```matlab
% Shivam_Swarnakar_184106011_ME415_Term Project
% MATLAB code for Semi-Explicit method-based 2D unsteady Navier Stokes
 solver for a staggered grid
clc;
clear;

%Inputs

 L1 = 1;
 L2 = 1;
 imax = 42;
 jmax = 42;
 Re = 100;
 U_0 = 1;
 rho = 1;
 mu = 1/Re;

%Grid Generation _ Uniform Grid

Dx = L1/(imax-2);    Dy = L2/(jmax-2);

Del_x = 0:Dx:L1;      Del_y = 0:Dy:L2; %xi & yj

xc(2:imax-1) = ( Del_x(2:imax-1)+Del_x(1:imax-2) )/2;

xc(1) = Del_x(1);  xc(imax) = Del_x(imax-1);

Dx(2:imax-1) = Del_x(2:imax-1)-Del_x(1:imax-2); %width of CV

Dx(1)=0;   Dx(imax)=0;

del_x(1:imax-1) = xc(2:imax) - xc(1:imax-1); %Distance between Cell
 Centers

yc(2:jmax-1) =  ( Del_y(2:jmax-1)+Del_y(1:jmax-2) )/2;

yc(1)= Del_y(1);  yc(jmax)= Del_y(jmax-1);

Dy(2:jmax-1)= Del_y(2:jmax-1)-Del_y(1:jmax-2);

Dy(1)=0; Dy(jmax)=0;

del_y(1:jmax-1) = yc(2:jmax)-yc(1:jmax-1);


%BC

%Top wall

   u(1:imax-1,jmax) = 1;
   v(1:imax,jmax-1) = 0;
```

```
%Bottom

   u(1:imax-1,1) = 0;
   v(1:imax,1) = 0;

%Left

   u(1,1:jmax) = 0;
   v(1,1:jmax-1) = 0;

%Right

    u(imax-1,1:jmax) = 0;
    v(imax,1:jmax-1) = 0;

%IC
    u(2:imax-2,2:jmax-1) = 0;
    v(2:imax-1,2:jmax-2) = 0;
    p(1:imax,1:jmax) = 0;

%Steady State Convergance tolerance

    E_st=0.0001; % U & V -velocity Steady State

    e=10^-8; % Mass-conservation

%Set delt

   dt_adv = (abs(U_0)/Dx(2))^-1;

   dt_diff = 0.5*(rho/mu)*((1/Dx(2)^2)+(1/Dy(2)^2))^-1;

