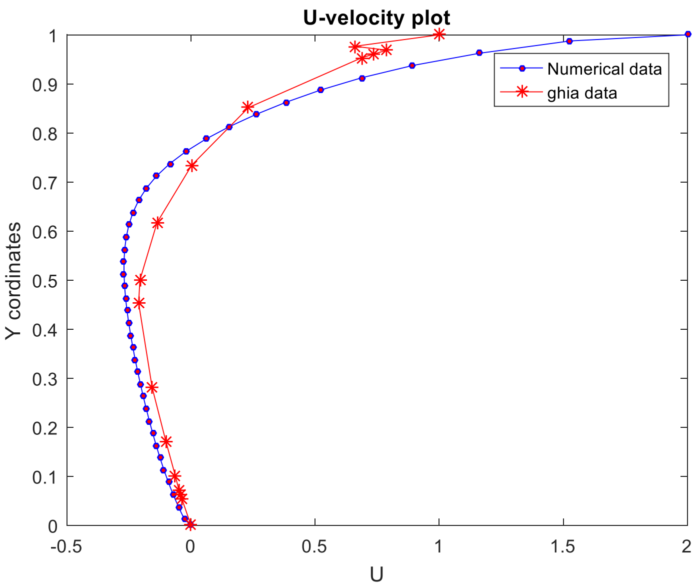
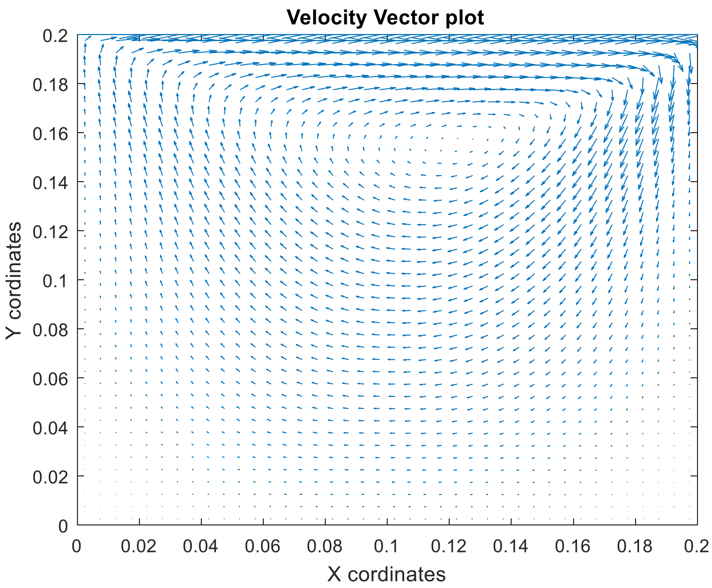
5/5/2018

ASSIGNMENT 4

COMPUTATIONAL FLUID DYNAMICS AND HEAT TRANSFER





173109003 | Sanit Prashant Bhatkar

Computational Fluid Dynamics and Heat Transfer (ME 415), Autumn 2018

Assignment 4: SIMPLE Algorithm

Name: Sanit Prashant Bhatkar

Roll No: 173109003

Problem 1:

The lid-driven flow is generally used as a benchmark for numerical methods. Consider a two-dimensional flow of an incompressible fluid in a square cavity with length (L) and width (W) 20 cm. The cavity walls at x = 0, x = L and y = 0 are stationary. The wall at y = W (the lid) is moving with a velocity (U) in the x-direction. Take kinematic viscosity (ν) as 0.004 m²/s.

Algorithm

1. **Guess a initial pressure field (P*)**

2. **Solve the imperfect velocity field, the starred velocities, based on the estimates of P***

$$a_e u_e^* = \sum a_{nb} u_{nb}^* + b + (P_P^* - P_E^*) A_e$$

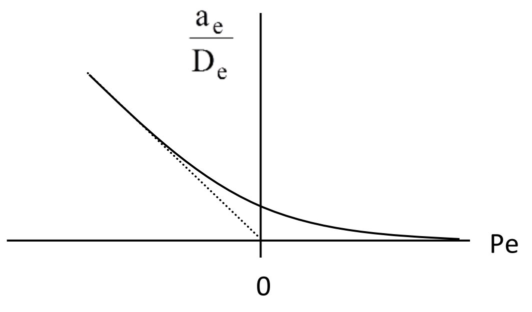
$$a_e v_n^* = \sum a_{nb} v_{nb}^* + b + (P_P^* - P_N^*) A_n$$

$$\left[J_e - F_e \phi_P\right] = a_E \left(\phi_P - \phi_E\right)$$

$$a_e = De \cdot AMAX\left[-Pe, A[Pe], 0\right]$$

$$a_w = Dw \cdot AMAX\left[Pw, A[Pw], 0\right]$$

$$Pe = \frac{Fe}{De} = \frac{\rho u_e}{\Gamma/\delta x}$$



3. **Pressure correction equation**

$$a_P P_P' = a_E P_E' + a_W P_w' + a_N P_N' + a_S P_S' + D^* \qquad \Rightarrow \qquad \nabla^2 P' = D^*$$

$$a_E^P = \rho d_e A_e \;\; ; \;\; d_e = \frac{A_e}{a_e} \;\; \& \;\; D^* = -\left\{\left[\rho u_e^* A_e - \rho u_w^* A_w\right] + \left[\rho v_n^* A_n - \rho v_s^* A_n\right]\right\}$$

$$\underbrace{u}_{\text{True}} = \underbrace{u^*}_{\text{Predicted}} + \underbrace{u'}_{\text{Corrector}}$$

$$v = v^* + v'$$

$$P = P^* + P'$$

$$\underbrace{\left[\rho u'_e A_e - \rho u'_w A_w\right] + \left[\rho v'_n A_n - \rho v'_n A_n\right]}_{\text{Predictor}} = -\underbrace{\left\{\left[\rho u^*_e A_e - \rho u^*_w A_w\right] + \left[\rho v^*_n A_n - \rho v^*_s A_n\right]\right\}}_{\text{D mass div. (Corrector)}}$$

$$\text{Predictor Eq.} \Rightarrow a_e u^*_e = \sum a_{nb} u^*_{nb} + b + (P^*_P - P^*_E) A_e$$

$$\text{Corrector Eq.} \Rightarrow a_e u'_e = \underbrace{\sum a_{nb} u'_{nb}}_{\text{neglected}} + (P'_P - P'_E) A_e$$

