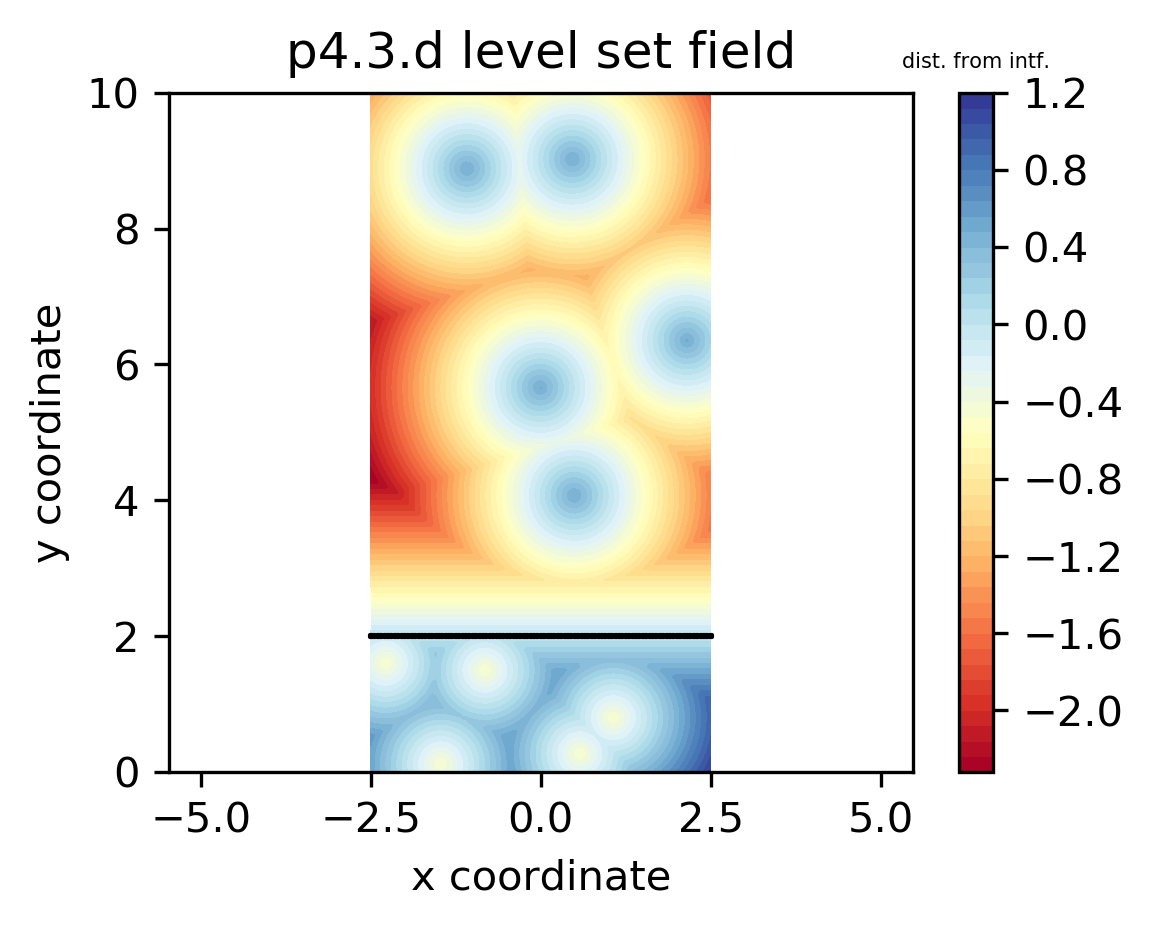
**一張含有 文字, 白板 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述一張含有 文字, 白板 的圖片

自動產生的描述**

**4.3 (d)**



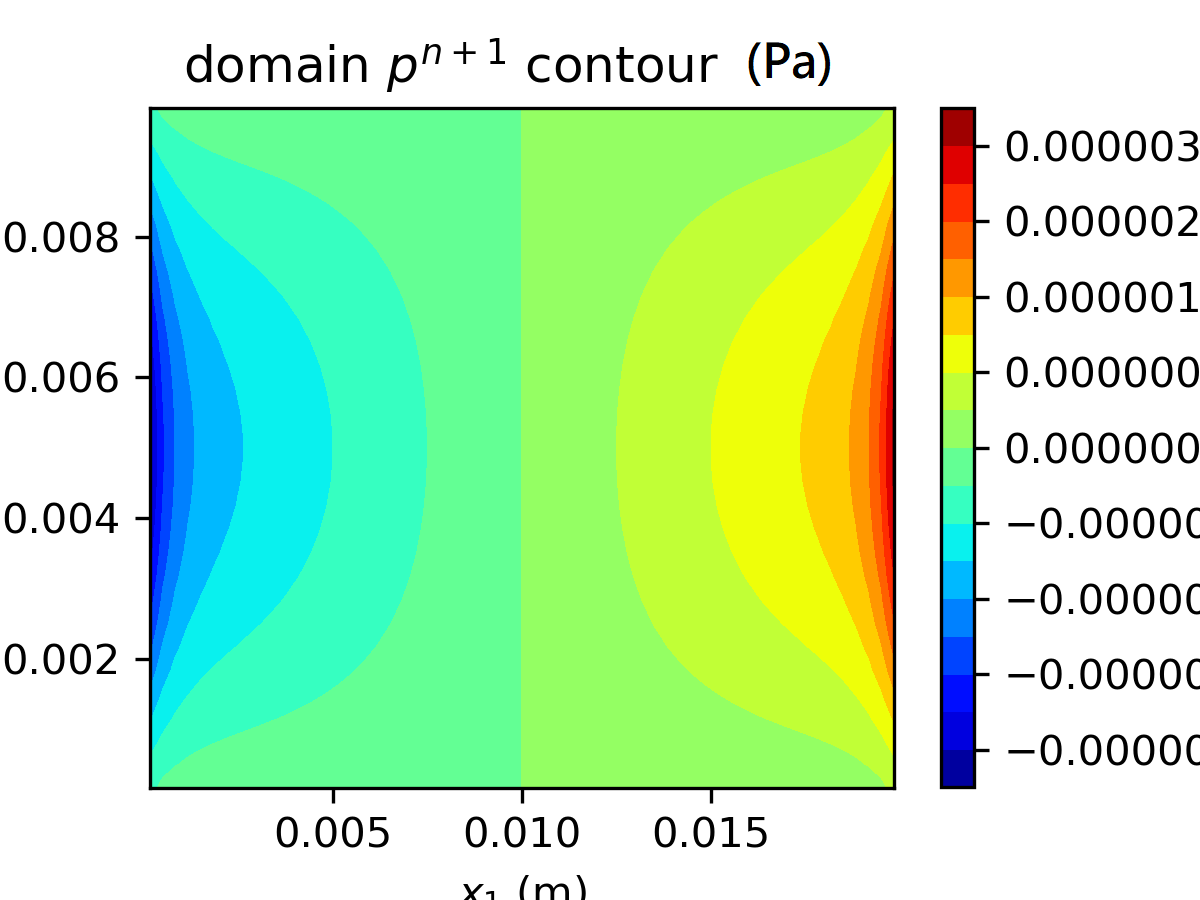
The code generating the level-set field is attached to this file. The water level is marked using the solid black line at y=2. Five bubbles and droplets are generated within liquid and gas domain respectively with randomly selected center coordinate. Radius of bubbles/droplets are assigned to be .