   delt = 0.05 * (min(dt_adv,dt_diff));

   Del_tau = U_0*delt/L1; %Non-Dimensional

   Unsteadiness=1;

   n=0;

while Unsteadiness > E_st

           n = n+1;

           %Pressure BC

           %Top

           p(1:imax,jmax) = p(1:imax,jmax-1);

           %Bottom

           p(1:imax,1) = p(1:imax,2);
```

```
            %Left

            p(1,1:jmax) = p(2,1:jmax);

            %Right

            p(imax,1:jmax) = p(imax-1,1:jmax);

            u_old = u;

            v_old = v;

            p_old = p;

            u_star = u;

            v_star = v;

            p_star = p;

  %Predictor Step of U-Velocity

       %Adevection Scheme is FOU

            %Fluxes in x-dirrection

           for i=1:imax-2
             for j=2:jmax-1

                 mx1(i,j) = (rho/2) *( u(i,j) + u(i+1,j) );

                 ax1(i,j)= ( max(mx1(i,j),0 ) * u(i,j)) +
( min(mx1(i,j),0) * u(i+1,j) );

                 dx1(i,j)= mu * ( u(i+1,j) - u(i,j) )/ Dx(i+1);

              end
           end

          %Fluxes in y-direction


           for i = 2 : imax-2
             for j= 1 : jmax-1

                 my1(i,j) = ( rho/2) *( v(i,j) + v(i+1,j) );

                 ay1(i,j)= ( max(my1(i,j),0) * u(i,j) ) +
( min(my1(i,j),0) * u(i+1,j) );

                 d_uy(i,j)= ( mu* (u(i,j+1) - u(i,j) )/ del_y(j));


               end
```

```
        end


%Temppral Evolation of U-Velocity

      for i = 2 : imax-2
          for j = 2 : jmax-1

