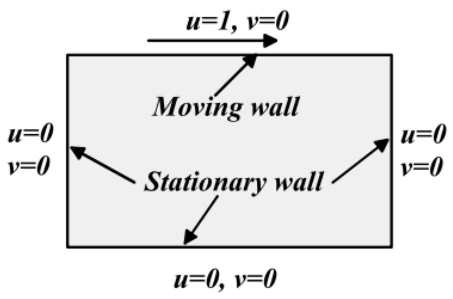
QUICK Scheme for Lid Driven Cavity Problem

Key words: QUICK Scheme; artificial compressibility method; lid driven cavity flow

1. Problem



The dimensionless governing equations are:

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

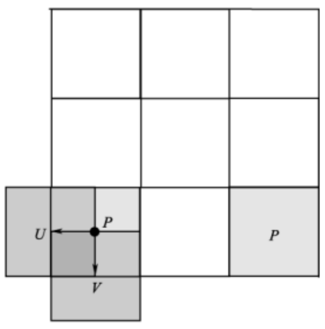
$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{\partial P}{\partial x}+\frac{1}{\mathrm{Re}}\nabla^2 u,$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{\partial P}{\partial y}+\frac{1}{\mathrm{Re}}\nabla^2 v.$$

2. Numerical Scheme

2.1 Mesh generation

Staggered mesh is employed:



2.2 Discretization

2.2.1 Integrate N-S equation in control volume,

$$\int_{\Delta V}\frac{\partial}{\partial x}(\rho u\phi)\,dV+\int_{\Delta V}\frac{\partial}{\partial y}(\rho v\phi)\,dV=\int_{\Delta V}\frac{\partial}{\partial x}\left(\mu\frac{\partial\phi}{\partial x}\right)dV+\int_{\Delta V}\frac{\partial}{\partial y}\left(\mu\frac{\partial\phi}{\partial y}\right)dV+\int_{\Delta V}S_\phi dV$$

Using Gauss law,

$$\begin{aligned}&\left[(\rho u\phi A)_e-(\rho u\phi A)_w\right]+\left[(\rho v\phi A)_n-(\rho v\phi A)_s\right]\\&\quad=\left(\mu A\frac{\partial\phi}{\partial x}\right)_e-\left(\mu A\frac{\partial\phi}{\partial x}\right)_w+\left(\mu A\frac{\partial\phi}{\partial y}\right)_n-\left(\mu A\frac{\partial\phi}{\partial y}\right)_s+\bar{S}_\phi\Delta x\Delta y\end{aligned}$$

Define

$$F_e=(\rho u)_e A_e,\quad F_w=(\rho u)_w A_w$$
$$F_n=(\rho v)_n A_n,\quad F_s=(\rho v)_s A_s$$
$$D_e=\frac{\mu_e A_e}{\delta x_{PE}},\quad D_w=\frac{\mu_w A_w}{\delta x_{PW}},$$
$$D_n=\frac{\mu_n A_n}{\delta y_{PN}},\quad D_s=\frac{\mu_s A_s}{\delta y_{PS}}$$

,then the momentum equation becomes

$$\begin{aligned}F_e\phi_e-F_w\phi_w+F_n\phi_n-F_s\phi_s&=D_e(\phi_E-\phi_P)-D_w(\phi_P-\phi_W)+\\&\quad D_n(\phi_N-\phi_P)-D_s(\phi_P-\phi_S)+\bar{S}_\phi\Delta x\Delta y\end{aligned}$$

For 3rd order upwind scheme(QUICK Scheme), the discretization form is

$$\begin{aligned}a_P\phi_P&=a_W\phi_W+a_E\phi_E+a_{EE}\phi_{EE}+a_{EE}\phi_{EE}+\\&\quad a_S\phi_S+a_N\phi_N+a_{SS}\phi_{SS}+a_{NN}\phi_{NN}+\bar{S}_\phi\Delta V\end{aligned}$$

where the common coefficient is:

$$a_W=D_w+\frac{6}{8}\alpha_w F_w+\frac{1}{8}\alpha_e F_e+\frac{3}{8}(1-\alpha_w)F_w$$
$$a_{WW}=-\frac{1}{8}\alpha_w F_w$$
$$a_E=D_e-\frac{3}{8}\alpha_e F_e-\frac{6}{8}(1-\alpha_e)F_e-\frac{1}{8}(1-\alpha_w)F_w$$
$$a_{EE}=\frac{1}{8}(1-\alpha_e)F_e$$
$$a_S=D_s+\frac{6}{8}\alpha_s F_s+\frac{1}{8}\alpha_n F_n+\frac{3}{8}(1-\alpha_s)F_s$$
$$a_{SS}=-\frac{1}{8}\alpha_s F_s$$
$$a_N=D_n-\frac{3}{8}\alpha_n F_n-\frac{6}{8}(1-\alpha_n)F_n-\frac{1}{8}(1-\alpha_s)F_s$$
$$a_{NN}=\frac{1}{8}(1-\alpha_n)F_n$$

$$a_P=a_W+a_E+a_{WW}+a_{EE}+a_S+a_N+a_{SS}+a_{NN}+(F_e-F_w+F_n-F_s)$$

2.2.2 Discretize the source term (including derivative of time and pressure gradient)

$$\int_{\Delta V} \bar{S}_\phi \cdot dV = -\frac{(\rho u)_P^{n+1} - (\rho u)_P^n}{\Delta t} \Delta x \Delta y - (p_e - p_w)\Delta y$$

2.2.3 For the computation of pressure field, Artificial Compressibility Method is adopted:

$$\frac{p_P^{n+1} - p_P^n}{\beta\,\Delta t} + \left[\frac{\delta(\rho u_i)}{\delta x_i}\right]_P^{n+1} = 0\,.$$

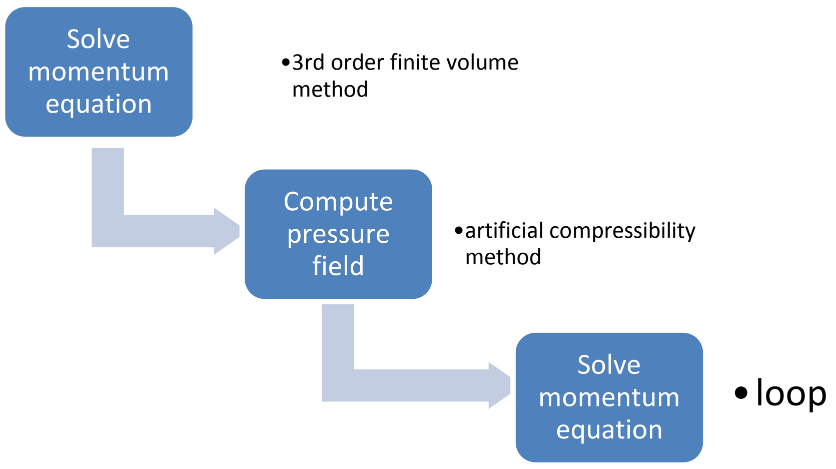
2.3 Boundary condition

As staggered mesh is used, the velocity boundary condition can be implemented as

```
!!! compute exterior region boundary with physical boundary condition
do i=2,N-1
    un(i,1) = -un(i,2)
    un(i,M+1) = 2.0-un(i,M)
enddo
do j=1,M+1
    un(1,j) = 0.0d0
    un(N,j) = 0.0d0
enddo

do i=2,N
    vn(i,1) = 0.0
    vn(i,M) = 0.0
enddo
do j=1,M
    vn(1,j) = -vn(2,j)
    vn(N+1,j) = -vn(N,j)
enddo
```