If you sum both equations you get the original momentum equation. Substituting the velocity corrector equation into the mass conservation equation, one gets the pressure corrector equation:

$$\left[\rho d_e \left(P'_P - P'_E\right) A_e - \rho d_w \left(P'_P - P'_W\right) A_w\right] + \left[\rho d_n \left(P'_P - P'_N\right) A_n - \rho d_s \left(P'_P - P'_S\right) A_n\right]$$
$$= -\left\{\left[\rho u^*_e A_e - \rho u^*_w A_w\right] + \left[\rho v^*_n A_n - \rho v^*_s A_n\right]\right\}$$

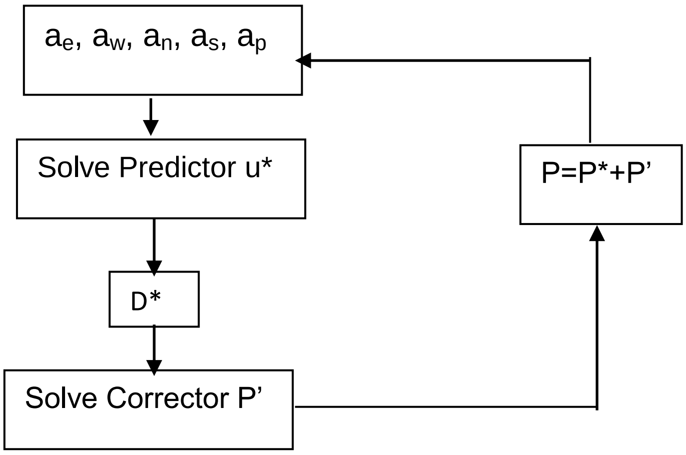
4. Update the pressure and velocities values to satisfie mass balance at each cell:

$$P^* = P^* + P'$$

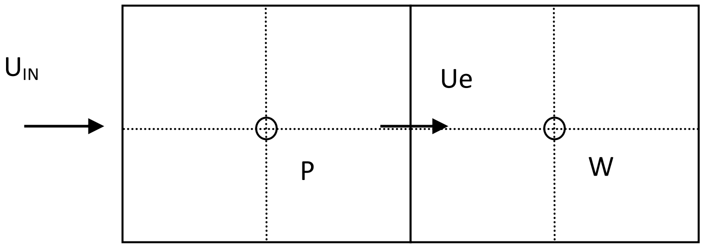
$$\underbrace{u^*_e}_{\text{Actualized}} = \underbrace{u^*_e}_{\text{Predicted}} + \underbrace{d_e\left(P'_P - P'_E\right)}_{\text{Correction}}$$

$$v^*_n = v^*_n + d_n\left(P'_P - P'_N\right)$$

5. **Return to step 1 using the updated pressure field as a new guess.**



Pressure Boundary Conditions



$$\left[\rho d_e\left(P_P' - P_E'\right)A_e - \underbrace{\rho d_w\left(P_P' - P_W'\right)A_w}_{\text{No Correction}}\right] + \left[\rho d_n\left(P_P' - P_N'\right)A_n - \rho d_s\left(P_P' - P_S'\right)A_n\right]$$

$$= -\left\{\left[\rho u_e^* A_e - \underbrace{\rho u_{IN} A_w}_{\text{No Prediction}}\right] + \left[\rho v_n^* A_n - \rho v_s^* A_n\right]\right\}$$

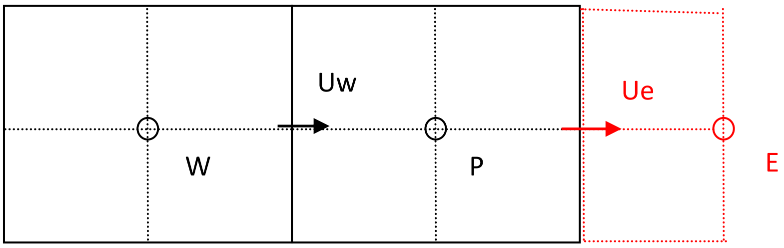
Since the U_{IN} is given, the flow rate across w face should not be expressed as a predicted field, u*, but in terms of the true values of u, i.e., u' will be zero – no correction applies. As a consequence P'_W will not appear or a_W will be zero in the P' equation.

$$a_P P_P' = a_E P_E' + a_N P_N' + a_S P_S' + D^*$$

$$D^* = -\left\{\left[\rho u_e^* A_e - \rho u_{IN} A_w\right] + \left[\rho v_n^* A_n - \rho v_s^* A_n\right]\right\}$$

OUTLET

Creating a ficticious node E and fixing the external pressure at P_E.



The the momentum predictor and corrector equations takes the form:

$$\text{Predictor Eq.} \Rightarrow a_e u_e^* = \sum a_{nb} u_{nb}^* + b + (P_P^* - P_E) A_e$$

$$\text{Corrector Eq.} \Rightarrow a_e u_e' = (P_P' - P_E') A_e$$

Substituting into the mass conservation equation:

$$\left[\rho d_e \left(P_P' - P_E'\right) A_e - \rho d_w \left(P_P' - P_W'\right) A_w\right] + \left[\rho d_n \left(P_P' - P_N'\right) A_n - \rho d_s \left(P_P' - P_S'\right) A_n\right]$$
$$= -\left\{\left[\rho u_e^* A_e - \rho u_w^* A_w\right] + \left[\rho v_n^* A_n - \rho v_s^* A_n\right]\right\}$$