             Au(i,j) = ( ax1(i,j) - ax1(i-1,j) )* del_y(j) +
( ay1(i,j) - ay1(i,j-1) )* Dx(i);

             Du(i,j) = ( dx1(i,j) - dx1(i-1,j) )* Dy(j) + ( d_uy(i,j)
- d_uy(i,j-1) )* del_x(i);

             Su(i,j) = ( p_star(i,j) - p_star(i+1,j) ) *Dy(j);

             u_star(i,j) = u(i,j) + ( (delt/(rho*del_x(i)*Dy(j)) ) *
( Du(i,j) - Au(i,j) + Su(i,j)) );

             mx_star(i,j) =  rho*u_star(i,j);

             end
      end
           % mass flux prediction at Left boundary

            mx_star(1,1:jmax) = rho * u_star(1,1:jmax);

          % mass flux prediction at Right boundary

           mx_star(imax-1,1:jmax) = rho * u_star(imax-1,1:jmax);


%Predictor Step of V-Velocity

   %Adevection Scheme is FOU

       %Fluxes in x-dirrection

       for j=2:jmax-2
          for i=1:imax-1

            mx2(i,j) = (rho/2) *( u(i,j) + u(i,j+1));

            ax2(i,j) = ( max(mx2(i,j),0) * v(i,j)) +
( min(mx2(i,j),0 ) * v(i,j+1));

            dx2(i,j) = (mu * ( v(i+1,j) - v(i,j) ) /del_x(i));


             end
          end

    %Fluxes in y-direction
```

```
        for i = 2 : imax-1
           for j= 1 : jmax-2

                my2(i,j) = (rho/2) * ( v(i,j+1) + v(i,j) );

                ay2(i,j)= ( max(my2(i,j),0) * v(i,j)) +
 ( min(my2(i,j),0) * v(i,j+1));

                dy2(i,j)= mu* ( v(i,j+1)- v(i,j) )/ Dy(j+1);

              end
           end

%Temppral Evolation of V-Velocity

    for i= 2 : imax-1