2.4 Flow chart

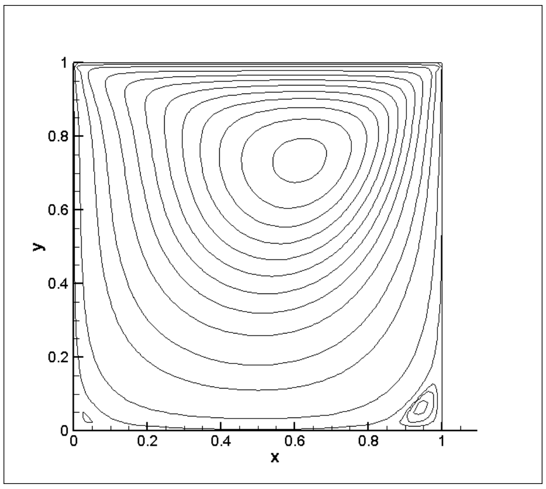


3. Results

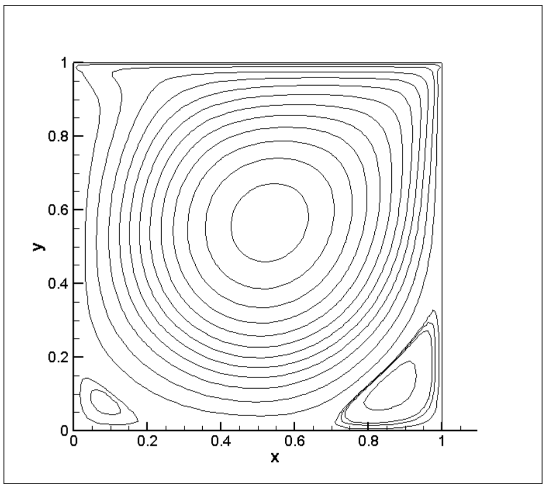
Flow patterns (psi)

