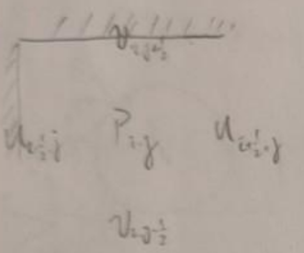
NE577. Hwk 4.

Name: Chang Kai Tai
StuID. 200328541

4.1

$$\frac{P^{n+1}_{i+1,j}+P^{n+1}_{i-1,j}+P^{n+1}_{i,j+1}+P^{n+1}_{i,j-1}-4P^{n+1}_{i,j}}{h^2}=\frac{\rho}{\Delta t}\left(\frac{u^*_{i+\frac12,j}-u^*_{i-\frac12,j}+v^*_{i,j+\frac12}-v^*_{i,j-\frac12}}{h}\right)$$

staggered grid at corner:



$$\begin{cases} u^{n+1}_{i+\frac12,j}=u^*_{i+\frac12,j}-\frac{1}{\rho}\frac{\Delta t}{h}\left(P^{n+1}_{i+1,j}-P^{n+1}_{i,j}\right)\\ v^{n+1}_{i,j+\frac12}=v^*_{i,j+\frac12}-\frac{1}{\rho}\frac{\Delta t}{h}\left(P^{n+1}_{i,j+1}-P^{n+1}_{i,j}\right)\end{cases}$$

insert to eqn above

$$\text{B.C.}:\begin{cases} u_{i-\frac12,j}=u_{b,j}\\ v_{i,j+\frac12}=v_{i,b}\end{cases}$$

$$\frac{P^{n+1}_{i+1,j}+P^{n+1}_{i-1,j}+P^{n+1}_{i,j+1}+P^{n+1}_{i,j-1}-4P^{n+1}_{i,j}}{h^2}=\frac{\rho}{\Delta t h}\left[\left(u^{n+1}_{i+\frac12,j}+\frac{\Delta t}{\rho h}\left(P^{n+1}_{i+1,j}-P^{n+1}_{i,j}\right)\right)-u^*_{i-\frac12,j}+v^*_{i,j+\frac12}-v^{n+1}_{i,j-\frac12}-\frac{\Delta t}{\rho h}\left(P^{n+1}_{i,j}-P^{n+1}_{i,j-1}\right)\right]$$

$$=\frac{\rho}{\Delta t h}\left(u^{n+1}_{i+\frac12,j}-u^*_{i-\frac12,j}+v^*_{i,j+\frac12}+v^{n+1}_{i,j-\frac12}\right)+\frac{P^{n+1}_{i+1,j}+P^{n+1}_{i,j-1}-2P^{n+1}_{i,j}}{h^2}$$

rearrange

$$\frac{P^{n+1}_{i+1,j}+P^{n+1}_{i,j+1}-2P^{n+1}_{i,j}}{h^2}=\frac{\rho}{\Delta t}\left(\frac{u^{n+1}_{i+\frac12,j}-u^*_{i-\frac12,j}+v^*_{i,j+\frac12}-v^{n+1}_{i,j-\frac12}}{h}\right)\quad\text{with}\begin{cases}u_{i-\frac12,j}=u_{b,j}\\ v_{i,j+\frac12}=v_{i,b}\end{cases}$$

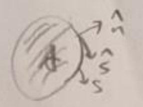
$$\frac{P^{n+1}_{i+1,j}+P^{n+1}_{i,j+1}-2P^{n+1}_{i,j}}{h^2}=\frac{\rho}{\Delta t}\left(\frac{u^{n+1}_{i+\frac12,j}-u_{b,j}+v_{i,b}-v^*_{i,j-\frac12}}{h}\right)\quad \text{B.C.}$$

---

*

#4.2.