The resulting level-set field is believed to be correct since it properly shows the feature that level-set values on the interfaces are . Then in the continuous phase, level-set value is positive/negative at liquid/gas. As for the dispersed phase, is negative/positive in bubbles/droplets with minimum/maximum value / respectively. Therefore, the level-set field shows the features as required.

**4.4 (a)**

All initiated with uniform . Code is attached behind.

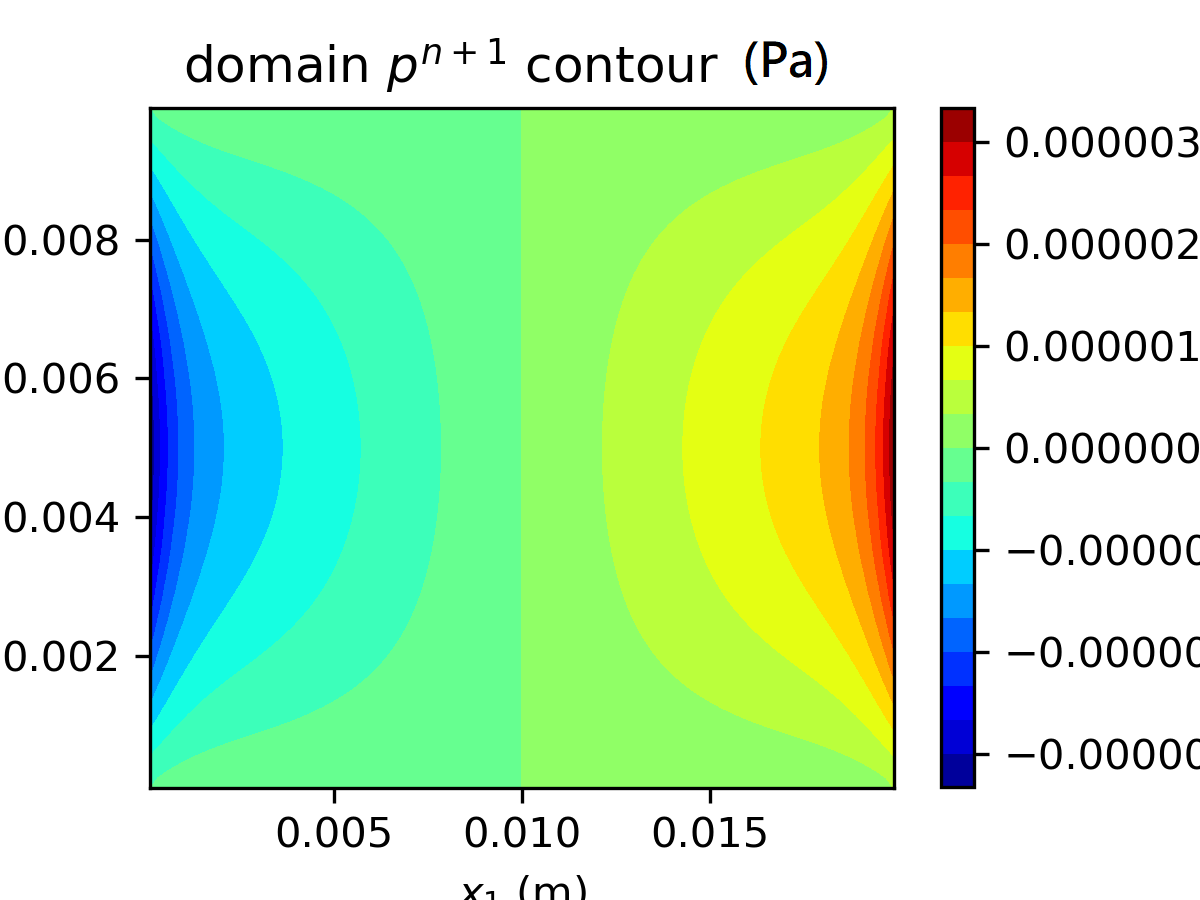