Grid details and the implemented boundary condition

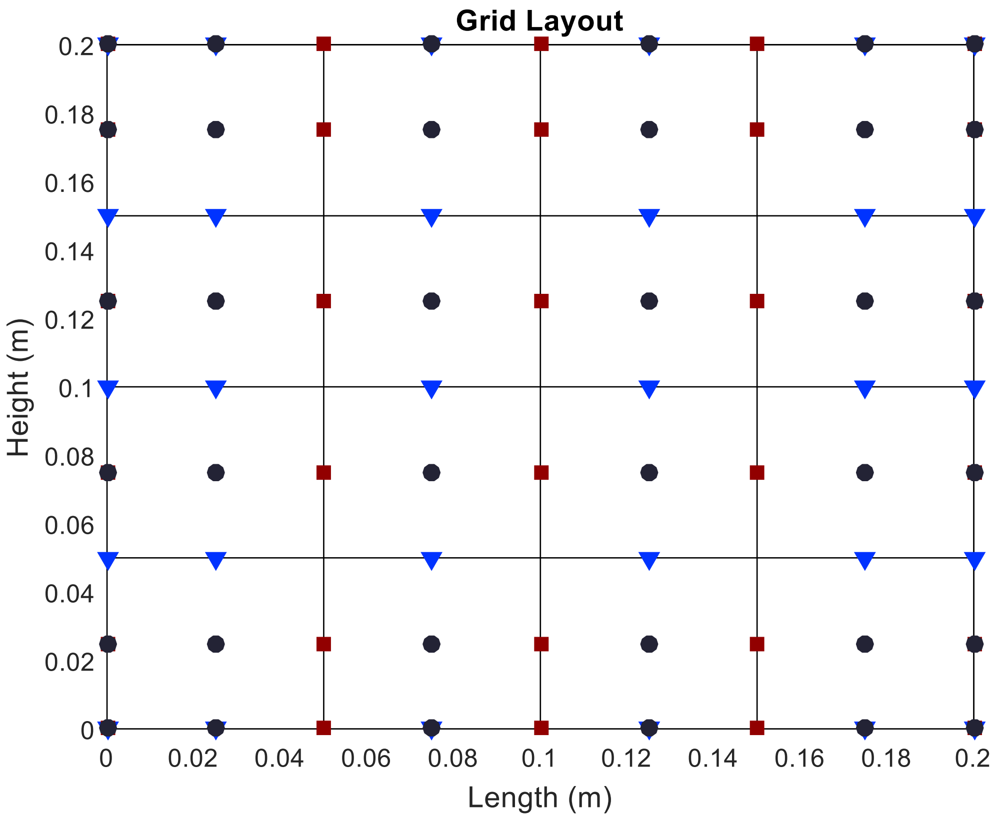


Fig 1. Generated grid for the given Cartesian geometry

Circles represent the grid points. Triangles represent y direction faces and squares represent x direction faces.

Boundary Conditions

- On the left boundary u = 0 and v = 0
- On the bottom boundary u = 0 and v = 0
- On the right boundary u = 0 and v = 0
- On the top boundary u = U and v = 0