(a) 2D Heaviside fn: $H(x,y) = \int \delta(x-x')\,\delta(y-y')\,dA$

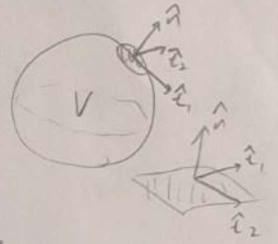


$$\nabla H = \int_A \nabla\, \delta(x-x')\delta(y-y')\,dA = -\int_A \hat\nabla\, \delta(x-x')\,\delta(y-y')\,dA$$ with identity: $\int_A \nabla\phi\, dA = \oint_S \phi\, \hat n\, ds$

$$= -\oint_S \delta(x-x')\delta(y-y')\,\hat n'\, ds'$$ with coordinate transform, $\delta(x-x')\delta(y-y') = \delta(s)\,\delta(\hat n)$

$$= -\oint_S \delta(\hat n')\,\delta(s)\,\hat n'\, ds' = \underline{-\delta(\hat n)\,\hat n}$$

(b) 3D Heaviside fn: $H(x,y,z) = \int_V \delta(x-x')\,\delta(y-y')\,\delta(z-z')\,dV$



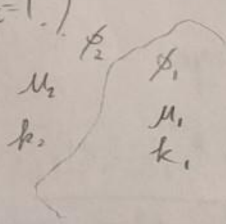
$$\nabla H(x,y,z) = \int_V \nabla\, \delta(x-x')\delta(y-y')\,\delta(z-z')\,dV = -\int_V \hat\nabla\, \delta(x-x')\,\delta(y-y')\,\delta(z-z')\,dv.$$

Likewise, $\int_V \nabla\phi\, dv = \oint_S \phi\, \hat n\, ds$.

$$\nabla H = -\oint_S \delta(x-x')\,\delta(y-y')\,\delta(z-z')\,\hat n'\, dS'$$ transform coor into $\hat n, \hat t_1, \hat t_2$.

$$= -\oint_S \delta(\hat n')\,\delta(\hat t_1')\,\delta(\hat t_2')\,\hat n'\, d\hat t_1'\, d\hat t_2' = -\delta(\hat n)\,\hat n.$$ (w/ $\int \delta(\hat t_i)\, d\hat t_i = 1$.)

(c) Assume μ, k are constant in each phase:



$$\mu(x,y,z) = \mu_1 H(x,y,z) + \mu_2\,(1 - H(x,y,z))$$

$$\nabla\mu = \nabla(\mu_1 H(x,y,z)) + \nabla(\mu_2\,(1 - H(x,y,z)))$$

$$= \mu_1 \nabla H - \mu_2 \nabla H = (\mu_1 - \mu_2)\,\nabla H = -(\mu_1 - \mu_2)\,\delta(\hat n)\,\hat n = \underline{\Delta\mu\,\delta(\hat n)\,\delta\hat n}$$

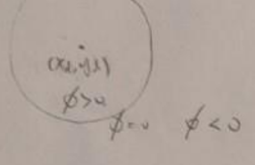
$k = k_1 H(x,y,z) + k_2 (1 - H(x,y,z))$

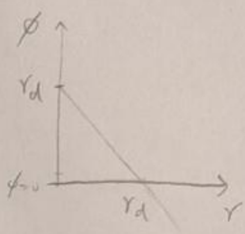
$\nabla k = k_1 \nabla H - k_2 \nabla H = (k_1 - k_2)\nabla H = (k_2 - k_1)\, \delta(\lambda)\hat{n} = \Delta k\, \delta(\lambda)\hat{n}$

#4.3

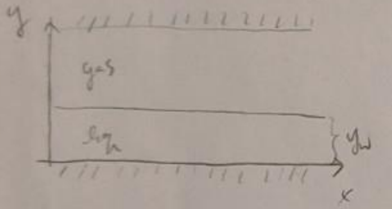
(a) single circular liquid drop w/ radius r_d, center @ (x_d, y_d)

$$\phi(x,y) = -\left[\sqrt{(x-x_d)^2 + (y-y_d)^2} - r_d\right]$$

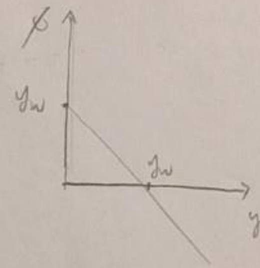




(b) horizontal stratified flow with liquid thickness y_w:



$$\phi(x,y) = -(y - y_w)$$



(c)

continue from result of (b), now adds N_d droplets, N_b bubbles into the stratified flow domain, then the $\phi(x,y)$ can be initialized as:

if $y > y_w$ (continuous: gas, dispersed: liquid)

$$\phi(x,y) = \max\left(-(y-y_w),\ -\left[\sqrt{(x-x_{d1})^2 + (y-y_{d1})^2} - r_{d1}\right],\ -\left[\sqrt{(x-x_{d2})^2 + (y-y_{d2})^2} - r_{d2}\right], \ldots, -\left[\sqrt{(x-x_{dN_d})^2 + (y-y_{dN_d})^2} - r_{dN_d}\right]\right)$$