1. Domain with 30 pressure cells

一張含有 螢幕擷取畫面 的圖片

自動產生的描述一張含有 遊戲 的圖片

自動產生的描述

1. Domain with 50 pressure cells

一張含有 螢幕擷取畫面 的圖片

自動產生的描述一張含有 遊戲 的圖片

自動產生的描述

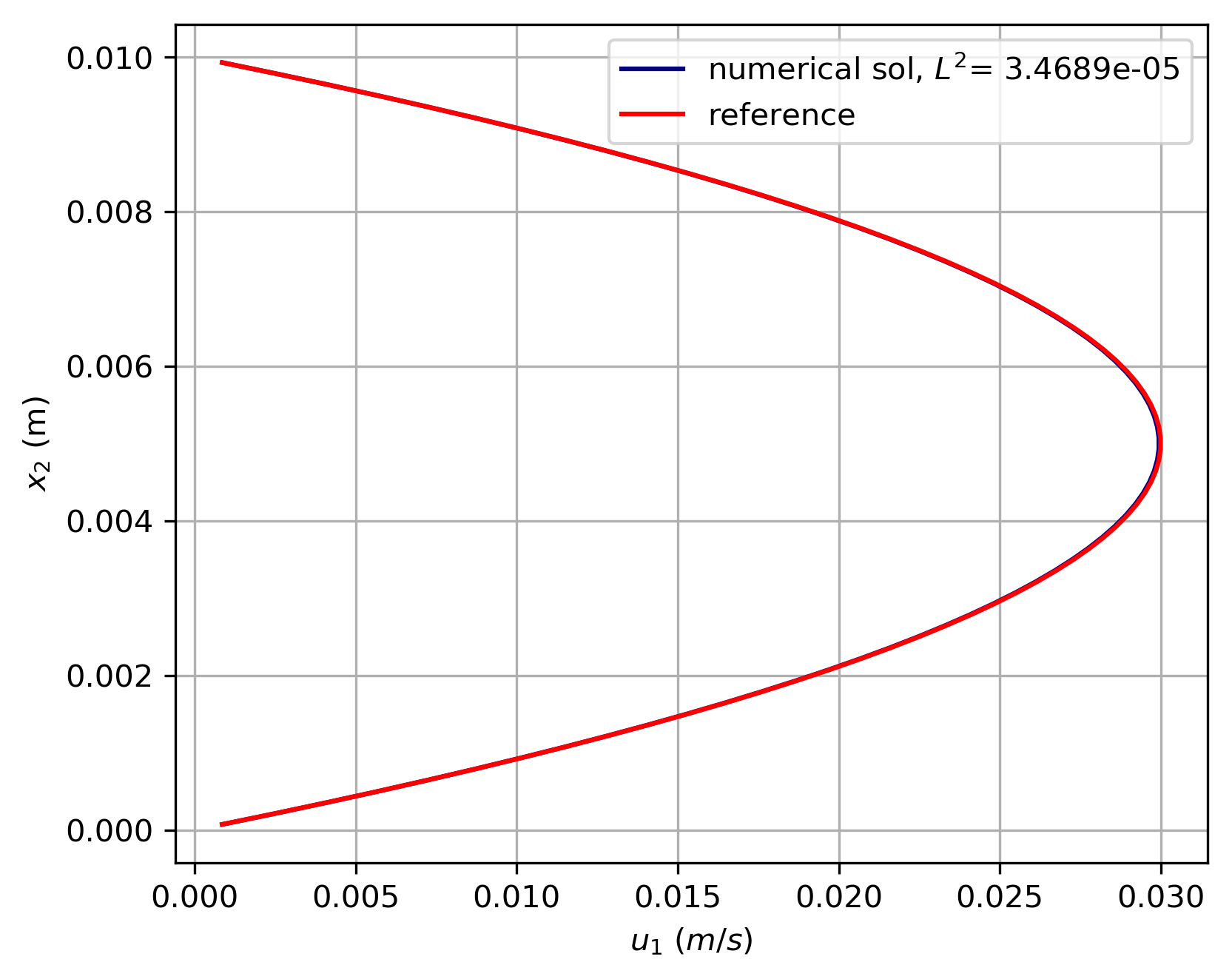
1. Domain with 70 pressure cells

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

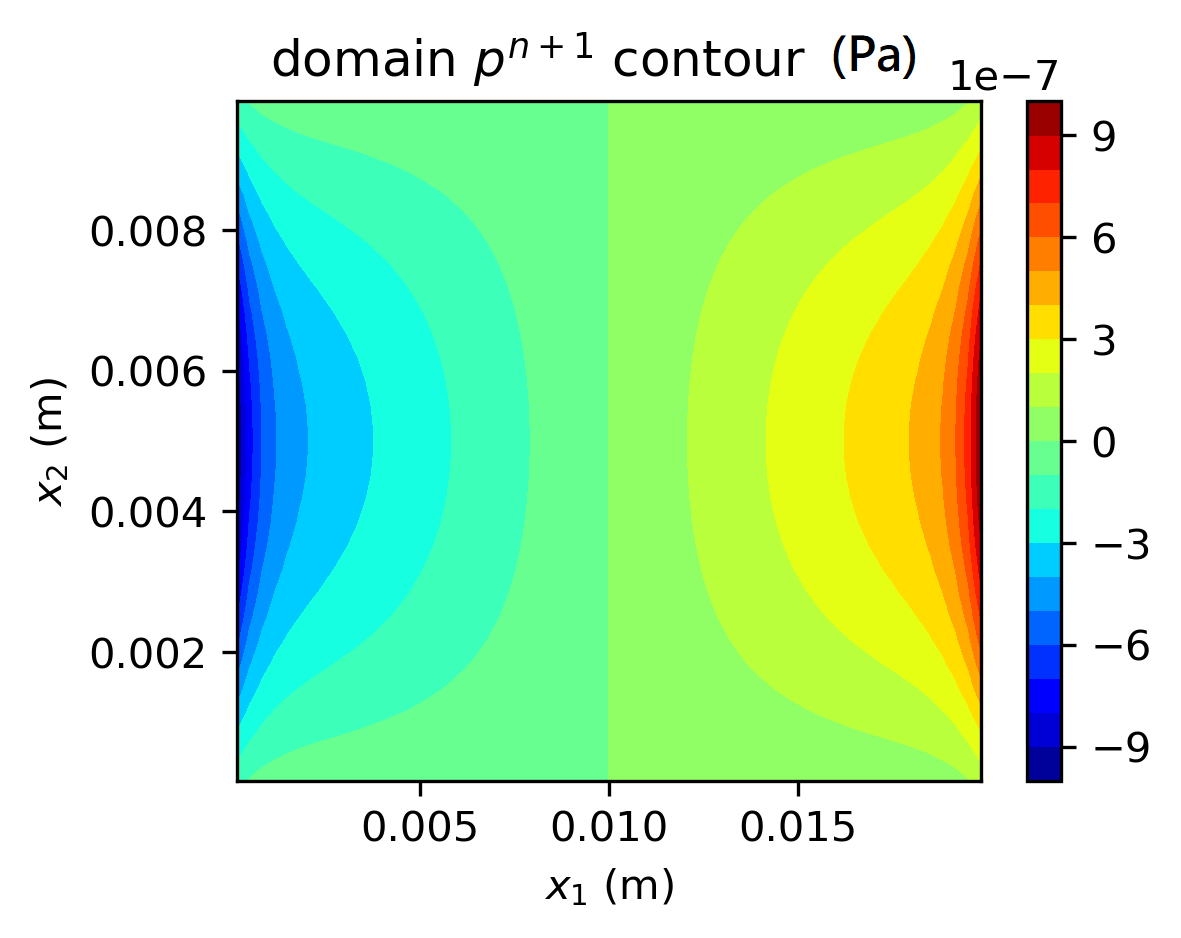
一張含有 螢幕擷取畫面 的圖片

自動產生的描述

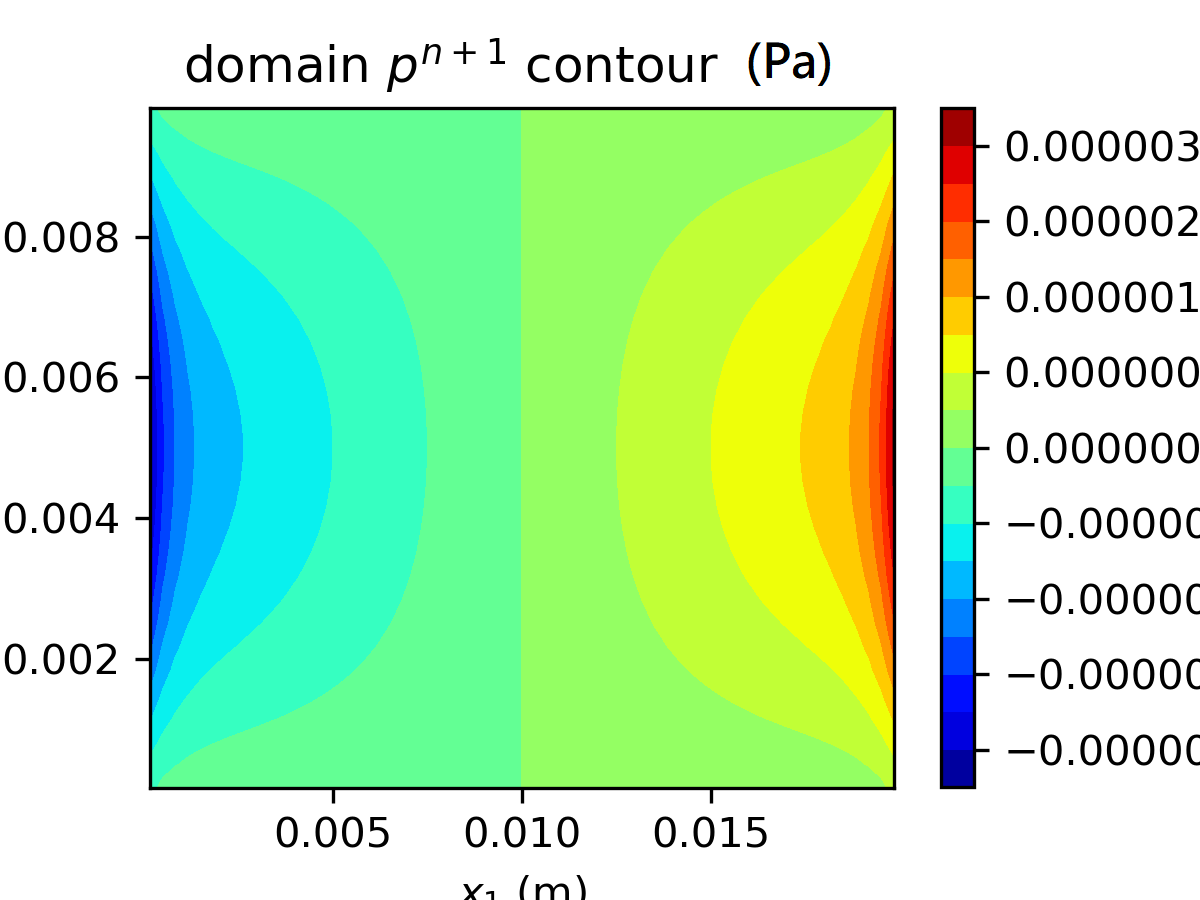


All cases eventually converged towards the reference parabolic velocity profile, with metric at order of . The resulting pressure field is hourglass shaped with largest pressure gradient occurred at channel center level, where the maximum velocity appears. However, the result of pressure field is somewhat inconsistent with the analytical solution derived in previous problem 3.4, where the ought to be . Here, the pressure difference across the inlet and outlet is far lower than the expected value. Such issue may originate from the boundary condition of pressure cells. It requires further examination to exclude the problem.