Code of the problem

```
%============================================================%
% Assignment 4 CHDHT ME 415
% Progarm uses FVM to solve the problem of 2D grid
% Designed only for rectangular co-ordinate system
%
% AUTHOR:
% Sanit P. Bhatkar (173109003@iitb.ac.in)
% Roll No: 173109003
% Place: IIT BOMBAY.
%============================================================%

clc
clearvars
flg=0;

%% Input parameters
fprintf('Lid Driven Flow\n');
choice=input('Input choice for Richardson Method \n1.Yes\n2.No\n');

%% Richardson scheme
if choice==1
fprintf('\nEnter grid size in decreasing order');

while flg<3

    Dh=input('\nInput grid size: ');
    rho=1;
    nu=0.004;
    un=input('\nVelocity: ');
    mu=rho*nu;
    l=0.2;
    w=0.2;
    Re=un*rho*l/nu;
    Dl=Dh;
    dl=Dl;
    dh=Dl;
    sx=(l/dl)+2;
    sy=(w/dh)+2;


%% Initializing pressure and velocity

%initial pressure
p=zeros(sx,sy);

% u velocity boundary conditions and initial condition.
u=zeros(sy,sx-1);
u(1:sy,1)=0;              %left wall boundary
u(1:sy,sx-1)=0;         %right wall boundary condition
u(1,2:sx-1)=0;            %bottom boundary
u(sy,1:sx-1)=un;          %top boundary
uold=u;
ustar=u;

% v velocity boundary conditions and initial condition.
v=zeros(sy-1,sx);
vstar=v;
vold=v;

%time step
```

```
Dt_a=Dh/abs(max(max(u)));
Dt_d=2*mu/rho*(1/Dh^2+1/Dl^2);
Dt_d=1/Dt_d;
dt=min(Dt_d,Dt_a);
dt=0.5*dt;
itr=0;

%% SIMPLE method

% main while
  epsi=1;

  while epsi>1e-3


%% U control volume fluxes

 %fluxes in x drection
    for i=1:sx-2
       for j=2:sy-1
           mxu(j,i)=rho*(u(j,i)+u(j,i+1))/2;
           axu(j,i)=max(mxu(j,i),0)*u(j,i)+min(mxu(j,i),0)*u(j,i+1);
           dxu(j,i)=mu*(u(j,i+1)-u(j,i))/Dh ;
           pxu=mxu(j,i)/dxu(j,i);
           axu(j,i)=max(mxu(j,i),0)*(dxu(j,i)*(max(0,(1-
0.1*abs(pxu))^5))+max(mxu(j,i),0)-
(dxu(j,i))^5)*u(j,i)+min(mxu(j,i),0)*(dxu(j,i)*(max(0,(1-
0.1*abs(pxu))^5))+min(mxu(j,i),0)-(dxu(j,i))^5)*u(j,i+1);
           if Dh==0.005
           axu(j,i)=max(mxu(j,i),0)*u(j,i)+min(mxu(j,i),0)*u(j,i+1);
           end
       end
    end

   %fluxes in y drection
      for i=2:sx-2
       for j=1:sy-1
           myu(j,i)=rho*(v(j,i)+v(j,i+1))/2;
           ayu(j,i)=max(myu(j,i),0)*u(j,i)+min(myu(j,i),0)*u(j+1,i);
           dyu(j,i)=mu*(u(j+1,i)-u(j,i))/Dl ;
           pyu=myu(j,i)/dyu(j,i);
           ayu(j,i)=max(myu(j,i),0)*(dyu(j,i)*(max(0,(1-
0.1*abs(pyu))^5))+max(myu(j,i),0)-
(dyu(j,i))^5)*u(j,i)+min(myu(j,i),0)*(dyu(j,i)*(max(0,(1-
0.1*abs(pyu))^5))+min(myu(j,i),0)-(dyu(j,i))^5)*u(j,i+1);
           if Dh==0.005
           ayu(j,i)=max(myu(j,i),0)*u(j,i)+min(myu(j,i),0)*u(j+1,i);
           end
       end
      end


%% U control volume Predictor Step

   for  i=2:sx-2
       for j=2:sy-1
           Add(j,i)=(axu(j,i)-axu(j,i-1))*Dl+(ayu(j,i)-ayu(j-1,i))*dl;
           Diff(j,i)=(dxu(j,i)-dxu(j,i-1))*Dl+(dyu(j,i)-dyu(j-1,i))*dl;
           Sourc(j,i)=(p(j,i)-p(j,i+1))*Dl;
           ustar(j,i)=uold(j,i)+(dt/(rho*Dl*dl))*(Diff(j,i)-Add(j,i)+Sourc(j,i));
       end
   end

%% V control volume fluxes
```

```
  %fluxes in x direction.
  for i=1:sx-1
      for j=2:sy-2
          mxv(j,i)=rho*(u(j,i)+u(j+1,i))/2;
          axv(j,i)=max(mxv(j,i),0)*v(j,i)+min(mxv(j,i),0)*v(j,i+1);
          dxv(j,i)=mu*(v(j,i+1)-v(j,i))/dl;
          pxv=mxv(j,i)/dxv(j,i);
          axv(j,i)=min(mxv(j,i),0)*(dxv(j,i)*(max(0,(1-
0.1*abs(pxv))^5))+max(mxv(j,i),0)-
(dxv(j,i))^5)*v(j,i+1)+max(mxv(j,i),0)*(dxv(j,i)*(max(0,(1-
0.1*abs(pxv))^5))+min(mxv(j,i),0)-(dxv(j,i))^5)*v(j,i);
          if Dh==0.005
          axv(j,i)=max(mxv(j,i),0)*v(j,i)+min(mxv(j,i),0)*v(j,i+1);
          end
      end
  end

  %fluxes in y direction.
  for i=2:sx-1
      for j=1:sy-2
          myv(j,i)=rho*(v(j+1,i)+v(j,i))/2;
          ayv(j,i)=max(myv(j,i),0)*v(j,i)+min(myv(j,i),0)*v(j+1,i);
          dyv(j,i)=mu*(v(j+1,i)-v(j,i))/Dl;
          pyv=myv(j,i)/dyv(j,i);
          ayv(j,i)=min(myv(j,i),0)*(dyv(j,i)*(max(0,(1-
0.1*abs(pyv))^5))+max(myv(j,i),0)-
(dyv(j,i))^5)*v(j,i+1)+max(myv(j,i),0)*(dyv(j,i)*(max(0,(1-
0.1*abs(pyv))^5))+min(myv(j,i),0)-dyv(j,i))*v(j,i);
          if Dh==0.005
          ayv(j,i)=max(myv(j,i),0)*v(j,i)+min(myv(j,i),0)*v(j+1,i);
          end
      end
  end