Mesh 81*81

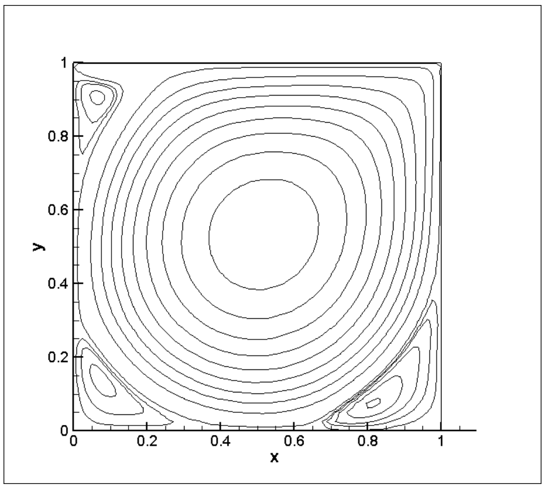
Re=100,



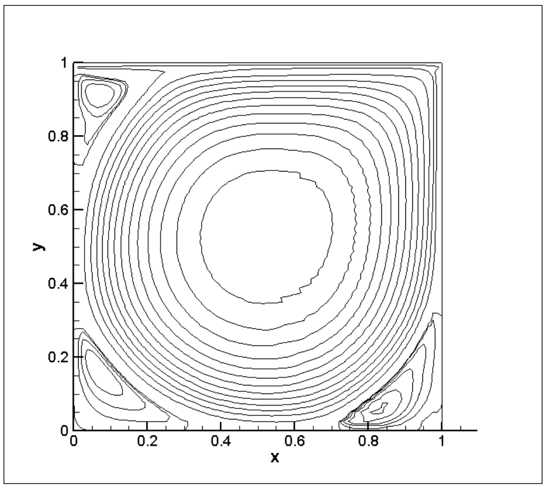
Re=1000,



Re=5000,



Re=10000,



<u>Source codes:</u>

```
!!!     This program sloves Lid Driven Cavity Flow problem using Artificial Compressibility Methods
!!!     Solve Momentum Equation with QUICK Scheme
!!!     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>


!!!                    Moving Wall
!!!                 |---------------|
!!!                 |               |
!!!                 |               |
!!!     Stationary  |               | Stationary
!!!        Wall     |               |     Wall
!!!                 |               |
!!!                 |               |
!!!                 |---------------|
!!!                  Stationary Wall


        program main
        implicit none
        integer, parameter :: N=81,M=81
        integer :: itc, itc_max, k
        real(8) :: u(N,M+1),v(N+1,M),p(N+1,M+1),psi(N,M),X(N), Y(M)
        real(8) ::
un(N,M+1),vn(N+1,M),pn(N+1,M+1),uc(N,M),vc(N,M),pc(N,M)
```

```
        real(8) :: c, c2, Re, dt, dx, dy, eps, error

!!! input initial data
        c = 1.5d0
        c2 = c*c     ! c2 = 2.25d0
        Re = 10000.0d0
        dt = 1e-5
        dx = 1.0d0/float(N-1)
        dy = 1.0d0/float(M-1)
        eps = 1e-8
        itc = 0
        itc_max = 1e7
        error=100.00d0
        k = 0

!!! set up initial flow field
        call initial(N,M,dx,dy,X,Y,u,v,p,psi)

        do while((error.GT.eps).AND.(itc.LT.itc_max))

            error=0.0d0

!!! Solve Momentum Equation with QUICK Scheme
            call quick(N,M,dx,dy,dt,Re,u,v,p,un,vn)

!!! Solve Continuity Equation
            call calpn(N,M,dx,dy,dt,c2,p,un,vn,pn)

!!! check convergence
            call check(N,M,dt,c2,error,u,v,p,un,vn,pn,itc)

!!! output preliminary results
            if (MOD(itc,100000).EQ.0) then
                call caluvp(N,M,u,v,p,uc,vc,pc)
                call calpsi(N,M,dx,dy,uc,vc,psi)
                k = k+1
                call output(N,M,X,Y,uc,vc,psi,k)
            endif

        enddo

!!! compute velocity components u, v and pressure p
        call caluvp(N,M,u,v,p,uc,vc,pc)
```

```
!!! compute Streamfunction
        call calpsi(N,M,dx,dy,uc,vc,psi)

!!! output data file
        call output(N,M,X,Y,uc,vc,psi,k)

        write(*,*)
        write(*,*)
'************************************************************'
        write(*,*) 'This program sloves Lid Driven Cavity Flow problem'
        write(*,*) 'using Artificial Compressibility Methods'
        write(*,*) 'N =',N,',        M =',M
        write(*,*) 'Re =',Re
        write(*,*) 'dt =',dt
        write(*,*) 'c (Artificial Compressibility coefficient) =',c
        write(*,*) 'eps =',eps
        write(*,*) 'itc =',itc
        write(*,*) 'Developing time=',dt*itc,'s'
        write(*,*)
'************************************************************'
        write(*,*)

        stop
        end program main


!!! set up initial flow field
        subroutine initial(N,M,dx,dy,X,Y,u,v,p,psi)
        implicit none
        integer :: N, M, i, j
        real(8) :: dx, dy
        real(8) :: u(N,M+1), v(N+1,M), p(N+1,M+1), psi(N,M), X(N), Y(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+1
            do j=1,M+1
                p(i,j) = 1.0d0
            enddo
        enddo
        do i=1,N
            do j=1,M+1
                u(i,j) = 0.0
                if(j.EQ.M+1) u(i,j) = 4.0d0/3.0d0
                if(j.EQ.M) u(i,j) = 2.0d0/3.0d0
            enddo
        enddo
        do i=1,N+1
            do j=1,M