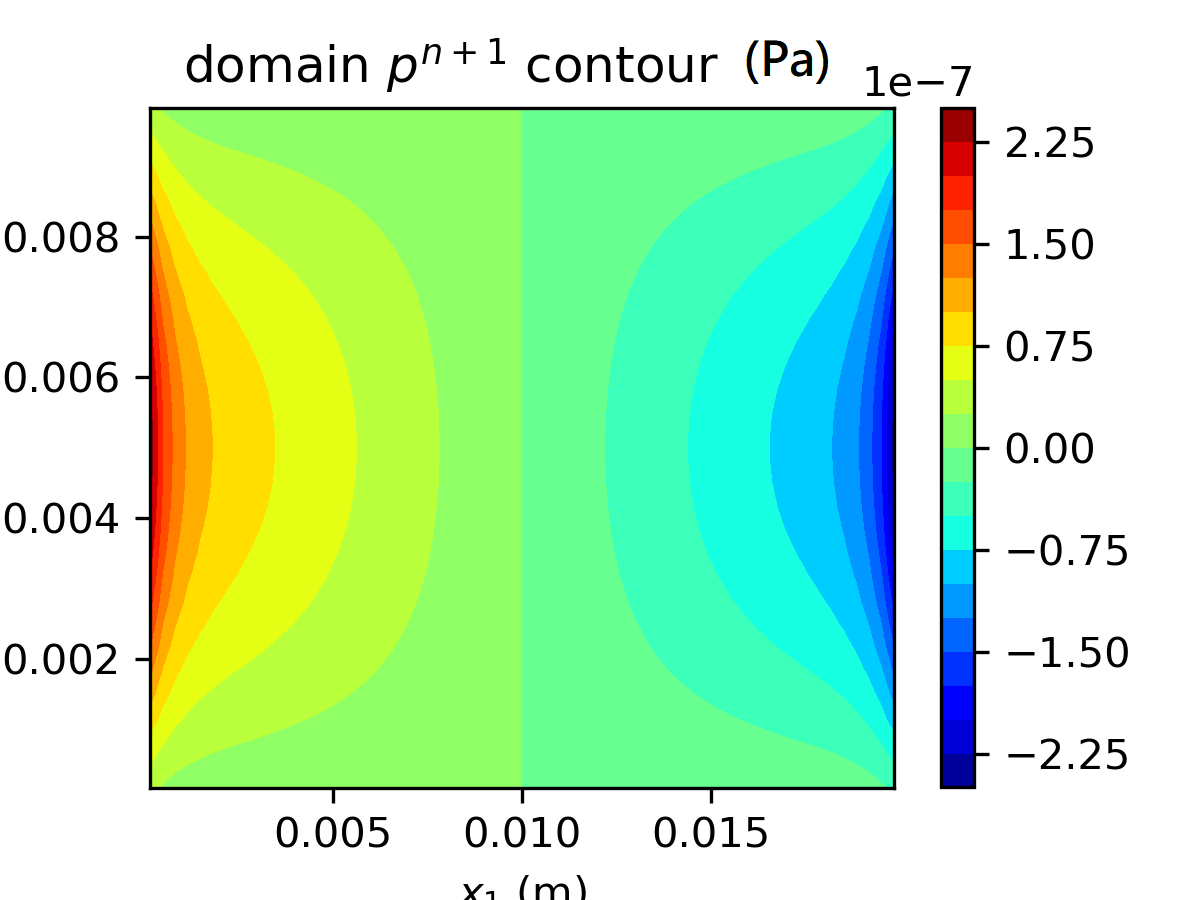
**4.4 (b)**

一張含有 螢幕擷取畫面 的圖片

自動產生的描述一張含有 遊戲 的圖片

自動產生的描述一張含有 螢幕擷取畫面 的圖片

自動產生的描述一張含有 遊戲 的圖片

自動產生的描述一張含有 螢幕擷取畫面 的圖片

自動產生的描述一張含有 遊戲 的圖片

自動產生的描述

From the results above, it can be concluded that the final steady-state solution is independent to the initial velocity assignment. The basic reason behind is deduced to be the driving pressure. In the predictor step:

And the corrector step keeps the field to be divergence free. Therefore, the final velocity field is independent to the initial condition. However, from the results of pressure field, the initial condition do affect final pressure solution since the flow is driven by the pressure gradient. For instance, the case with initial velocity would have almost opposite pressure field distribution trend compared to case with .

**Attachment A, source code for problem 4.3**

**import math**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import numpy.random as rdm**

**y\_w=2**

**x\_d=np.zeros(5)**

**y\_d=np.zeros(5)**

**x\_b=np.zeros(5)**

**y\_b=np.zeros(5)**

**r\_b=np.zeros(5)**

**r\_d=np.zeros(5)**

**for i in range(0,5):**

**x\_d[i]=rdm.uniform(-2.5,2.5)**

**y\_d[i]=rdm.uniform(2.0,10.0)**

**x\_b[i]=rdm.uniform(-2.5,2.5)**

**y\_b[i]=rdm.uniform(0.0,2.0)**

**#r\_b[i]=rdm.uniform(1e-4,5e-4)**

**r\_b[i]=0.5**

**#r\_d[i]=rdm.uniform(1e-4,5e-4)**

**r\_d[i]=0.5**

**def lvlset(x,y):**

**def ls\_drop(x,y,i):**

**return -1\*(math.sqrt((x-x\_d[i])\*\*2+(y-y\_d[i])\*\*2)-r\_d[i])**

**def ls\_bbl(x,y,i):**

**return (math.sqrt((x-x\_b[i])\*\*2+(y-y\_b[i])\*\*2)-r\_b[i])**

**if y>y\_w:**

**return max(-(y-y\_w), ls\_drop(x,y,0), ls\_drop(x,y,1), ls\_drop(x,y,2), ls\_drop(x,y,3), ls\_drop(x,y,4))**

**else:**

**return min(-(y-y\_w), ls\_bbl(x,y,0), ls\_bbl(x,y,1), ls\_bbl(x,y,2), ls\_bbl(x,y,3), ls\_bbl(x,y,4))**

**water\_lvl\_x=np.zeros([101])**

**water\_lvl\_y=np.zeros([101])**

**grid\_x=np.zeros([101,201])**

**grid\_y=np.zeros([101,201])**

**grid\_ls=np.zeros([101,201])**

**for j in range(0,201):**

**for i in range(0,101):**

**grid\_x[i,j]=-2.5+i\*5/100**

**grid\_y[i,j]=0+j\*10/200**

**water\_lvl\_x[i]=-2.5+i\*5/100**

**water\_lvl\_y[i]=2**

**grid\_ls[i,j]=lvlset(grid\_x[i,j],grid\_y[i,j])**

**plt.contourf(grid\_x,grid\_y,grid\_ls,50, cmap='RdYlBu')**

**clt=plt.colorbar()**

**clt.ax.set\_title('dist. from intf.',fontsize=5)**

**plt.title('p4.3.d level set field')**

**plt.axis('equal')**

**plt.scatter(water\_lvl\_x,water\_lvl\_y,color='black',s=0.1, marker=',')**

**plt.xlabel('x coordinate')**

**plt.ylabel('y coordinate')**

**plt.show()**

**Attachment B, source code for problem 4.4**

**import numpy as np**

**import math**

**import matplotlib.pyplot as plt**

**#problem constants**

**nu=1e-6**

**mu=1e-3**

**rho=1e+3**

**dt=0.0001**

**gradP=-2.4**

**n\_iter=0**

**'''**

**node generation section**

**'''**

**#domain length**

**Lx1=0.02**

**Lx2=0.01**

**#number of cells on each direction**

**Nx1=100**

**Nx2=50**

**cell\_vol=(Lx1/Nx1)\*(Lx2/Nx2)**

**#mesh spacing**

**h=Lx1/Nx1**

**#uave**

**u1\_ave=0.02**

**def Adv\_x\_n(i,j):**

**return (1/h)\*((0.5\*(cell\_S\_x\_un[i,j]+cell\_S\_x\_un[i+1,j]))\*\*2-(0.5\*(cell\_S\_x\_un[i,j]+cell\_S\_x\_un[i-1,j]))\*\*2+(0.5\*(cell\_S\_x\_un[i,j+1]+cell\_S\_x\_un[i,j]))\*(cell\_S\_y\_vn[i,j+1]+cell\_S\_y\_vn[i+1,j+1])-(0.5\*(cell\_S\_x\_un[i,j]+cell\_S\_x\_un[i,j-1]))\*(0.5\*(cell\_S\_y\_vn[i,j]+cell\_S\_y\_vn[i+1,j])))**