%% V control volume Predictor Step

   for  i=2:sx-1
       for j=2:sy-2
           Add(j,i)=(axv(j,i)-axv(j,i-1))*dh+(ayv(j,i)-ayv(j-1,i))*Dh;
           Diff(j,i)=(dxv(j,i)-dxv(j,i-1))*dh+(dyv(j,i)-dyv(j-1,i))*Dh;
           Sourc(j,i)=(p(j,i)-p(j+1,i))*Dh;
           vstar(j,i)=vold(j,i)+(dt/(rho*Dh*dh))*(Diff(j,i)-Add(j,i)+Sourc(j,i));
       end
   end


%% SIMPLE method loop

 pdash(1:sy,1:sx)=0;
 error=1;

   while error>1e-3

        pdash(1:sy,1)=pdash(1:sy,2);          %left boundary contion
        pdash(1:sy,sx)=pdash(1:sy,sx-1); %right boundary condition
        pdash(1,1:sx)=pdash(2,1:sx);          %bottom boundary
        pdash(sy,1:sx)=pdash(sy-1,1:sx); %top boundary condition

       for i=2:sx-1
           for j=2:sy-1
                b(j,i)=(ustar(j,i)-ustar(j,i-1))*rho*Dl+(vstar(j,i)-vstar(j-
1,i))*rho*Dh;
           end
```

```
        end

 %% point by point gauss seidel method

         aE=rho*dt*Dl/Dh;
         aW=aE;
         aN=rho*dt*Dh/Dl;
         aS=aN;
         aP=aW+aE+aS+aN;

        for i=2:sx-1
            for j=2:sy-1
                pdash(j,i)=aE*pdash(j,i+1)+aW*pdash(j,i-
1)+aN*pdash(j+1,i)+aS*pdash(j-1,i)-b(j,i);
                pdash(j,i)=pdash(j,i)/aP;
                p(j,i)=p(j,i)+pdash(j,i);
            end
        end
        for i=2:sx-1
            for j=2:sy-1
                if i<(sx-1)
                    ustar(j,i)=ustar(j,i)+(dt/(rho*dl*Dl))*(pdash(j,i)-
pdash(j,i+1))*Dl;
                end
                if j<(sy-1)
                    vstar(j,i)=vstar(j,i)+(dt/(rho*Dh*dh))*(pdash(j,i)-
pdash(j+1,i))*Dh;
                end
            end
        end

     error=max(max(abs(b)));
    end

     espiu=max(max(abs(ustar-uold)))/dt;
     epsiv=max(max(abs(vstar-vold)))/dt;

     epsi=max(espiu,epsiv);

     vold=vstar;
     v=vstar;
     uold=ustar;
     u=ustar;

     itr=itr+1;


   end

   flg=flg+1;

   %% Control volume velocity calculation

   U=zeros(sy,sx);
   V=zeros(sy,sx);
   U(sy,1:sx)=un;
      for j=2:sy-1
          for i=2:sx-1
              U(j,i)=(u(j,i)+u(j,i-1))/2;
          end
      end
      for j=2:sy-1
          for i=2:sx-1
```

```
                V(j,i)=(v(j,i)+v(j-1,i))/2;
            end
        end

    %% calculation for midsection velocities

      c=mod(sx,2);
      if c==0
          n=sx/2;
          avgu=(U(:,n)+U(:,n+1))/2;
          avgv=(V(n,:)+V(n+1,:))/2;
      else
          n=(sx+1)/2;
          avgu=U(:,n);
          avgv=V(n,:);
      end

      if flg==1
          dum1=abs(avgu);
          dvm1=abs(avgv);
          Dh1=Dh;

      elseif flg==2
          dum2=abs(avgu);
          dvm2=abs(avgv);
          Dh2=Dh;

      else
          dum3=abs(avgu);
          dvm3=abs(avgv);
          Dh3=Dh;

      end

end

rich_erru1=abs(dum1(1)-dum2(2));
rich_erru2=abs(dum2(2)-dum3(4));
rich_errv1=abs(dvm1(1)-dvm2(2));
rich_errv2=abs(dvm2(2)-dvm3(4));

eu=(log(rich_erru1/rich_erru2))/log(2);
ev=(log(rich_errv1/rich_errv2))/log(2);

cu=rich_erru1/((Dh2)^eu-(Dh1)^eu);
err_richu1=cu*(Dh1)^eu
err_richu2=cu*(Dh2)^eu
err_richu3=cu*(Dh3)^eu

cv=rich_erru1/((Dh2)^ev-(Dh1)^ev);
err_richv1=cv*(Dh1)^ev
err_richv2=cv*(Dh2)^ev
err_richv3=cv*(Dh3)^ev


%% ----------------------Normal loop-------------------------------

else


rho=1;
nu=0.004;
un=input('\nInput Lid Velocity: ');
```

```
mu=rho*nu;
l=0.2;
w=0.2;
Re=un*rho*l/nu;
Dh=input('\nGrid size in x direction: ');
Dl=input('\nGrid size in y direction: ');
dl=Dh;
dh=Dl;
sx=(l/dl)+2;
sy=(w/dh)+2;


%% Initializing pressure and velocity

%initial pressure
p=zeros(sx,sy);

% u velocity boundary conditions and initial condition.
u=zeros(sy,sx-1);
u(1:sy,1)=0;                %left wall boundary
u(1:sy,sx-1)=0;           %right wall boundary condition
u(1,2:sx-1)=0;              %bottom boundary
u(sy,1:sx-1)=un;          %top boundary
uold=u;
ustar=u;

% v velocity boundary conditions and initial condition.
v=zeros(sy-1,sx);
vstar=v;
vold=v;

%time step
Dt_a=Dh/abs(max(max(u)));
Dt_d=2*mu/rho*(1/Dh^2+1/Dl^2);
Dt_d=1/Dt_d;
dt=min(Dt_d,Dt_a);
dt=0.5*dt;
itr=0;

%% SIMPLE method

% main while
  epsi=1;

  while epsi>1e-3


%% U control volume fluxes

 %fluxes in x drection
    for i=1:sx-2
       for j=2:sy-1
           mxu(j,i)=rho*(u(j,i)+u(j,i+1))/2;
