```
%   ------------------------------------------------------------------------
%   Robert Steven Sellers
%   Computer Project #4, Computational Fluid Dynamics MAE 5440, Spring 2010
%   Due Wed. April 21, 2010
%
%   A) Driven cavity solver based on staggered grid Navier-Stokes solution
%   B) Plane parallel surface solver
%   ------------------------------------------------------------------------

clear all;
format short
%clc
node = 51;      %   Nodes along plate, total nodes = n*n
dx = 1/node;
dy = dx;

omega = .5;     %   under relaxation
mu = .01;       %   Viscosity constant

%dy = 1/nodex; %for part 2

u = zeros(node+1,node+1);
v = zeros(node+1,node+1);
u(:,node+1)=1;
ustar = u;
%u(2:nodex,1:node)=.0001; %part 2
centerline = zeros(node+1,1);

%v(node+2,:)=1;  %these are u-velo BS's to test if v code is correct
vstar = v;
p = zeros(node+1,node+1);
pstar = p;
pprime = zeros(node+1,node+1);
S = zeros(node,node);

mdote = 0;
mdotw = 0;
mdotn = 0;
mdots = 0;

AuP = zeros(node+1,node+1);
AuE = zeros(node+1,node+1);
AuW = zeros(node+1,node+1);
AuN = zeros(node+1,node+1);
AuS = zeros(node+1,node+1);

AvP = zeros(node+1,node+1);
AvE = zeros(node+1,node+1);
AvW = zeros(node+1,node+1);
AvN = zeros(node+1,node+1);
AvS = zeros(node+1,node+1);

ap = zeros(node+1,node+1);

ae = zeros(node+1,node+1);
aw = zeros(node+1,node+1);
an = zeros(node+1,node+1);
as = zeros(node+1,node+1);

for iter = 1:50000;
fprintf('Grand iteration: %i\n',iter);
```

```
%   Populate AuP matrix
ustar = u;
vstar = v;
pstar = p;
for i=2:node
    for j = 1:node
        %   Assign mass flowrate values for clarity
        mdote = .5*(ustar(i,j)+ustar(i+1,j))*dy; %.5*(uP+uE)
        mdotw = .5*(ustar(i,j)+ustar(i-1,j))*dy; %.5*(uP+uW)
        mdotn = .5*(vstar(i-1,j+1)+vstar(i,j+1))*dx; %**GOOD
        mdots = .5*(vstar(i-1,j)+vstar(i,j))*dx; %**GOOD

        AuE(i,j) = max(-mdote,0)+(mu*dy)/dx;
        AuN(i,j) = max(-mdotn,0)+(mu*dx)/dy;
        AuW(i,j) = max(mdotw,0)+(mu*dy)/dx;
        AuS(i,j) = max(mdots,0)+(mu*dx)/dy;

        %   Top and bottom faces are the only ones that need attention
        if (j==node)
            %   Top face
            %   Reassign AuN to include BC
            AuN(i,j) = max(-mdotn,0)+(mu*dx)/(.5*dy);
        elseif (j==1)
            %   Bottom face
            %   Reassign AuS to include BC
            AuS(i,j) = max(mdots,0)+(mu*dx)/(.5*dy);
        end
            AuP(i,j) = (AuE(i,j)+AuN(i,j)+AuW(i,j)+AuS(i,j))/omega;
     end
end

%   Populate AvP matrix
for i=1:node
    for j = 2:node

        mdote = .5*(ustar(i+1,j)+ustar(i+1,j-1))*dy; %*GOOD
        mdotw = .5*(ustar(i,j)+ustar(i,j-1))*dy; %**GOOD
        mdotn = .5*(vstar(i,j)+vstar(i,j+1))*dx;
        mdots = .5*(vstar(i,j)+vstar(i,j-1))*dx;

        AvE(i,j) = max(-mdote,0)+(mu*dy)/dx;
        AvN(i,j) = max(-mdotn,0)+(mu*dx)/dy;
        AvW(i,j) = max(mdotw,0)+(mu*dy)/dx;
        AvS(i,j) = max(mdots,0)+(mu*dx)/dy;

        if (i==node)
            %   Right wall
            AvE(i,j) = max(-mdote,0)+(mu*dy)/(.5*dx);
        elseif (i==1)
            %   Left wall
            AvW(i,j) = max(mdotw,0)+(mu*dy)/(.5*dx);
        end
            AvP(i,j) = (AvE(i,j)+AvN(i,j)+AvW(i,j)+AvS(i,j))/omega;
     end
end


for k = 1:30
%   Solve u-momentum equation
for i=2:node
    for j = 1:node
```

```
        uP=ustar(i,j);
        uW=ustar(i-1,j);
        uE=ustar(i+1,j);
        uN=ustar(i,j+1);

        if (j==1) uS=0; else uS=ustar(i,j-1); end

        %if (i==nodex) %PART 2
            %apply dU/dx=0 boundary condition
        %    ustar(nodex+1,j) = ustar(nodex,j);
        %end

        ustar(i,j) = (1-omega)*u(i,j)+1/AuP(i,j)* ...