        for j = 2 : jmax-2

                Av(i,j) = ( ax2(i,j)- ax2(i-1,j) ) * del_y(j) +
 ( ay2(i,j)-ay2(i,j-1) )* Dx(i);

                Dv(i,j) = ( dx2(i,j) - dx2(i-1,j) )*del_y(j) +
 ( dy2(i,j) - dy2(i,j-1) )* Dx(i);

                Sv(i,j) = ( p_star(i,j) - p_star(i,j+1) )* Dx(i);

                v_star(i,j) = v(i,j) + ( (delt/(rho*Dx(i)*del_y(j)) )*
 ( Dv(i,j) -Av(i,j) + Sv(i,j) ) );

                my_star(i,j)= rho* v_star(i,j);

              end
    end

%mass flux prediction at Bottom boundary

              my_star(1:imax,1)= rho* v_star(1:imax,1);

%my* - mass flux prediction at Top boundary

              my_star(1:imax,jmax-1) = rho* v_star(1:imax,jmax-1);



    %Mass Imbalance

    %Mass_source prediction

    for i = 2 : imax-1
        for j = 2 : jmax-1

    S_m_star(i,j)= ( mx_star(i,j) - mx_star(i-1,j) )* Dy(j) +
 ( my_star(i,j)- my_star(i,j-1) )* Dx(i);
```

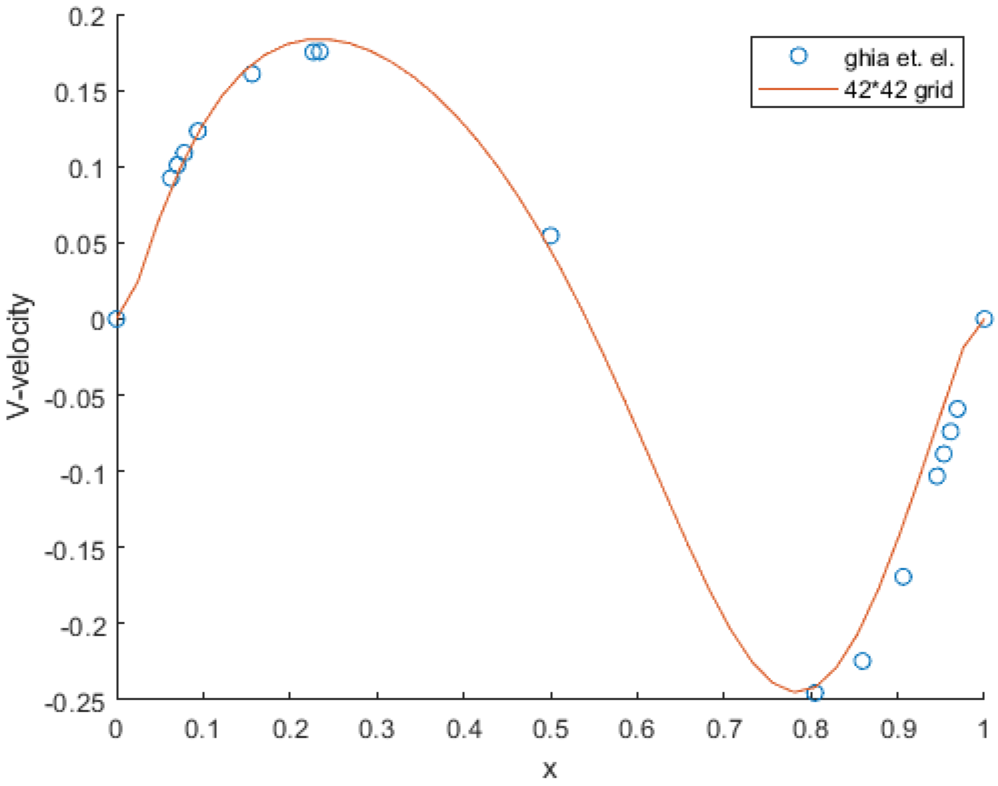
```
            end
         end

%Corrector Step


        p_c = zeros(imax,jmax);

%Gauss Seidal Iteration Loop for Pressure

    N=0;

   while ( max(abs(S_m_star),[],"all") > e)

            N = N+1;

            %BC for pressure correction

            %Top

            p_c(1:imax,jmax) = p_c(1:imax,jmax-1);

            %Bottom

            p_c(1:imax,1) = p_c(1:imax,2);

            %Left

            p_c(1,1:jmax) = p_c(2,1:jmax);

            %Right

            p_c(imax,1:jmax) = p_c(imax-1,1:jmax);