**def Dif\_x\_n(i,j):**

**return (1/(h\*\*2))\*(cell\_S\_x\_un[i+1,j]+cell\_S\_x\_un[i-1,j]+cell\_S\_x\_un[i,j+1]+cell\_S\_x\_un[i,j-1]-4\*cell\_S\_x\_un[i,j])**

**#def Adv\_y\_n(i,j):**

**# return (1/h)\*((0.5\*())\*\*2-(0.5\*())\*\*2+(0.5\*())\*(0.5\*())-(0.5\*())\*(0.5\*()))**

**def Dif\_y\_n(i,j):**

**return (1/(h\*\*2))\*(cell\_S\_y\_vn[i+1,j]+cell\_S\_y\_vn[i-1,j]+cell\_S\_y\_vn[i,j+1]+cell\_S\_y\_vn[i,j-1]-4\*cell\_S\_y\_vn[i,j])**

**def ref\_vel\_prof(x2):**

**'''**

**function returning reference analytic sol**

**'''**

**return -1200\*((x2-0.005)\*\*2)+0.03**

**epstot=100.0**

**p\_iter=0**

**#cell centroid coor**

**#the +2 stands for ghost cells on each direction**

**cell\_cent\_x=np.zeros([Nx1+2,Nx2+2])**

**cell\_cent\_y=np.zeros([Nx1+2,Nx2+2])**

**#cell\_cent\_un=np.zeros([Nx1+2,Nx2+2])**

**#cell\_cent\_us=np.zeros([Nx1+2,Nx2+2])**

**#cell\_cent\_unn=np.zeros([Nx1+2,Nx2+2])**

**cell\_cent\_pn=np.zeros([Nx1+2,Nx2+2])**

**cell\_cent\_pnn=np.zeros([Nx1+2,Nx2+2])**

**#cell corner coor**

**cell\_cor\_x=np.zeros([Nx1+3,Nx2+3])**

**cell\_cor\_y=np.zeros([Nx1+3,Nx2+3])**

**#surf area of the cell**

**cell\_S\_x=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_y=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_x\_coor\_x=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_x\_coor\_y=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_y\_coor\_x=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_y\_coor\_y=np.zeros([Nx1+2,Nx2+2])**

**#normal vector of cell surfaces**

**cell\_S\_x\_nx=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_x\_ny=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_y\_nx=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_y\_ny=np.zeros([Nx1+2,Nx2+2])**

**#surface velocities**

**cell\_S\_x\_un=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_x\_us=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_x\_unn=np.zeros([Nx1+2,Nx2+2])**

**#cell\_S\_x\_vn=np.zeros([Nx1+2,Nx2+2])**

**#cell\_S\_x\_vs=np.zeros([Nx1+2,Nx2+2])**

**#cell\_S\_x\_vnn=np.zeros([Nx1+2,Nx2+2])**

**#cell\_S\_x\_v=np.zeros([Nx1+2,Nx2+2])**

**#cell\_S\_y\_un=np.zeros([Nx1+2,Nx2+2])**

**#cell\_S\_y\_us=np.zeros([Nx1+2,Nx2+2])**

**#cell\_S\_y\_unn=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_y\_vn=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_y\_vs=np.zeros([Nx1+2,Nx2+2])**

**cell\_S\_y\_vnn=np.zeros([Nx1+2,Nx2+2])**

**#cell\_S\_y\_v=np.zeros([Nx1+2,Nx2+2])**

**#reference velocity profile**

**ref\_S\_u=np.zeros([Nx2+2])**

**L\_sq=np.array([1.0,1.0])**

**#corner coor initialization**

**for j in range(0,Nx2+3):**

**for i in range(0, Nx1+3):**

**cell\_cor\_x[i,j]=(Lx1/Nx1)\*(i-1)**

**cell\_cor\_y[i,j]=(Lx2/Nx2)\*(j-1)**

**#cell cent coor storage**

**for j in range(0, Nx2+2):**

**for i in range(0, Nx1+2):**

**cell\_cent\_x[i,j]='{:10.6e}'.format(0.25\*(cell\_cor\_x[i,j]+cell\_cor\_x[i+1,j]+cell\_cor\_x[i,j+1]+cell\_cor\_x[i+1,j+1]))**

**cell\_cent\_y[i,j]='{:10.6e}'.format(0.25\*(cell\_cor\_y[i,j]+cell\_cor\_y[i+1,j]+cell\_cor\_y[i,j+1]+cell\_cor\_y[i+1,j+1]))**

**cell\_S\_x\_coor\_x[i,j]=(cell\_cor\_x[i,j]+cell\_cor\_x[i,j+1])/2**

**cell\_S\_x\_coor\_y[i,j]=(cell\_cor\_y[i,j]+cell\_cor\_y[i,j+1])/2**

**cell\_S\_y\_coor\_x[i,j]=(cell\_cor\_x[i,j]+cell\_cor\_x[i+1,j])/2**

**cell\_S\_y\_coor\_y[i,j]=(cell\_cor\_y[i,j]+cell\_cor\_y[i+1,j])/2**

**#initial conditions**

**cell\_S\_x\_un[i,j]=0.02**

**#cell\_S\_y\_un[i,j]=0.00**

**cell\_S\_x[i,j]=abs(cell\_cor\_y[i,j]-cell\_cor\_y[i,j+1])**

**cell\_S\_y[i,j]=abs(cell\_cor\_x[i,j]-cell\_cor\_x[i+1,j])**

**cell\_S\_x\_nx[i,j]=(cell\_cor\_y[i,j+1]-cell\_cor\_y[i,j])/cell\_S\_x[i,j]**

**cell\_S\_x\_ny[i,j]=(cell\_cor\_x[i,j+1]-cell\_cor\_x[i,j])/cell\_S\_x[i,j]**

**cell\_S\_y\_nx[i,j]=(cell\_cor\_y[i+1,j]-cell\_cor\_y[i,j])/cell\_S\_y[i,j]**

**cell\_S\_y\_ny[i,j]=(cell\_cor\_x[i+1,j]-cell\_cor\_x[i,j])/cell\_S\_y[i,j]**

**plt.contourf(cell\_cent\_x[1:Nx1+1, 1:Nx2+1], cell\_cent\_y[1:Nx1+1, 1:Nx2+1], cell\_S\_x\_un[1:Nx1+1, 1:Nx2+1], 20, cmap='inferno')**