           dxu(j,i)=mu*(u(j,i+1)-u(j,i))/Dh ;
           pxu=mxu(j,i)/dxu(j,i);
           axu(j,i)=max(mxu(j,i),0)*(dxu(j,i)*(max(0,(1-
0.1*abs(pxu))^5))+max(mxu(j,i),0)-
(dxu(j,i))^5)*u(j,i)+min(mxu(j,i),0)*(dxu(j,i)*(max(0,(1-
0.1*abs(pxu))^5))+min(mxu(j,i),0)-(dxu(j,i))^5)*u(j,i+1);
           if Dh==0.005
           axu(j,i)=max(mxu(j,i),0)*u(j,i)+min(mxu(j,i),0)*u(j,i+1);
           end
       end
```

```
    end

   %fluxes in y drection
      for i=2:sx-2
       for j=1:sy-1
           myu(j,i)=rho*(v(j,i)+v(j,i+1))/2;
           ayu(j,i)=max(myu(j,i),0)*u(j,i)+min(myu(j,i),0)*u(j+1,i);
           dyu(j,i)=mu*(u(j+1,i)-u(j,i))/Dl ;
           pyu=myu(j,i)/dyu(j,i);
           ayu(j,i)=max(myu(j,i),0)*(dyu(j,i)*(max(0,(1-
0.1*abs(pyu))^5))+max(myu(j,i),0)-
(dyu(j,i))^5)*u(j,i)+min(myu(j,i),0)*(dyu(j,i)*(max(0,(1-
0.1*abs(pyu))^5))+min(myu(j,i),0)-(dyu(j,i))^5)*u(j,i+1);
           if Dh==0.005
           ayu(j,i)=max(myu(j,i),0)*u(j,i)+min(myu(j,i),0)*u(j+1,i);
           end
       end
      end


%% U control volume Predictor Step

   for  i=2:sx-2
       for j=2:sy-1
           Add(j,i)=(axu(j,i)-axu(j,i-1))*Dl+(ayu(j,i)-ayu(j-1,i))*dl;
           Diff(j,i)=(dxu(j,i)-dxu(j,i-1))*Dl+(dyu(j,i)-dyu(j-1,i))*dl;
           Sourc(j,i)=(p(j,i)-p(j,i+1))*Dl;
           ustar(j,i)=uold(j,i)+(dt/(rho*Dl*dl))*(Diff(j,i)-Add(j,i)+Sourc(j,i));
       end
   end

%% V control volume fluxes

  %fluxes in x direction.
  for i=1:sx-1
      for j=2:sy-2
          mxv(j,i)=rho*(u(j,i)+u(j+1,i))/2;
          axv(j,i)=max(mxv(j,i),0)*v(j,i)+min(mxv(j,i),0)*v(j,i+1);
          dxv(j,i)=mu*(v(j,i+1)-v(j,i))/dl;
          pxv=mxv(j,i)/dxv(j,i);
          axv(j,i)=min(mxv(j,i),0)*(dxv(j,i)*(max(0,(1-
0.1*abs(pxv))^5))+max(mxv(j,i),0)-
(dxv(j,i))^5)*v(j,i+1)+max(mxv(j,i),0)*(dxv(j,i)*(max(0,(1-
0.1*abs(pxv))^5))+min(mxv(j,i),0)-(dxv(j,i))^5)*v(j,i);
          if Dh==0.005
          axv(j,i)=max(mxv(j,i),0)*v(j,i)+min(mxv(j,i),0)*v(j,i+1);
          end
      end
  end

  %fluxes in y direction.
  for i=2:sx-1
      for j=1:sy-2
          myv(j,i)=rho*(v(j+1,i)+v(j,i))/2;
          ayv(j,i)=max(myv(j,i),0)*v(j,i)+min(myv(j,i),0)*v(j+1,i);
          dyv(j,i)=mu*(v(j+1,i)-v(j,i))/Dl;
          pyv=myv(j,i)/dyv(j,i);
          ayv(j,i)=min(myv(j,i),0)*(dyv(j,i)*(max(0,(1-
0.1*abs(pyv))^5))+max(myv(j,i),0)-
(dyv(j,i))^5)*v(j,i+1)+max(myv(j,i),0)*(dyv(j,i)*(max(0,(1-
0.1*abs(pyv))^5))+min(myv(j,i),0)-dyv(j,i))*v(j,i);
          if Dh==0.005
          ayv(j,i)=max(myv(j,i),0)*v(j,i)+min(myv(j,i),0)*v(j+1,i);
          end
```

```
      end
    end

%% V control volume Predictor Step

    for  i=2:sx-1
        for j=2:sy-2
            Add(j,i)=(axv(j,i)-axv(j,i-1))*dh+(ayv(j,i)-ayv(j-1,i))*Dh;
            Diff(j,i)=(dxv(j,i)-dxv(j,i-1))*dh+(dyv(j,i)-dyv(j-1,i))*Dh;
            Sourc(j,i)=(p(j,i)-p(j+1,i))*Dh;
            vstar(j,i)=vold(j,i)+(dt/(rho*Dh*dh))*(Diff(j,i)-Add(j,i)+Sourc(j,i));
        end
    end


%% SIMPLE method loop

 pdash(1:sy,1:sx)=0;
 error=1;

    while error>1e-3

         pdash(1:sy,1)=pdash(1:sy,2);          %left boundary contion
         pdash(1:sy,sx)=pdash(1:sy,sx-1); %right boundary condition
         pdash(1,1:sx)=pdash(2,1:sx);          %bottom boundary
         pdash(sy,1:sx)=pdash(sy-1,1:sx); %top boundary condition

        for i=2:sx-1
            for j=2:sy-1
                  b(j,i)=(ustar(j,i)-ustar(j,i-1))*rho*Dl+(vstar(j,i)-vstar(j-
1,i))*rho*Dh;
            end
        end

 %% point by point gauss seidel method

         aE=rho*dt*Dl/Dh;
         aW=aE;
         aN=rho*dt*Dh/Dl;
         aS=aN;
         aP=aW+aE+aS+aN;

        for i=2:sx-1
            for j=2:sy-1
                pdash(j,i)=aE*pdash(j,i+1)+aW*pdash(j,i-
1)+aN*pdash(j+1,i)+aS*pdash(j-1,i)-b(j,i);
                pdash(j,i)=pdash(j,i)/aP;
                p(j,i)=p(j,i)+pdash(j,i);
            end
        end
        for i=2:sx-1
            for j=2:sy-1
                if i<(sx-1)
                    ustar(j,i)=ustar(j,i)+(dt/(rho*dl*Dl))*(pdash(j,i)-
pdash(j,i+1))*Dl;
                end
                if j<(sy-1)
                    vstar(j,i)=vstar(j,i)+(dt/(rho*Dh*dh))*(pdash(j,i)-
pdash(j+1,i))*Dh;
                end
            end
        end
```

```
    error=max(max(abs(b)));
   end