            p_c_old = p_c;

%Pressure Correction for Internal grid points

        for  i=2:imax-1
                for  j=2:jmax-1

                     S_m_corr(i,j)=-( delt*Dy(j) )*(( (p_c_old(i
+1,j) - p_c_old(i,j) )/ del_x(i) )-( (p_c_old(i,j)- p_c(i-1,j) ) )/
del_x(i-1) )- ( delt*Dx(i) ) * ( ( ( p_c_old(i,j+1) - p_c_old(i,j) ) /
del_y(j) )- ( ( p_c_old(i,j) - p_c(i,j-1) )/del_y(j-1) ) );

                     ap(i,j) = delt*( ( (1/del_x(i))+(1/
del_x(i-1) ) )*Dy(i) +( (1/del_y(j)) + (1/del_y(j-1) ) )* Dx(i));

                     p_c(i,j) = p_c_old(i,j) - (1/ap(i,j)) *
 ( S_m_star(i,j)+S_m_corr(i,j) );

                end
```

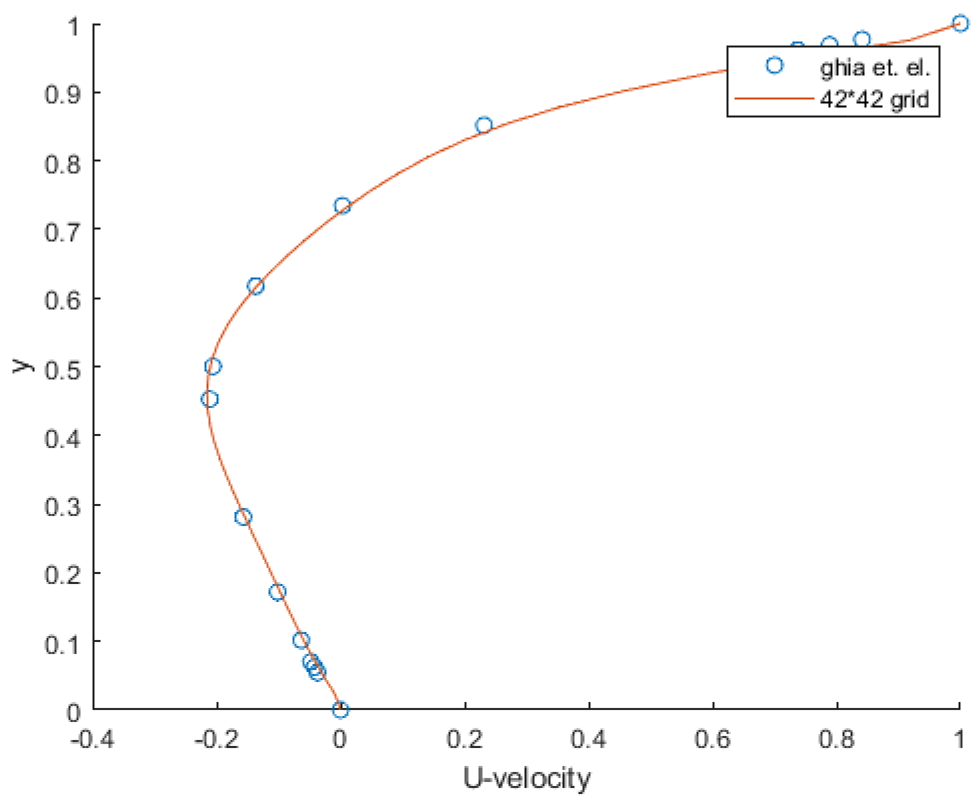
```
        end

    %Updating the values of mass flux correction, predicted mass flux and
     predicted velocity

            for i=1:imax-1
                for j=2:jmax-1

                    %mass_x flux correction

                    mx_c(i,j) = -(delt*( p_c(i+1,j)- p_c(i,j) ) /
     del_x(i) );

                   %Predicted mass flux_x

                    mx_star(i,j) = mx_star(i,j) + mx_c(i,j);

                   %Predicted U-velocity

                    u_star(i,j) = mx_star(i,j) /rho;
                end
            end


            for   i=2:imax-1
                for j=1:jmax-1

                  %mass_y flux correction

                   my_c(i,j) = -( delt )*( p_c(i,j+1) - p_c(i,j) )/
    del_y(j);