            (AuE(i,j)*uE+AuW(i,j)*uW+AuN(i,j)*uN+AuS(i,j)*uS + ...
            (pstar(i-1,j)-pstar(i,j))*dy);

    end
end

%for j=1:node %Part 2
%         mdotin = .5*(ustar(nodex-2,j)+ustar(nodex-1,j));
%         mdotout = .5*(ustar(nodex,j)+ustar(nodex+1,j));
%         ustar(nodex,j) = (mdotin/mdotout) * ustar(nodex,j);
%end


%   Solve v-momentum equation
for i=1:node
    for j = 2:node
        vP=vstar(i,j);
        if (i==1) vW=0; else vW=vstar(i-1,j); end
        vE=vstar(i+1,j);
        vN=vstar(i,j+1);
        vS=vstar(i,j-1);

        vstar(i,j) = (1-omega)*v(i,j) + 1/(AvP(i,j))* ...
            (AvE(i,j)*vE+AvW(i,j)*vW+AvN(i,j)*vN+AvS(i,j)*vS + ...
            (pstar(i,j-1)-pstar(i,j))*dx);

    end
end
end

%   Populate little "a" values for the pressure solver
for i = 1:node
    for j = 1:node
         if (i==node)
             %   Right face
             ae(i,j) =0;
         else
             ae(i,j) = (dy^2)/(AuP(i+1,j));
         end
         if (i==1)
             %   Left face
             aw(i,j) = 0;
         else
             aw(i,j) = (dy^2)/(AuP(i,j));
         end

         if (j==node)
             %   Top face
```

```
            an(i,j) = 0;
        else
            an(i,j) = (dx^2)/(AvP(i,j+1));
        end
        if (j==1)
            %   Bottom face
            as(i,j) = 0;
        else
            as(i,j) = (dx^2)/(AvP(i,j));
        end
    end
end

%   Find source term and ap terms
for i = 1:node
    for j = 1:node
        %S(i,j) = (ue(i,j) - uw(i,j))*dy + (vn(i,j) - vs(i,j))*dx;
        S(i,j) = (ustar(i+1,j) - ustar(i,j))*dy + (vstar(i,j+1) - vstar(i,j))*dx;
        ap(i,j) = ae(i,j) + aw(i,j) + an(i,j) + as(i,j);
    end
end

%   Initialize the pprime matrix and solve it
pprime = zeros(node+1, node+1);
for k = 1:100
    for i = 1:node
        for j = 1:node
            if (i==1) pW=0; else pW=pprime(i-1,j); end
            if (j==1) pS=0; else pS=pprime(i,j-1); end
            pprime(i,j) = pprime(i,j) + (1.6/ap(i,j)) * (ae(i,j)*pprime(i+1,j) +
aw(i,j)*pW + an(i,j)*pprime(i,j+1)+ as(i,j)*pS - S(i,j) - ap(i,j)*pprime(i,j));
        end
    end
end

%   output source to guage continuity
fprintf('Source before correction: %g\n', sqrt(sum(sum(S.*S))));
before = sqrt(sum(sum(S.*S)));

%   Correct pressure (under relax)
for i = 1:node
    for j = 1:node
        p(i,j) = pstar(i,j) + omega*pprime(i,j);
    end
end
%   Correct u-momentum to satisfy continuity
for i=2:node
    for j = 1:node
        u(i,j) = ustar(i,j) + (dy/AuP(i,j))*(pprime(i-1,j)-pprime(i,j));
    end
end
%   Correct v-momentum to satisfy continuity
for i=1:node
    for j = 2:node
        v(i,j) = vstar(i,j) + (dx/AvP(i,j))*(pprime(i,j-1)-pprime(i,j));
    end
end
%   Find source AFTER velocities have been corrected
for i = 1:node
    for j = 1:node
        S(i,j) = (u(i+1,j) - u(i,j))*dy + (v(i,j+1) - v(i,j))*dx;
    end
```

```
end

after = sqrt(sum(sum(S.*S)));
fprintf('After pressure correction: %g\n\n', after);
%break
if (before < 1e-10)
    fprintf('\n\nSource before is below 1e-10, you rock!\n\n');
    break;
end
end

%   Create centerline plot
for i = 1:node;
    centerline(i) = u((node+1)/2,i);
end

%   Calculate drag per pg. 278 second ordr central difference
drag = 0;
for i = 1:node-1
    drag = drag + -mu*(dx*(u(i,node)-1))/(.5*dy);
end
drag

%   Interpolate u & v values so they lie on the same node for the quiver
%   function (which creates velocity vectors)
uquiv = zeros(1:node, 1:node);
vquiv = zeros(1:node, 1:node);
for i = 1:node
    for j = 1:node
    uquiv(i,j) = (u(i,j)+u(i+1,j))/2;
    vquiv(i,j) = (v(i,j)+v(i,j+1))/2;
    end
end
%Output centerline
 for i = 1:node
         fprintf('%f \t %f\n', i*dx, centerline(i));
 end
```