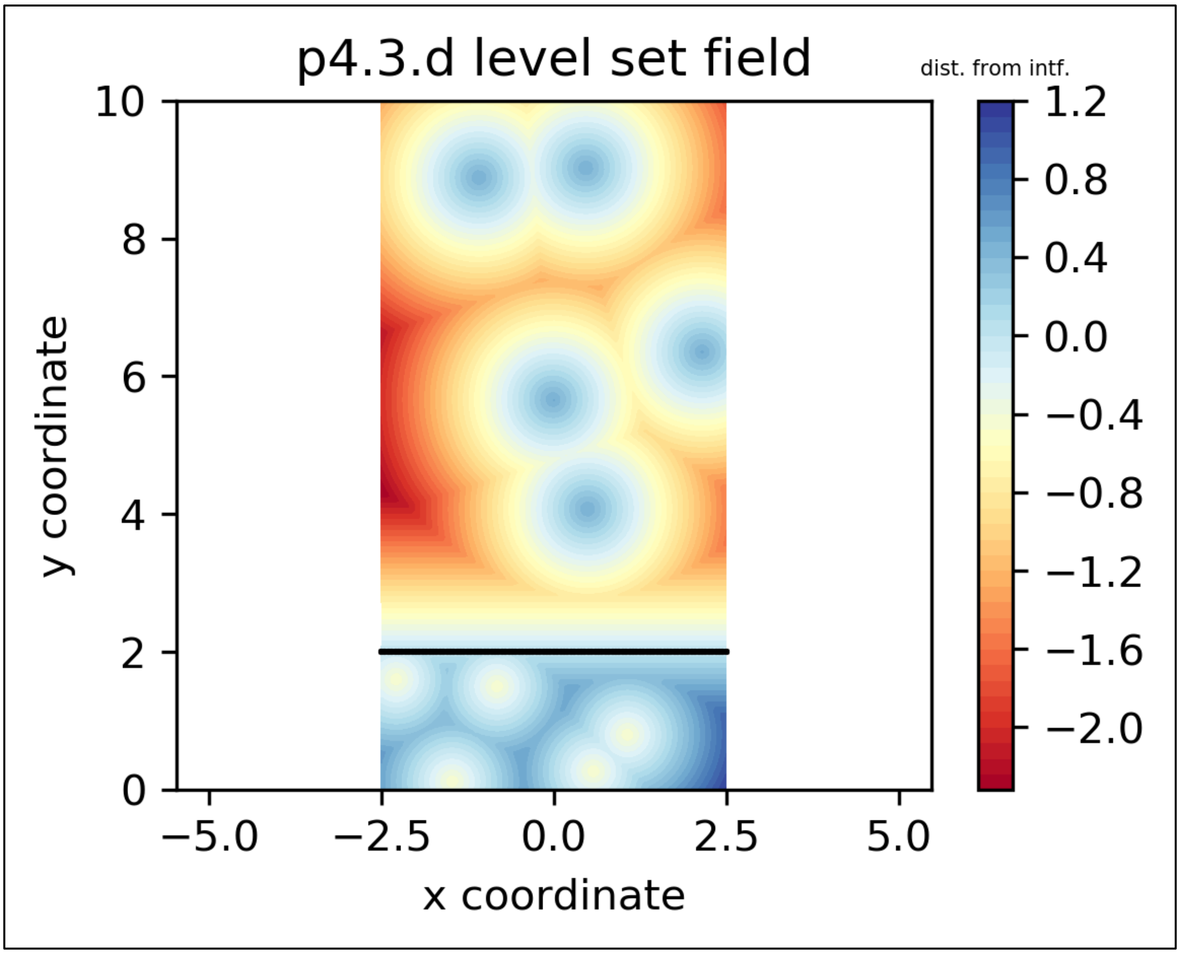
droplets in gas phase

else ($y \le y_w$)

$$\phi(x,y) = \min\left(-(y-y_w),\ \sqrt{(x-x_{b1})^2 + (y-y_{b1})^2} - r_{b1},\ \ldots\ \sqrt{(x-x_{bN_b})^2 + (y-y_{bN_b})^2} - r_{bN_b}\right)$$

bubbles in liquid phase.

