

**Worcester Polytechnic Institute**  
**Mechanical Engineering Department**  
**ME 612 Computational Fluid Dynamics**  
Project 4. Due May 3, 2010

## **Computing flow by the MAC Method**

Modify the program described in class in the three ways described below.

1. The advection terms in the attached program are discretized using the so-called conservative form. Change the discretization to the non-conservative (convective) form.
2. Add an inflow with a given velocity of  $U=1$  to the lower half of the left boundary. An outflow velocity of  $U=1$ , using exactly the same number of grid points should be added to the left hand side of the top boundary.
3. Add the advection and diffusion of a scalar function, using the fluid velocities. Set the diffusion of the scalar equal to the kinematic viscosity of the fluid. Specify the scalar as one at the inlet and set the normal gradient to zero everywhere else. You can store the scalar at the pressure points.

Take the density of the fluid to be 1 and the viscosity to be 0.1. Start by using 16 by 16 grid points. Run the program to steady state. Check the accuracy by doubling the resolution.

You should hand in a discussion of what you have done and the tests that you conducted to demonstrate the accuracy of your solution. Your report should include a printout of your code and plots of the steady-state solution (velocity vectors and pressure contours) for at least two different resolutions. You can also compute the streamfunction and the vorticity.

```

nx=16;ny=16;dt=0.005;nstep=200;mu=0.1;maxit=100;beta=1.2;h=1/nx;
u=zeros(nx+1,ny+2);v=zeros(nx+2,ny+1);p=zeros(nx+2,ny+2);
ut=zeros(nx+1,ny+2);vt=zeros(nx+2,ny+1);c=zeros(nx+2,ny+2)+0.25;
uu=zeros(nx+1,ny+1);vv=zeros(nx+1,ny+1);w=zeros(nx+1,ny+1);
c(2,3:ny)=1/3;c(nx+1,3:ny)=1/3;c(3:nx,2)=1/3;c(3:nx,ny+1)=1/3;
c(2,2)=1/2;c(2,ny+1)=1/2;c(nx+1,2)=1/2;c(nx+1,ny+1)=1/2;
un=1;us=0;ve=0;vw=0;time=0.0;

```

```

for is=1:nstep

```

```

    u(1:nx+1,1)=2*us-u(1:nx+1,2);u(1:nx+1,ny+2)=2*un-u(1:nx+1,ny+1);
    v(1,1:ny+1)=2*vw-v(2,1:ny+1);v(nx+2,1:ny+1)=2*ve-v(nx+1,1:ny+1);

```

```

    for i=2:nx,for j=2:ny+1    % temporary u-velocity
        ut(i,j)=u(i,j)+dt*(-(0.25/h)*((u(i+1,j)+u(i,j))^2-(u(i,j)+...
            u(i-1,j))^2+(u(i,j+1)+u(i,j))*(v(i+1,j)+...
            v(i,j))-(u(i,j)+u(i,j-1))*(v(i+1,j-1)+v(i,j-1))))+...
            (mu/h^2)*(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)-4*u(i,j)));
    end,end

```

```

    for i=2:nx+1,for j=2:ny    % temporary v-velocity
        vt(i,j)=v(i,j)+dt*(-(0.25/h)*((u(i,j+1)+u(i,j))*(v(i+1,j)+...
            v(i,j))-(u(i-1,j+1)+u(i-1,j))*(v(i,j)+v(i-1,j))+...
            (v(i,j+1)+v(i,j))^2-(v(i,j)+v(i,j-1))^2))+...
            (mu/h^2)*(v(i+1,j)+v(i-1,j)+v(i,j+1)+v(i,j-1)-4*v(i,j)));
    end,end

```

```

    for it=1:maxit    % solve for pressure
        for i=2:nx+1,for j=2:ny+1
            p(i,j)=beta*c(i,j)*(p(i+1,j)+p(i-1,j)+p(i,j+1)+p(i,j-1)-...
                (h/dt)*(ut(i,j)-ut(i-1,j)+vt(i,j)-vt(i,j-1)))+(1-beta)*p(i,j);
        end,end
    end

```

```

        % correct the velocity

```

```

    u(2:nx,2:ny+1)=...
        ut(2:nx,2:ny+1)-(dt/h)*(p(3:nx+1,2:ny+1)-p(2:nx,2:ny+1));
    v(2:nx+1,2:ny)=...
        vt(2:nx+1,2:ny)-(dt/h)*(p(2:nx+1,3:ny+1)-p(2:nx+1,2:ny));