                v(i,j) = 0.0d0
            enddo
        enddo

        do i=1,N
            do j=1,M
                psi(i,j) = 0.0d0
            enddo
        enddo

        return
        end subroutine initial


!!! Solve Momentum Equation with QUICK Scheme
        subroutine quick(N,M,dx,dy,dt,Re,u,v,p,un,vn)
        implicit none
        integer :: N, M, i, j
        real(8) :: u(N,M+1),v(N+1,M),p(N+1,M+1),un(N,M+1),vn(N+1,M)
        real(8) :: miu, Re, dx, dy, dt
        real(8) :: fw, fe, fs, fn, df, aw, aww, ae, aee, as, ass, an, ann,ap
        real(8) :: alpha

        miu = 1.0/Re

!!!!!!!!!!!!!!!!!!!!compute x-direction velocity component un!!!!!!!!!!!!!!!!!!!!
        do i=3,N-2
            do j=3,M-1

                fw = 0.5d0*(u(i-1,j)+u(i,j))*dy
                fe = 0.5d0*(u(i,j)+u(i+1,j))*dy
                fs = 0.5d0*(v(i,j-1)+v(i+1,j-1))*dx
                fn = 0.5d0*(v(i,j)+v(i+1,j))*dx
                df = fe-fw+fn-fs
```

```
                    !!! common coefficient in 3rd-order upwind QUICK Scheme
                    aw =
miu+0.75d0*alpha(fw)*fw+0.125d0*alpha(fe)*fe+0.375d0*(1.0d0-alpha(fw))
*fw
                    aww = -0.125d0*alpha(fw)*fw
                    ae =
miu-0.375d0*alpha(fe)*fe-0.75d0*(1.0-alpha(fe))*fe-0.125d0*(1.0d0-alpha(f
w))*fw
                    aee = 0.125d0*(1.0d0-alpha(fe))*fe
                    as =
miu+0.75d0*alpha(fs)*fs+0.125d0*alpha(fn)*fn+0.375d0*(1.0d0-alpha(fs))*f
s
                    ass = -0.125d0*alpha(fs)*fs
                    an =
miu-0.375d0*alpha(fn)*fn-0.75d0*(1.0-alpha(fn))*fn-0.125d0*(1.0d0-alpha(f
s))*fs
                    ann = 0.125d0*(1.0d0-alpha(fn))*fn
                    ap = aw+ae+as+an+aww+aee+ass+ann+df

                    un(i,j) = u(i,j) +
dt/dx/dy*( -ap*u(i,j)+aw*u(i-1,j)+ae*u(i+1,j)+aww*u(i-2,j)+aee*u(i+2,j)&

+as*u(i,j-1)+an*u(i,j+1)+ass*u(i,j-2)+ann*u(i,j+2) ) -
dt*(p(i+1,j)-p(i,j))/dx

               enddo
          enddo

          !!! compute interior region boundary with 1st-order upwind discrete scheme
          j=2
          do i=3,N-2
               call upbound_u(N,M,dx,dy,dt,Re,u,v,p,un,i,j)
          enddo
          j=M
          do i=3,N-2
               call upbound_u(N,M,dx,dy,dt,Re,u,v,p,un,i,j)
          enddo
          i=2
          do j=2,M
               call upbound_u(N,M,dx,dy,dt,Re,u,v,p,un,i,j)
          enddo
          i=N-1
          do j=2,M
               call upbound_u(N,M,dx,dy,dt,Re,u,v,p,un,i,j)
```

```
        enddo

        !!! compute exterior region boundary with physical boundary condition
        do i=2,N-1
            un(i,1) = -un(i,2)
            un(i,M+1) = 2.0-un(i,M)
        enddo
        do j=1,M+1
            un(1,j) = 0.0d0
            un(N,j) = 0.0d0
        enddo
!!!!!!!!!!!!!!!!!!!!compute x-direction velocity component un!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!compute y-direction velocity component vn!!!!!!!!!!!!!!!!!!!!
        do i=3,N-1
            do j=3,M-2

                fw = 0.5d0*(u(i-1,j)+u(i-1,j+1))*dy
                fe = 0.5d0*(u(i,j)+u(i,j+1))*dy
                fs = 0.5d0*(v(i,j-1)+v(i,j))*dx
                fn = 0.5d0*(v(i,j)+v(i,j+1))*dx
                df = fe-fw+fn-fs

                !!! common coefficient in 3rd-order upwind QUICK Scheme
                aw =
miu+0.75d0*alpha(fw)*fw+0.125d0*alpha(fe)*fe+0.375d0*(1.0d0-alpha(fw))
*fw
                aww = -0.125d0*alpha(fw)*fw
                ae =
miu-0.375d0*alpha(fe)*fe-0.75d0*(1.0-alpha(fe))*fe-0.125d0*(1.0d0-alpha(f
w))*fw
                aee = 0.125d0*(1.0d0-alpha(fe))*fe