4.3 (d)



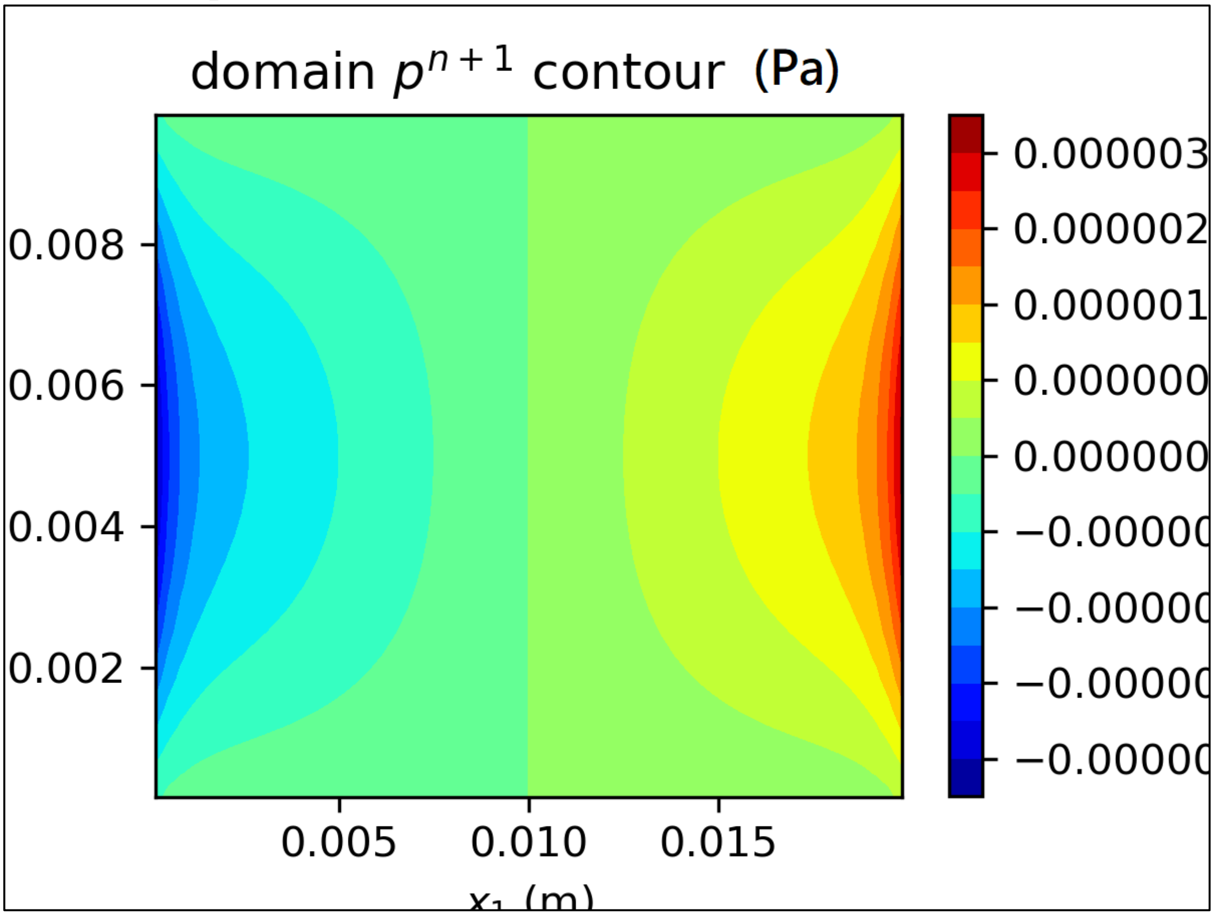
The code generating the level-set field is attached to this file. The water level is marked using the solid black line at y=2. Five bubbles and droplets are generated within liquid and gas domain respectively with randomly selected center coordinate. Radius of bubbles/droplets are assigned to be $\frac{10}{20} = 0.5$.