```

```

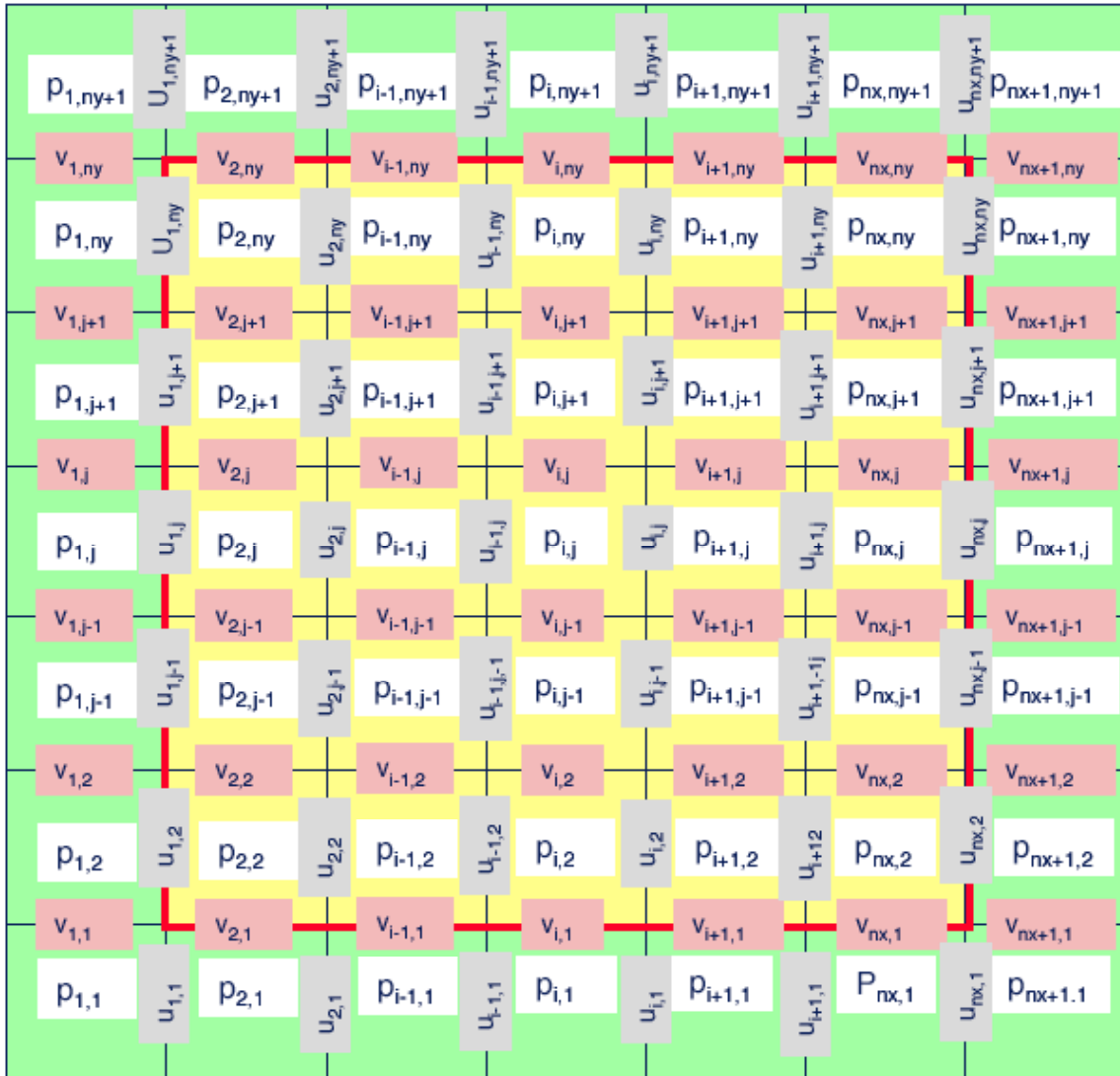
    time=time+dt    % plot the results
    uu(1:nx+1,1:ny+1)=0.5*(u(1:nx+1,2:ny+2)+u(1:nx+1,1:ny+1));
    vv(1:nx+1,1:ny+1)=0.5*(v(2:nx+2,1:ny+1)+v(1:nx+1,1:ny+1));
    w(1:nx+1,1:ny+1)=(u(1:nx+1,2:ny+2)-u(1:nx+1,1:ny+1)-...
        v(2:nx+2,1:ny+1)+v(1:nx+1,1:ny+1))/(2*h);
    hold off,quiver(flipud(rot90(uu)),flipud(rot90(vv)),'r');
    hold on;contour(flipud(rot90(w)),20),axis equal,pause(0.01)

```

```

end

```



The grid. The pressures in the ghost cells are set to zero.