                  %Predicted mass flux_y

                   my_star(i,j)= my_star(i,j) + my_c(i,j);

                   %Predicted V-velocity

                   v_star(i,j) = my_star(i,j) /rho;


                end
            end


            for  i=2:imax-1
                for j=2:jmax-1

                    S_m_star(i,j)= ( mx_star(i,j)-
     mx_star(i-1,j) )*Dy(j) + ( my_star(i,j)-my_star(i,j-1) )*Dx(i);

                    p_star(i,j) = p_star(i,j) + p_c(i,j);

                end
```

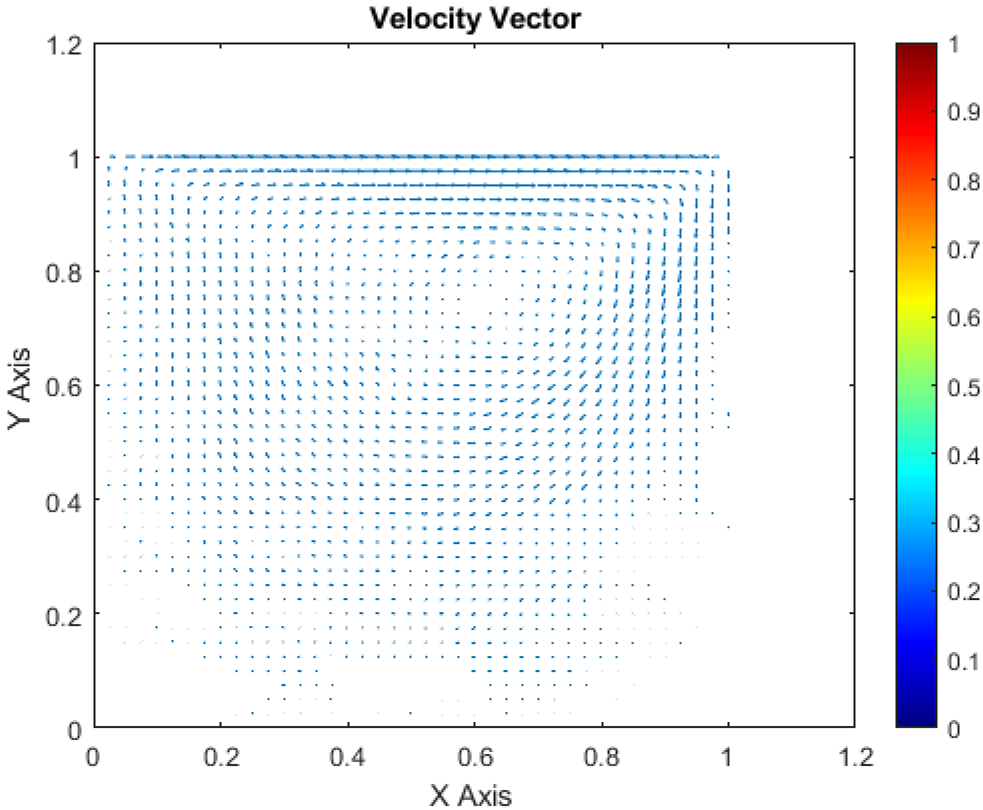
```
            end

      %End of Gauss Seidal loop

    end


%Next time step

      p = p_star;
      u = u_star;
      v = v_star;

%Updating unsteadiness value

         Unsteadiness=max( max(abs((u-u_old)/Del_tau),[],"all" ),
 max(abs((v-v_old) /Del_tau ),[],"all") );
end

xv = [1.0000 0.0000; 0.9688 -0.05906;0.9609 -0.07391;
    0.9531 -0.08864;
    0.9453  -0.10313;
 0.9063 -0.16914;
 0.8594 -0.22445;
 0.8047 -0.24533;
 0.5 0.05454;
 0.2344 0.17527;
 0.2266 0.17507;
 0.1563 0.16077;
 0.0938 0.12317;
 0.0781 0.1089;
 0.0703 0.10091;
 0.0625 0.09233;
 0.0000 0.00000;];

yu = [1        1.0000;
 0.9766   0.84123;
 0.9688   0.78871;
 0.9609   0.73722;
 0.9531   0.68717;
 0.8516   0.23151;
 0.7344   0.00332;
 0.6172  -0.136641;
 0.5     -0.20581;
 0.4531  -0.2109;
 0.2813  -0.15662;
 0.1719  -0.1015;
 0.1016  -0.063434;
 0.0703  -0.04775;
 0.0625  -0.04192;
 0.0547  -0.03717;
 0.0000   0.00000; ];