**plt.colorbar()**

**plt.show()**

**L\_sq\_r=L\_sq[1]/L\_sq[0]**

**for i in range(1, Nx2+1):**

**ref\_S\_u[i]=ref\_vel\_prof(cell\_S\_x\_coor\_y[0,i])**

**while L\_sq[1]>=1e-5:**

**L\_sq[0]=L\_sq[1]**

**#predictor step:**

**for j in range(1, Nx2+1):**

**for i in range(1, Nx1+1):**

**#cell\_S\_x\_us[i,j]=cell\_S\_x\_un[i,j]+dt\*(-Adv\_x\_n(i,j)+nu\*Dif\_x\_n(i,j))**

**cell\_S\_x\_us[i,j]=cell\_S\_x\_un[i,j]+dt\*(nu\*Dif\_x\_n(i,j)-gradP/rho)**

**#B.C. update**

**#for j in range(0, Nx2+2):**

**# cell\_S\_x\_us[0,j]=cell\_S\_x\_us[-2,j]**

**# cell\_S\_x\_us[-1,j]=cell\_S\_x\_us[1,j]**

**#for i in range(0, Nx1+2):**

**# cell\_S\_x\_us[i,0]=-cell\_S\_x\_us[i,1]**

**# cell\_S\_x\_us[i,-1]=-cell\_S\_x\_us[i,-2]**

**#corrector step, pressure iteration**

**epstot=100.0**

**while epstot>3e-5:**

**epstot=0.0**

**for j in range(2, Nx2):**

**U\_s=(rho/(dt\*(Lx1/Nx1)))\*(cell\_S\_x\_us[2,j]-cell\_S\_x\_unn[1,j]+cell\_S\_y\_vs[1,j+1]-cell\_S\_y\_vs[1,j])**

**cell\_cent\_pnn[1,j]=(cell\_vol\*U\_s-(cell\_cent\_pn[2,j]+cell\_cent\_pn[1,j+1]+cell\_cent\_pn[1,j-1]))/(-3)**

**epstot+=(cell\_cent\_pnn[1,j]-cell\_cent\_pn[1,j])\*\*2**

**cell\_cent\_pn[1,j]=cell\_cent\_pnn[1,j]**

**for i in range(2, Nx1):**

**for j in range(1,Nx2+1):**

**U\_s=(rho/(dt\*(Lx1/Nx1)))\*(cell\_S\_x\_us[i+1,j]-cell\_S\_x\_us[i,j]+cell\_S\_y\_vs[i,j+1]-cell\_S\_y\_vs[i,j])**

**cell\_cent\_pnn[i,j]=(cell\_vol\*U\_s-(cell\_cent\_pn[i+1,j]+cell\_cent\_pn[i-1,j]+cell\_cent\_pn[i,j+1]+cell\_cent\_pn[i,j-1]))/(-4)**

**#print('{:4.4e}, {:4.4e}'.format(cell\_cent\_pnn[i,j], cell\_cent\_pn[i,j]))**

**epstot+=(cell\_cent\_pnn[i,j]-cell\_cent\_pn[i,j])\*\*2**

**cell\_cent\_pn[i,j]=cell\_cent\_pnn[i,j]**

**for j in range(2, Nx2):**

**U\_s=(rho/(dt\*(Lx1/Nx1)))\*(cell\_S\_x\_unn[-1,j]-cell\_S\_x\_us[-2,j]+cell\_S\_y\_vs[-2,j+1]-cell\_S\_y\_vs[-2,j])**

**cell\_cent\_pnn[-2,j]=(cell\_vol\*U\_s-(cell\_cent\_pn[-3,j]+cell\_cent\_pn[-2,j+1]+cell\_cent\_pn[-2,j-1]))/(-3)**

**epstot+=(cell\_cent\_pnn[-2,j]-cell\_cent\_pn[-2,j])\*\*2**

**cell\_cent\_pn[-2,j]=cell\_cent\_pnn[-2,j]**

**#coroner update**

**U\_s=(rho/(dt\*(Lx1/Nx1)))\*(cell\_S\_x\_us[2,1]-cell\_S\_x\_unn[1,1]+cell\_S\_y\_vs[1,2]-cell\_S\_y\_vnn[1,1])**

**cell\_cent\_pnn[1,1]=(cell\_vol\*U\_s-(cell\_cent\_pn[2,1]+cell\_cent\_pn[1,2]))/(-2)**

**epstot+=(cell\_cent\_pnn[1,1]-cell\_cent\_pn[1,1])\*\*2**

**cell\_cent\_pn[1,1]=cell\_cent\_pnn[1,1]**

**U\_s=(rho/(dt\*(Lx1/Nx1)))\*(cell\_S\_x\_us[2,-2]-cell\_S\_x\_unn[1,-2]+cell\_S\_y\_vnn[1,-2]-cell\_S\_y\_vnn[1,-3])**

**cell\_cent\_pnn[1,-2]=(cell\_vol\*U\_s-(cell\_cent\_pn[2,-2]+cell\_cent\_pn[1,-3]))/(-2)**

**epstot+=(cell\_cent\_pnn[1,-2]-cell\_cent\_pn[1,-2])\*\*2**

**cell\_cent\_pn[1,-2]=cell\_cent\_pnn[1,-2]**

**U\_s=(rho/(dt\*(Lx1/Nx1)))\*(cell\_S\_x\_unn[-2+1,1]-cell\_S\_x\_us[-2,1]+cell\_S\_y\_vs[-2,2]-cell\_S\_y\_vnn[-2,1])**

**cell\_cent\_pnn[-2,1]=(cell\_vol\*U\_s-(cell\_cent\_pn[-2-1,1]+cell\_cent\_pn[-2,2]))/(-2)**

**epstot+=(cell\_cent\_pnn[-2,1]-cell\_cent\_pn[-2,1])\*\*2**

**cell\_cent\_pn[-2,1]=cell\_cent\_pnn[-2,1]**

**U\_s=(rho/(dt\*(Lx1/Nx1)))\*(cell\_S\_x\_unn[-2+1,-2]-cell\_S\_x\_us[-2,-2]+cell\_S\_y\_vnn[-2,-2+1]-cell\_S\_y\_vs[-2,-2])**