    espiu=max(max(abs(ustar-uold)))/dt;
    epsiv=max(max(abs(vstar-vold)))/dt;

    epsi=max(espiu,epsiv);

    vold=vstar;
    v=vstar;
    uold=ustar;
    u=ustar;

    itr=itr+1;
%     plot(itr,epsi,'ro');
%     hold on;

  end


  %% Output


  %plotting of Grid points
  x(1)=0;
  x(sx-1)=l;

  for i=2:sx-2
  x(i)=x(i-1)+Dh;
  end

  y=x;

  %%plotting control surfaces
  xc(1)=x(1);xc(sx)=x(sx-1);
  for i=2:sx-1
      xc(i)=(x(i)+x(i-1))/2;
  end

  yc=xc;

  [xx yy]=meshgrid(xc,yc);

  % vector plot
  U=zeros(sy,sx);
  V=zeros(sy,sx);
  U(sy,1:sx)=un;
     for j=2:sy-1
         for i=2:sx-1
             U(j,i)=(u(j,i)+u(j,i-1))/2;
         end
     end
     for j=2:sy-1
         for i=2:sx-1
             V(j,i)=(v(j,i)+v(j-1,i))/2;
         end
     end




    figure
```

```
    quiver(xx,yy,U,V,3)
    xlim([0 l])
    ylim([0 w])
    xlabel('X cordinates');
    ylabel('Y cordinates');
    title('Velocity Vector plot')

    figure
    contourf(xc,yc,p,15);
    xlim([0 l])
    ylim([0 w])
    xlabel('X cordinates');
    ylabel('Y cordinates');
    title('pressure contours')

    figure
    colormap jet
    contourf(xc,yc,U,15);
    xlim([0 l])
    ylim([0 w])
    xlabel('X cordinates');
    ylabel('Y cordinates');
    title('U-velocity contour');


    %% calculation for midsection velocities

    c=mod(sx,2);
    if c==0
        n=sx/2;
        avgu=(U(:,n)+U(:,n+1))/2;
        avgv=(V(n,:)+V(n+1,:))/2;
    else
        n=(sx+1)/2;
        avgu=U(:,n);
        avgv=V(n,:);
    end

    %% Comparison with ghia et al

    % Note: Ghia data at Re 100 is calculated. Take un=0.4 for this.

    figure
    plot(avgu,yc/w,'b-o','MarkerFaceColor','r','MarkerSize',3)
    ylim([0 1])
    xlabel('U');
    ylabel('Y cordinates');
    title('U-velocity plot');
    hold on
    u_ghia=[1.0000 0.65928 0.78871 0.73722 0.68717 0.23151 0.00332 -0.136641 -
0.20581 -0.2109 -0.15662 -0.1015 -0.063434 -0.04775 -0.04192 -0.03717 0.00000];
    y_ghia=[1 0.9766 0.9688 0.9609 0.9531 0.8516 0.7344 0.6172 0.5 0.4531 0.2813
0.1719 0.1016 0.0703 0.0625 0.0547 0.0000];
    if un==2;
    plot(u_ghia,y_ghia,'r-*','MarkerFaceColor','r');
    legend('Numerical data','ghia data');
    end


    figure
    plot(xc/l,avgv,'b-o','MarkerFaceColor','r','MarkerSize',3);
    xlim([0 1])
    xlabel('X cordinates');
    ylabel('V-velocity');
```

```
    title('V-velocity plot');
    hold on;
    x_ghia=[1 0.9688 0.9609 0.9531 0.9453 0.9063 0.8594 0.8047 0.5 0.2344 0.2266
0.1563 0.0938 0.0781 0.0703 0.0625 0];
    v_ghia= [0 -0.05906 -0.07391 -0.08864 -0.10313 -0.16914 -0.22445 -0.24533
0.05454 0.17527 0.17507 0.16077 0.12317 0.1089 0.100091 0.09233 0];

    if un==2;
    plot(x_ghia,v_ghia,'r-d','MarkerFaceColor','r');
    legend('Numerical data','ghia data');
    end






fprintf('\nReynold''s Number is : %d',Re);
hold off


%% Grid visualization

%Note: Grid calculation is done again as the code was sluggish
%At higher grid points grid did not look good
%So the plotted grid has nothing to do with user input

%% Control points Calculation


sx=(l/dl)+2;
xv(1,1)=0;
xv(sx,1)=l;
xv(2,1)=xv(1,1)+(dl/2);
for m=3:sx-1
    xv(m,1)=xv(m-1,1)+dl;
end

%sy represents the number of points in a grid in y direction

sy=(w/dh)+2;
dh=w/(sy-2);
yv(1,1)=0;
yv(sy,1)=w;
yv(2,1)=yv(1,1)+(dh/2);

for m=3:sy-1
    yv(m,1)=yv(m-1,1)+dh;
end

%% Control volume calculation

xcv=zeros(sx-1,1);
xcv(1)=x(1);
xcv(end)=x(end);

for m=2:sx-2
    xcv(m,1)=(xv(m,1)+xv(m+1,1))/2;
end

ycv=zeros(sy-1,1);
ycv(1)=y(1);
```

```
ycv(end)=y(end);

for m=2:sy-2
    ycv(m,1)=(yv(m,1)+yv(m+1,1))/2;
end


%% Grid plot

figure
m=10;

r=ones(sy-1,sx-1);
map = [1,1,1];
colormap(map)
pcolor(xcv,ycv,r)
title('Grid Layout');
xlabel('Length (m)');
ylabel('Height (m)');
hold on

%U control volume points
b=jet(20);
for i=1:sy-1

    plot(xv,ycv(i),'v','MarkerFaceColor',b(m,:),'MarkerEdgeColor',b(m,:));

end

%V control volume points
b=hot(20);
for i=1:sx-1

    plot(xcv(i),yv,'s','MarkerFaceColor',b(m,:),'MarkerEdgeColor',b(m,:));

end

b=bone(20);

% Actual Grid points
for i=1:sy

    plot(xv,yv(i),'o','MarkerFaceColor',b(m,:),'MarkerEdgeColor',b(m,:));

end
hold off

end




%__________________________________________________________________________________
```