                as =
miu+0.75d0*alpha(fs)*fs+0.125d0*alpha(fn)*fn+0.375d0*(1.0d0-alpha(fs))*f
s
                ass = -0.125d0*alpha(fs)*fs
                an =
miu-0.375d0*alpha(fn)*fn-0.75d0*(1.0-alpha(fn))*fn-0.125d0*(1.0d0-alpha(f
s))*fs
                ann = 0.125d0*(1.0d0-alpha(fn))*fn
                ap = aw+ae+as+an+aww+aee+ass+ann+df

                vn(i,j) = v(i,j) +
dt/dx/dy*( -ap*v(i,j)+aw*v(i-1,j)+ae*v(i+1,j)+aww*v(i-2,j)+aee*v(i+2,j)
&

+as*v(i,j-1)+an*v(i,j+1)+ass*v(i,j-2)+ann*v(i,j+2) ) -
dt*(p(i,j+1)-p(i,j))/dy


            enddo
        enddo

        !!! compute interior region boundary with 1st-order upwind discrete scheme
        j=2
        do i=3,N-1
            call upbound_v(N,M,dx,dy,dt,Re,u,v,p,vn,i,j)
        enddo
        j=M-1
        do i=3,N-1
            call upbound_v(N,M,dx,dy,dt,Re,u,v,p,vn,i,j)
        enddo
        i=2
        do j=2,M-1
            call upbound_v(N,M,dx,dy,dt,Re,u,v,p,vn,i,j)
        enddo
        i=N
        do j=2,M-1
            call upbound_v(N,M,dx,dy,dt,Re,u,v,p,vn,i,j)
        enddo

       !!! compute exterior region boundary with physical boundary condition
        do i=2,N
            vn(i,1) = 0.0
            vn(i,M) = 0.0
        enddo
        do j=1,M
            vn(1,j) = -vn(2,j)
            vn(N+1,j) = -vn(N,j)
        enddo
!!!!!!!!!!!!!!!!!!!!!compute y-direction velocity component vn!!!!!!!!!!!!!!!!!!!!!

        return
        end subroutine quick


!!! if(f_k.GT.0) then alpha_k = 1     (k=w,e,s,n)
!!! if(f_k.LT.0) then alpha_k = 0     (k=w,e,s,n)
```

```
        function alpha(x)
        implicit none
        real(8) :: alpha, x

        if(x.GT.0.0d0) then
            alpha = 1.0d0
        elseif(x.LT.0.0d0) then
            alpha = 0.0d0
        endif

        return
        end function alpha


!!! compute interior region boundary with 1st-order upwind discrete scheme-->un
        subroutine upbound_u(N,M,dx,dy,dt,Re,u,v,p,un,i,j)
        implicit none
        integer :: N, M, i, j
        real(8) :: dx, dy, dt, Re, miu
        real(8) :: u(N,M+1),v(N+1,M),p(N+1,M+1),un(N,M+1)
        real(8) :: aw, ae, as, an, df, ap

        miu = 1.0d0/Re

        aw = miu+MAX(0.5d0*(u(i-1,j)+u(i,j))*dy,0.0d0)
        ae = miu+MAX(0.0d0,-0.5d0*(u(i,j)+u(i+1,j))*dy)
        as = miu+MAX(0.5d0*(v(i,j-1)+v(i+1,j-1))*dx,0.0d0)
        an = miu+MAX(0.0d0,-0.5d0*(v(i,j)+v(i+1,j))*dx)
        df =
0.5d0*(u(i+1,j)-u(i-1,j))*dy+0.5*(v(i,j)+v(i+1,j)-v(i,j-1)-v(i+1,j-1))*dx
        ap = aw+ae+as+an+df

        un(i,j) =
u(i,j)+dt/dx/dy*(-ap*u(i,j)+aw*u(i-1,j)+ae*u(i+1,j)+as*u(i,j-1)+an*u(i,j+
1))-dt*(p(i+1,j)-p(i,j))/dx

        return
        end subroutine upbound_u


!!! compute interior region boundary with 1st-order upwind discrete scheme-->vn
        subroutine upbound_v(N,M,dx,dy,dt,Re,u,v,p,vn,i,j)
        implicit none
        integer :: N, M, i, j
        real(8) :: dx, dy, dt, Re, miu
        real(8) :: u(N,M+1),v(N+1,M),p(N+1,M+1),vn(N+1,M)
        real(8) :: aw, ae, as, an, df, ap

        miu = 1.0d0/Re

        aw = miu+MAX(0.5d0*(u(i-1,j)+u(i-1,j+1))*dy,0.0d0)
        ae = miu+MAX(0.0d0,-0.5d0*(u(i,j)+u(i,j+1))*dy)
        as = miu+MAX(0.5d0*(v(i,j-1)+v(i,j))*dx,0.0d0)
        an = miu+MAX(0.0d0,-0.5d0*(v(i,j)+v(i,j+1))*dx)
        df =
0.5d0*(u(i,j)+u(i,j+1)-u(i-1,j)-u(i-1,j+1))*dy+0.5*(v(i,j+1)-v(i,j-1))*dx
        ap = aw+ae+as+an+df

        vn(i,j) = v(i,j)+dt/dx/dy*(-ap*v(i,j)+aw*v(i-1,j)+ae*v(i+1,j)
+as*v(i,j-1)+an*v(i,j+1))-dt*(p(i,j+1)-p(i,j))/dy

        return
        end subroutine upbound_v


!!! Solve Continuity Equation