**cell\_cent\_pnn[-2,-2]=(cell\_vol\*U\_s-(cell\_cent\_pn[-2-1,-2]+cell\_cent\_pn[-2,-2-1]))/(-2)**

**epstot+=(cell\_cent\_pnn[-2,-2]-cell\_cent\_pn[-2,-2])\*\*2**

**cell\_cent\_pn[-2,-2]=cell\_cent\_pnn[-2,-2]**

**#print('{:4.4e}'.format(epstot))**

**#updating pressure B.C.**

**#for j in range(0, Nx2+2):**

**# cell\_cent\_pnn[0,j]=cell\_cent\_pnn[1,j]**

**# cell\_cent\_pnn[-1,j]=cell\_cent\_pnn[-2,j]**

**#for i in range(0, Nx1+2):**

**# cell\_cent\_pnn[i,0]=cell\_cent\_pnn[i,1]**

**# cell\_cent\_pnn[i,-1]=cell\_cent\_pnn[i,-2]**

**if p\_iter%500==0:**

**plt.contourf(cell\_cent\_x[1:Nx1+1, 1:Nx2+1], cell\_cent\_y[1:Nx1+1, 1:Nx2+1], cell\_cent\_pnn[1:Nx1+1, 1:Nx2+1], 20, cmap='jet')**

**plt.colorbar()**

**plt.xlabel('$x\_1$ (m)')**

**plt.ylabel('$x\_2$ (m)')**

**plt.title('domain $p^{n+1}$ contour ($m/s$)')**

**plt.savefig('hw4\_4\_50\_init\_002\_p\_contour.png')**

**plt.show()**

**print('eps\_tot= {:5.4e}'.format(epstot))**

**p\_iter+=1**

**#print('eps\_tot= {:5.4e}'.format(epstot))**

**#corrector step:**

**for j in range(1, Nx2+1):**

**for i in range(1, Nx1+1):**

**cell\_S\_x\_unn[i,j]=cell\_S\_x\_us[i,j]-(1/rho)\*(dt)\*(cell\_cent\_pnn[i,j]-cell\_cent\_pnn[i-1,j])**

**#B.C. update**

**for j in range(0, Nx2+2):**

**cell\_S\_x\_unn[0,j]=cell\_S\_x\_unn[-2,j]**

**cell\_S\_x\_unn[-1,j]=cell\_S\_x\_unn[1,j]**

**for i in range(0, Nx1+2):**

**cell\_S\_x\_unn[i,0]=-cell\_S\_x\_unn[i,1]**

**cell\_S\_x\_unn[i,-1]=-cell\_S\_x\_unn[i,-2]**

**for j in range(1, Nx2+1):**

**for i in range(1, Nx1+1):**

**cell\_S\_x\_un[i,j]=cell\_S\_x\_unn[i,j]**

**for j in range(0, Nx2+2):**

**cell\_S\_x\_un[0,j]=cell\_S\_x\_un[-2,j]**

**cell\_S\_x\_un[-1,j]=cell\_S\_x\_un[1,j]**

**for i in range(0, Nx1+2):**

**cell\_S\_x\_un[i,0]=-cell\_S\_x\_un[i,1]**

**cell\_S\_x\_un[i,-1]=-cell\_S\_x\_un[i,-2]**

**sq\_sum\_error=0**

**for i in range(1,Nx2+1):**

**sq\_sum\_error+=(ref\_S\_u[i]-cell\_S\_x\_un[50,i])\*\*2**

**L\_sq[1]=math.sqrt(sq\_sum\_error/(Nx2+1))**

**if n\_iter%500==0:**

**print('iter= '+str(n\_iter)+', L\_sq= {:.4e}'.format(L\_sq[0]))**

**plt.plot(cell\_S\_x\_un[50,1:Nx2+1],cell\_S\_x\_coor\_y[50,1:Nx2+1], color='navy', label='numerical sol, $L^2$= {:10.4e}'.format(L\_sq[0]))**

**plt.plot(ref\_S\_u[1:Nx2+1] ,cell\_S\_x\_coor\_y[50,1:Nx2+1], color='red', label='reference')**

**plt.xlabel('$u\_1$ ($m/s$)')**

**plt.ylabel('$x\_2$ (m)')**

**plt.legend()**

**plt.grid()**

**plt.savefig('hw4\_4\_50\_init\_002\_v\_profile.png')**

**plt.show()**

**plt.contourf(cell\_cent\_x[1:Nx1+1, 1:Nx2+1], cell\_cent\_y[1:Nx1+1, 1:Nx2+1], cell\_S\_x\_un[1:Nx1+1, 1:Nx2+1], 20, cmap='inferno')**

**plt.colorbar()**

**plt.xlabel('$x\_1$ (m)')**

**plt.ylabel('$x\_2$ (m)')**

**plt.title('domain $u\_1$ contour ($m/s$)')**

**plt.savefig('hw4\_4\_50\_init\_002\_v\_contour.png')**

**plt.show()**

**print('{:10d}, {:5.7e}'.format(n\_iter, L\_sq[1]))**

**L\_sq\_r=L\_sq[1]/L\_sq[0]**

**n\_iter+=1**

**print('iter= '+str(n\_iter)+', L\_sq= {:.4e}'.format(L\_sq[0]))**

**plt.plot(cell\_S\_x\_un[50,1:Nx2+1],cell\_S\_x\_coor\_y[50,1:Nx2+1], color='navy', label='numerical sol, $L^2$= {:10.4e}'.format(L\_sq[0]))**

**plt.plot(ref\_S\_u[1:Nx2+1] ,cell\_S\_x\_coor\_y[50,1:Nx2+1], color='red', label='reference')**

**plt.xlabel('$u\_1$ ($m/s$)')**

**plt.ylabel('$x\_2$ (m)')**

**plt.legend()**

**plt.savefig('hw4\_4\_50\_init\_002\_v\_profile.png')**

**plt.grid()**

**plt.show()**

**plt.contourf(cell\_cent\_x[1:Nx1+1, 1:Nx2+1], cell\_cent\_y[1:Nx1+1, 1:Nx2+1], cell\_S\_x\_un[1:Nx1+1, 1:Nx2+1], 20, cmap='inferno')**

**plt.colorbar()**

**plt.xlabel('$x\_1$ (m)')**

**plt.ylabel('$x\_2$ (m)')**

**plt.title('domain $u\_1$ contour ($m/s$)')**

**plt.savefig('hw4\_4\_50\_init\_002\_v\_contour.png')**

**plt.show()**