Results and Discussion

A. Reynold's number calculation

1. U = 2 m/s

Lid Driven Flow

Input choice for Richardson Method
1.Yes
2.No
2

Input Lid Velocity: 2

Grid size in x direction: 0.005

Grid size in y direction: 0.005

Reynold's Number is : 100

2. U = 8 m/s

Lid Driven Flow

Input choice for Richardson Method

1.Yes

2.No

2

Input Lid Velocity: 8

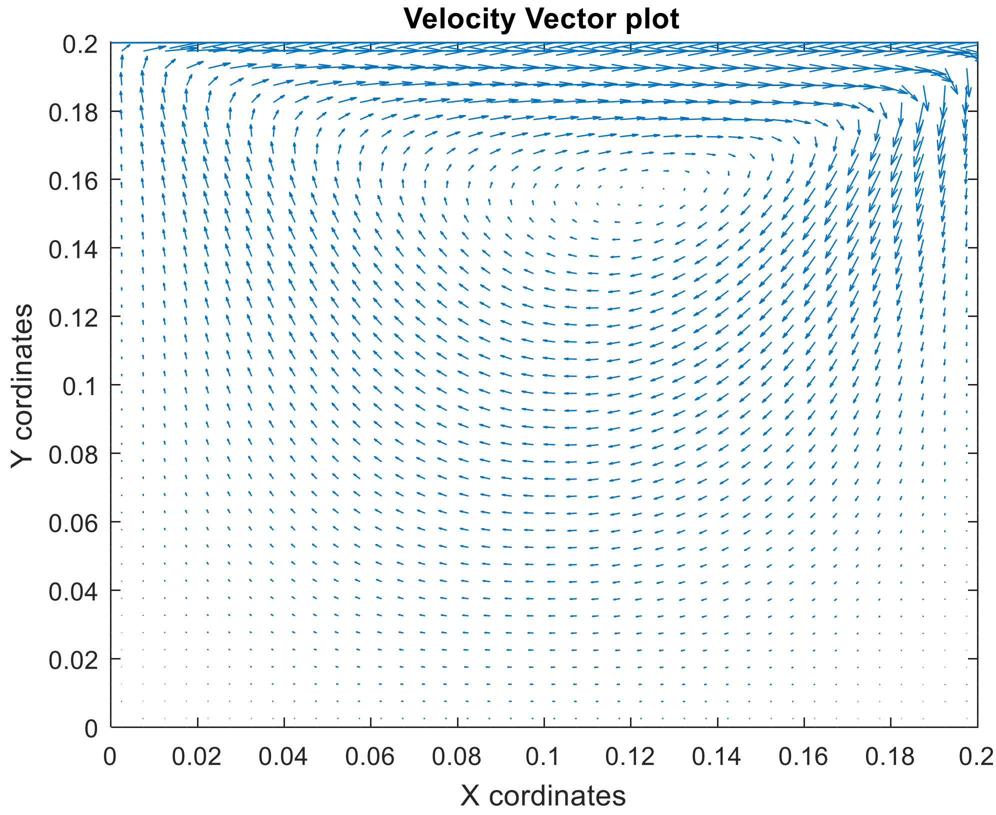
Grid size in x direction: 0.01

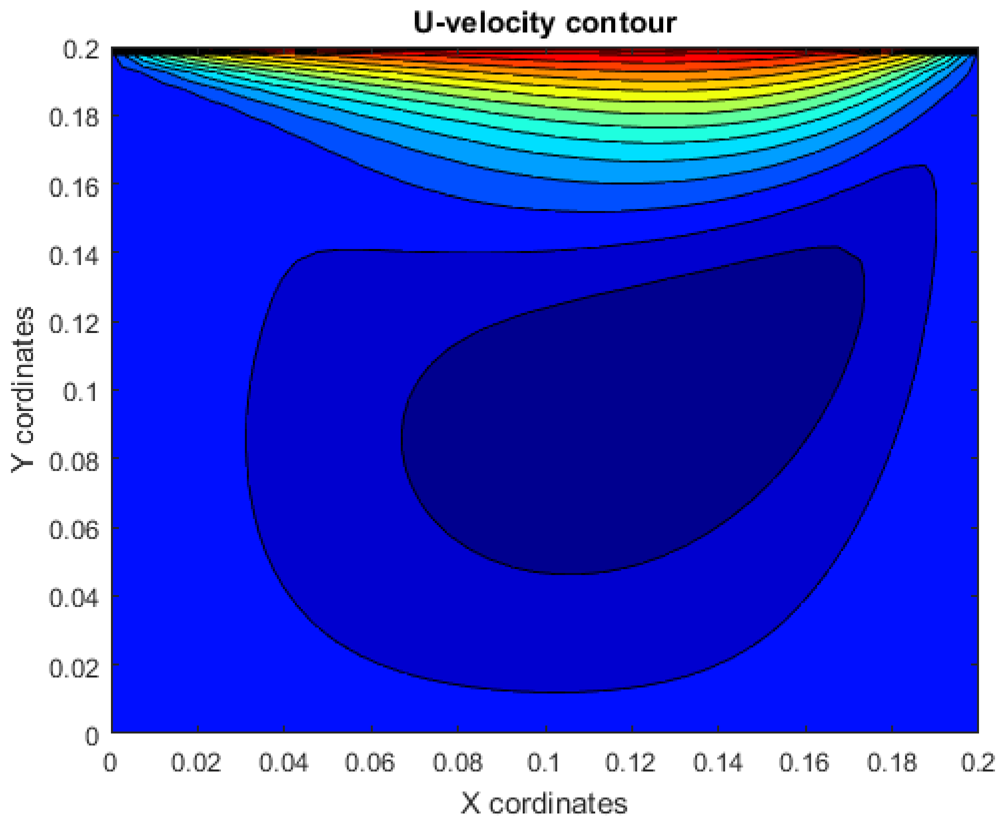
Grid size in y direction: 0.01

Reynold's Number is : 400

B. Velocity calculation

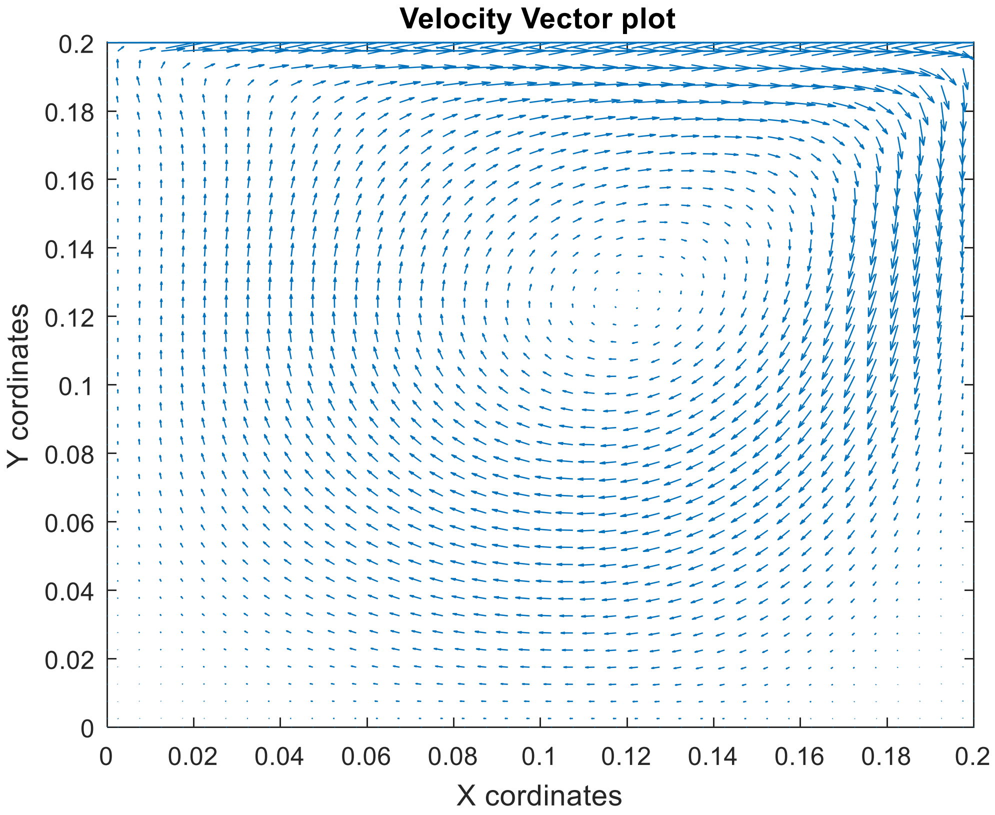
U = 2 m/s

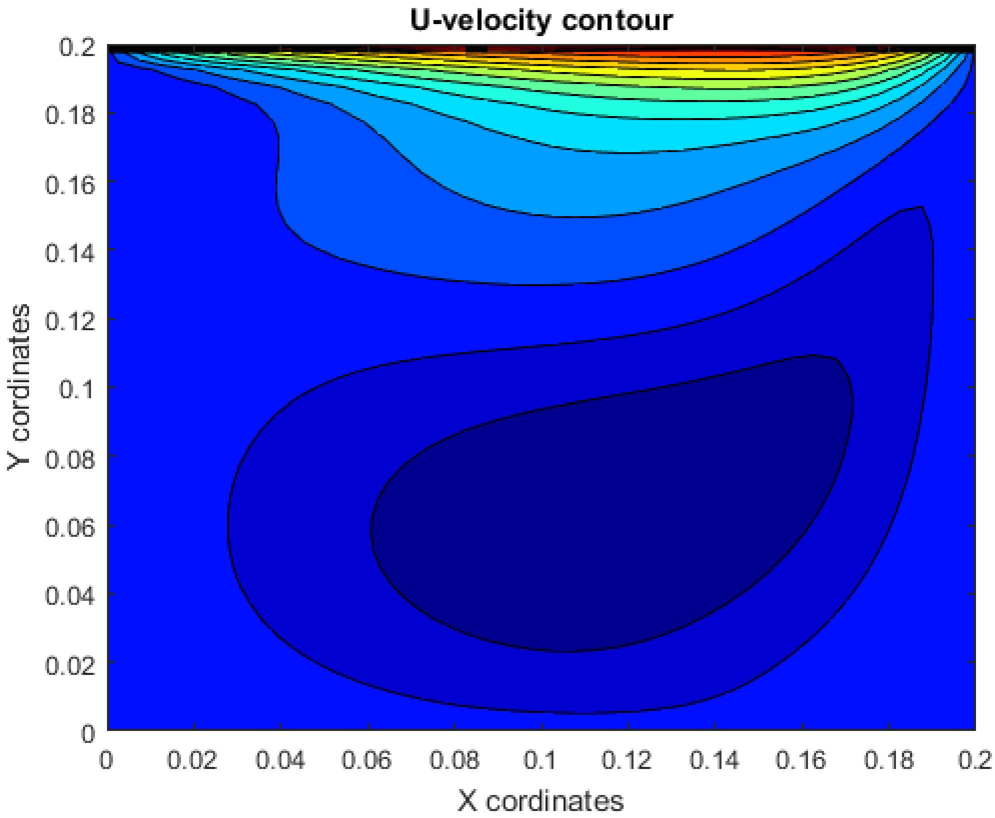




C. Velocity calculation

U = 8m/s





D. Richardson extrapolation

U = 2m/s

Lid Driven Flow

Input choice for Richardson Method

1.Yes

2.No

1

Enter grid size in decreasing order

Input grid size: 0.02

Velocity: 2

Input grid size: 0.01

Velocity: 2

Input grid size: 0.005

Velocity: 2

err_richu1 =

-0.1039

err_richu2 =

-0.0578

err_richu3 =

-0.0322

err_richv1 =

-0.0921

err_richv2 =

-0.0460

err_richv3 =

-0.0230

U = 8m/s

Lid Driven Flow

Input choice for Richardson Method

1.Yes

2.No

1

Enter grid size in decreasing order

Input grid size: 0.02

Velocity: 8

Input grid size: 0.01

Velocity: 8

Input grid size: 0.005

Velocity: 8

err_richu1 =

0.0242

err_richu2 =

0.1281

err_richu3 =

0.6781

err_richv1 =

0.0258

err_richv2 =

0.1297

err_richv3 =

0.6519

E. Comparison with ghia paper