The general program layout is:

Initialize parameters and arrays, set time step

for is=1:nstep

set boundary conditions for  
tangential velocity (ghost points)

Find predicted velocity (ut)

Solve for pressure using SOR

Find the projected velocity (u) by adding  
the pressure gradient

PLOT

end

The predicted velocities are found by:

$$ut(i,j)=u(i,j)+dt*(-(0.25/h)*((u(i+1,j)+u(i,j))^2-(u(i,j)+... \\ u(i-1,j))^2+(u(i,j+1)+u(i,j))*(v(i+1,j)+... \\ v(i,j))-(u(i,j)+u(i,j-1))*(v(i+1,j-1)+v(i,j-1))))+... \\ (\mu/h^2)*(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1))-4*u(i,j));$$
$$vt(i,j)=v(i,j)+dt*(-(0.25/h)*((u(i,j+1)+u(i,j))*(v(i+1,j)+... \\ v(i,j))-(u(i-1,j+1)+u(i-1,j))*(v(i,j)+v(i-1,j)))+... \\ (v(i,j+1)+v(i,j))^2-(v(i,j)+v(i,j-1))^2)+... \\ (\mu/h^2)*(v(i+1,j)+v(i-1,j)+v(i,j+1)+v(i,j-1))-4*v(i,j));$$

The pressure iteration is:

$$p(i,j)=beta*c(i,j)*(p(i+1,j)+p(i-1,j)+p(i,j+1)+p(i,j-1))-... \\ (h/dt)*(ut(i,j)-ut(i-1,j)+vt(i,j)-vt(i,j-1)))+(1-beta)*p(i,j);$$

where:

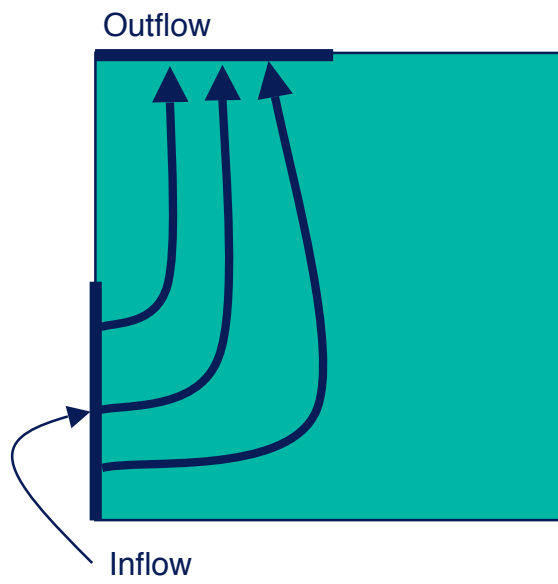
$c(i,j)=1/4$  for interior nodes;  $c(i,j)=1/3$  for boundary nodes;  
 $c(i,j)=1/2$  for corner nodes; and  $p(i,j)=0$  for ghost points

To plot you first have to find the velocities at a common node:

```
uu(1:nx,1:ny)=0.5*(u(1:nx,2:ny+1)+u(1:nx,1:ny));  
vv(1:nx,1:ny)=0.5*(v(2:nx+1,1:ny)+v(1:nx,1:ny));  
quiver(flipud(rot90(uu)),flipud(rot90(vv)),'r');
```

If you want to compute vorticity and plot it with the velocity:

```
w(1:nx,1:ny)=(u(1:nx,2:ny+1)-u(1:nx,1:ny)-... \\ (2:nx+1,1:ny)+v(1:nx,1:ny))/(2*h);  
hold off, quiver(flipud(rot90(uu)),flipud(rot90(vv)),'r');  
hold on; contour(flipud(rot90(w)),20, axis equal,pause(0.01))
```



### Conservative form

$$\frac{\partial u^2}{\partial x} + \frac{\partial vu}{\partial y}$$
$$\frac{\partial vu}{\partial x} + \frac{\partial v^2}{\partial y}$$

### Convective form

$$u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y}$$
$$u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y}$$