        subroutine calpn(N,M,dx,dy,dt,c2,p,un,vn,pn)
        implicit none
        integer :: N, M, i, j
        real(8) :: p(N+1,M+1), un(N,M+1), vn(N+1,M), pn(N+1,M+1)
        real(8) :: dx, dy, dt, c2

        do i=2,N
            do j=2,M
                pn(i,j) = p(i,j)-dt*c2*(   ( un(i,j)-un(i-1,j) )/dx +
( vn(i,j)-vn(i,j-1) ) /dy   )
            enddo
        enddo

        !!! boundary condition
        do i=2,N
            pn(i,1) = pn(i,2)
            pn(i,M+1) = pn(i,M)
        enddo
        do j=1,M+1
            pn(1,j) = pn(2,j)
            pn(N+1,j) = pn(N,j)
        enddo
        return
        end subroutine calpn


!!! check convergence
        subroutine check(N,M,dt,c2,error,u,v,p,un,vn,pn,itc)
        implicit none
        integer :: N, M, i, j, itc
        real(8) :: dt, c2, error, temp
        real(8) :: u(N,M+1), v(N+1,M), p(N+1,M+1), un(N,M+1), vn(N+1,M),
pn(N+1,M+1)
        real(8) :: erru, errv, errp

        itc = itc+1
        erru = 0.0d0
        errv = 0.0d0
        errp = 0.0d0

        do i=1,N
            do j=1,M+1
                temp = ABS(un(i,j)-u(i,j))/dt
                if(temp.GT.erru) erru = temp
                u(i,j) = un(i,j)
            enddo
        enddo

        do i=1,N+1
            do j=1,M
                temp = ABS(vn(i,j)-v(i,j))/dt
                if(temp.GT.errv) errv = temp
                v(i,j) = vn(i,j)
            enddo
        enddo

        do i=1,N+1
            do j=1,M+1
                temp = ABS(pn(i,j)-p(i,j))/c2/dt
                if(temp.GT.errp) errp = temp
                p(i,j) = pn(i,j)
            enddo
        enddo

        error = MAX(erru,(MAX(errv,errp)))
        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 check


!!! compute velocity components u, v and pressure p
        subroutine caluvp(N,M,u,v,p,uc,vc,pc)
        implicit none
        integer :: N, M, i, j
        real(8) :: u(N,M+1), v(N+1,M), p(N+1,M+1), uc(N,M), vc(N,M), pc(N,M)

        do i=1,N
            do j=1,M
                uc(i,j) = 0.5d0*(u(i,j)+u(i,j+1))
                vc(i,j) = 0.5d0*(v(i,j)+v(i+1,j))
                pc(i,j) = 0.25d0*(p(i,j)+p(i+1,j)+p(i,j+1)+p(i+1,j+1))
            enddo
        enddo

        return
        end subroutine caluvp


!!! compute Streamfunction
        subroutine calpsi(N,M,dx,dy,u,v,psi)
        implicit none
        integer :: N, M, i, j
        real(8) :: dx, dy
        real(8) :: u(N,M), v(N,M), psi(N,M)

!       do j=1,M
!           psi(1,j) = 0.0d0
!           psi(N,j) = 0.0d0
!       enddo
!       do i=1,N
!           psi(i,1) = 0.0d0
!                   psi(i,M) = 0.0d0
!               enddo

        do i=3,N-2
            do j=2,M-3
            psi(i,j+1) = u(i,j)*2.0d0*dy+psi(i,j-1)
            !psi(i+1,j) = -v(i-1,j)*2.0d0*dx+psi(i-1,j) ! Alternative and equivalent psi formulae
            enddo
        enddo

        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)-2.0d0*dy)
        enddo


        return
        end subroutine calpsi

!!! output data file
        subroutine output(N,M,X,Y,uc,vc,psi,k)
        implicit none
        integer :: N, M, i, j, k
        real(8) :: X(N), Y(M), uc(N,M), vc(N,M), psi(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), uc(i,j), vc(i,j), psi(i,j)
```

```
          enddo
      enddo

100    format(2x,10(e12.6,'        '))
101    format('Title="Lid Driven Cavity Flow(Artificial Compressibility Methods)"')
102    format('Variables=x,y,u,v,psi')
103    format('zone',1x,'i=',1x,i5,2x,'j=',1x,i5,1x,'f=point')

      close(02)

      return
      end subroutine output
```