max) then
                u_max = u(mid_x,j)
                temp = j
            endif
        enddo
        u_max_loc = (temp-1)*dy

        do i=1,N
            if(v(i,mid_y).GT.v_max) then
                v_max = v(i,mid_y)
                temp = i
            endif
        enddo
        v_max_loc = (temp-1)*dx

        do j=1,M
            !!!Nu(j) = -(-11.0d0*T(1,j)+18.0d0*T(2,j)-9.0d0*T(3,j)+2.0d0*T(4,j))/6.0d0/dy
            !!!Nu(j) = -(-3.0d0*T(1,j)+4.0d0*T(2,j)-T(3,j))/2.0d0/dy
            Nu(j) = -(T(2,j)-T(1,j))/dx
            if(Nu(j).GT.Nu_max) then
                Nu_max = Nu(j)
                temp_max = j
            elseif(Nu(i).LT.Nu_min) then
                Nu_min = Nu(j)
                temp_min = j
```

```fortran
            endif
        enddo
        Nu_max_loc = (temp_max-1)*dx
        Nu_min_loc = (temp_min-1)*dx

        write(03,*)
        write(03,*) 'psi_mid =',psi_mid
        write(03,*) 'u_max =',u_max,'at y =',u_max_loc
        write(03,*) 'v_max =',v_max,'at x =',v_max_loc
        write(03,*) 'Nu_max =',Nu_max,'at y=',Nu_max_loc
        write(03,*) 'Nu_min =',Nu_min,'at y=',Nu_min_loc
        write(03,*)

        return
        end subroutine validation

!!! output data file
        subroutine output(N,M,X,Y,u,v,psi,T,k)
        implicit none
        integer :: N, M, i, j, k
        real(8) :: X(N), Y(M), u(N,M), v(N,M), psi(N,M),T(N,M)
        character*16 filename

        filename='0000cavity.dat'
        filename(1:1) = CHAR(ICHAR('0')+MOD(k/1000,10))
        filename(2:2) = CHAR(ICHAR('0')+MOD(k/100,10))
        filename(3:3) = CHAR(ICHAR('0')+MOD(k/10,10))
        filename(4:4) = CHAR(ICHAR('0')+MOD(k,10))

        open(unit=02,file=filename,status='unknown')
        write(02,101)
        write(02,102)
        write(02,103) N, M
        do j=1,M
            do i = 1,N
                write(02,100) X(i), Y(j), u(i,j), v(i,j), psi(i,j), T(i,j)
            enddo
        enddo

100     format(2x,10(e12.6,'      '))
101     format('Title="Buoyancy Driven Cavity Flow(Vorticity-Streamfunction
Methods)"')
102     format('Variables=x,y,u,v,psi,T')
103     format('zone',1x,'i=',1x,i5,2x,'j=',1x,i5,1x,'f=point')
```

```
close(02)

return
end subroutine output
```