```

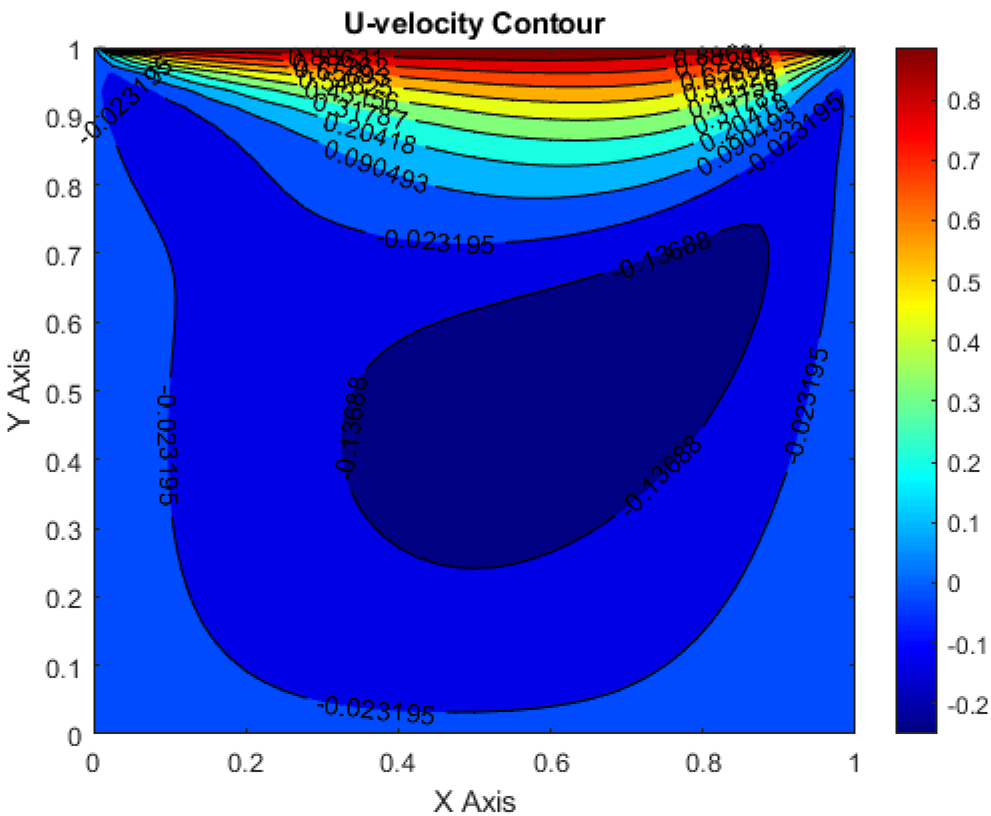
```
xv_(1:17) = xv(17:-1:1,2);
yu_(1:17) = yu(17:-1:1,2);


x(1:17) = xv(17:-1:1,1);
scatter(x,xv_(:))
hold on
y = linspace(0,1,imax);
plot(y,v(:,(imax-2)/2+1))
legend('ghia et. el.','42*42 grid')
xlabel('x')
ylabel('V-velocity')
hold off
figure (1)

figure(2)
x = yu(17:-1:1,1);
scatter(yu_(:),x)
hold on
y = linspace(0,1,imax);
plot(u((imax-2)/2+1,:),y)
xlabel('U-velocity')
ylabel('y')
legend('ghia et. el.','42*42 grid')
hold off




figure(3)
x1=Del_x(2:imax-1);
y1=Del_y(2:jmax-1);
u1=u(2:imax-1,2:jmax-1);
v1=v(2:imax-1,2:jmax-1);
[X,Y] = meshgrid(x1,y1);
quiver(X,Y,u1',v1')
title('Velocity Vector')
xlabel('X Axis ')
ylabel('Y Axis ')
colorbar
colormap(jet)


figure(4)
[X,Y] = meshgrid(Del_x,yc);
[c,h] = contourf(X,Y,u',10);
clabel(c,h);
colorbar
colormap(jet)
title('U-velocity Contour')
xlabel('X Axis')
ylabel('Y Axis')

figure(5)
```

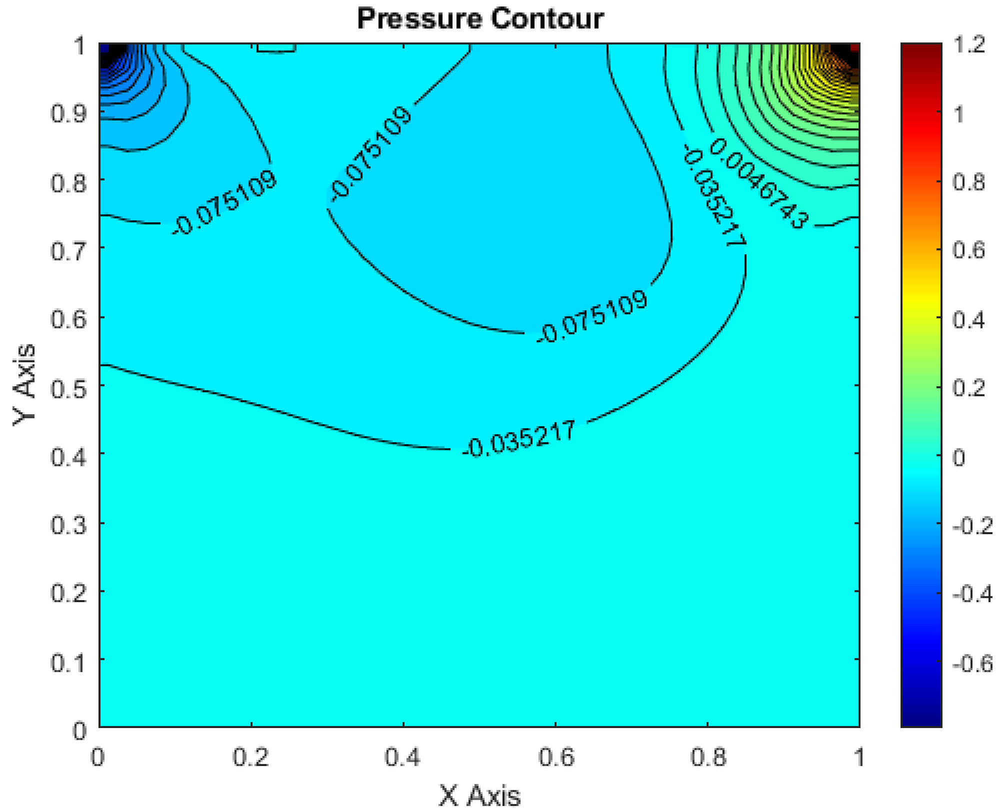
```
[X,Y] = meshgrid(xc,yc);
[c,h] = contourf(X,Y,p',50);
clabel(c,h);
colorbar
colormap(jet)
title('Pressure Contour')
xlabel('X Axis')
ylabel('Y Axis')
```




ghia et. el.
42*42 grid
y
U-velocity
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Velocity Vector
Y Axis
X Axis
1.2
1
0.8
0.6
0.4
0.2
0
0.9
0.7
0.5
0.3
0.1


U-velocity Contour
Y Axis
X Axis
-0.023195
0.090493
0.20418
0.31787
-0.13688
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2



Published with MATLAB® R2020b