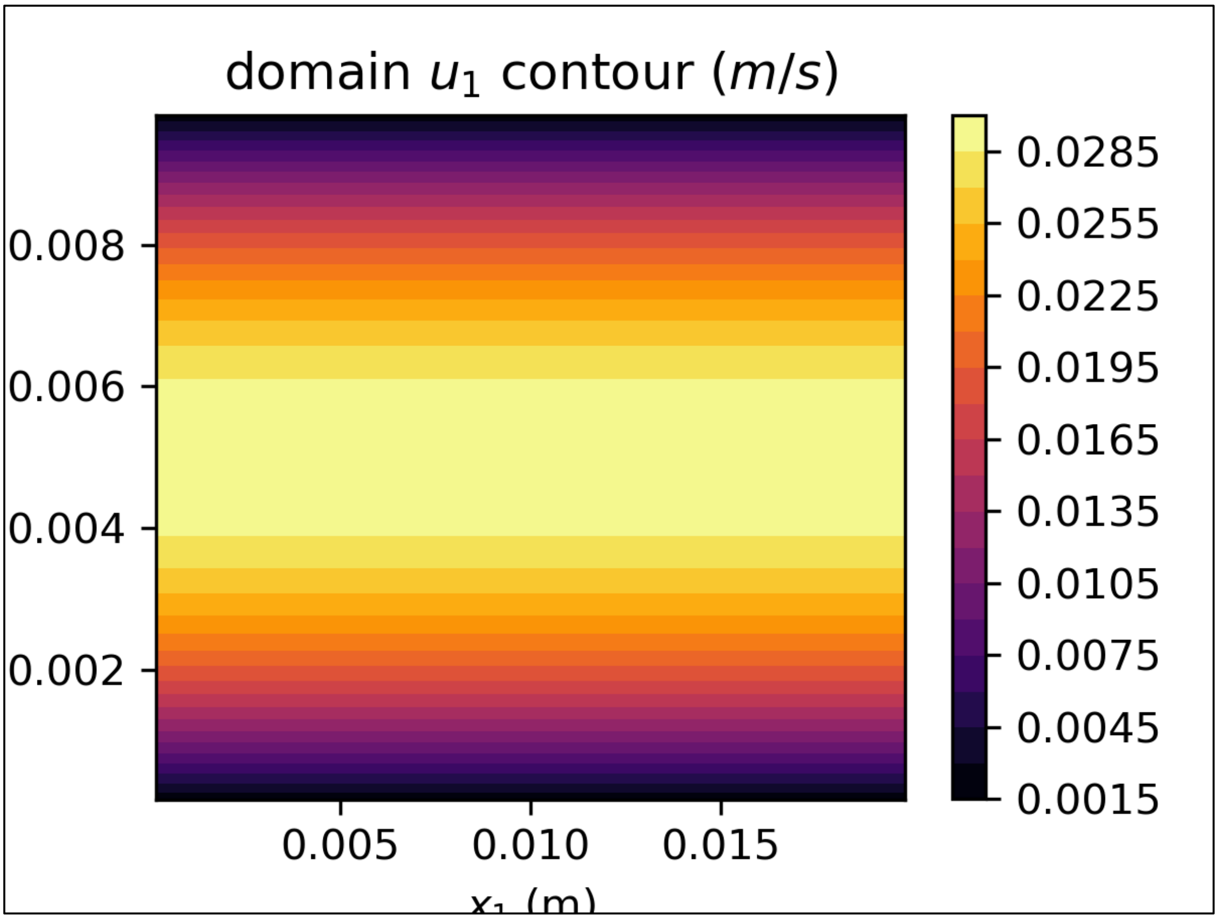
The resulting level-set field is believed to be correct since it properly shows the feature that level-set values on the interfaces are 0. Then in the continuous phase, level-set value is positive/negative at liquid/gas. As for the dispersed phase, ϕ is negative/positive in bubbles/droplets with minimum/maximum value $-0.5/0.5$ respectively. Therefore, the level-set field shows the features as required.

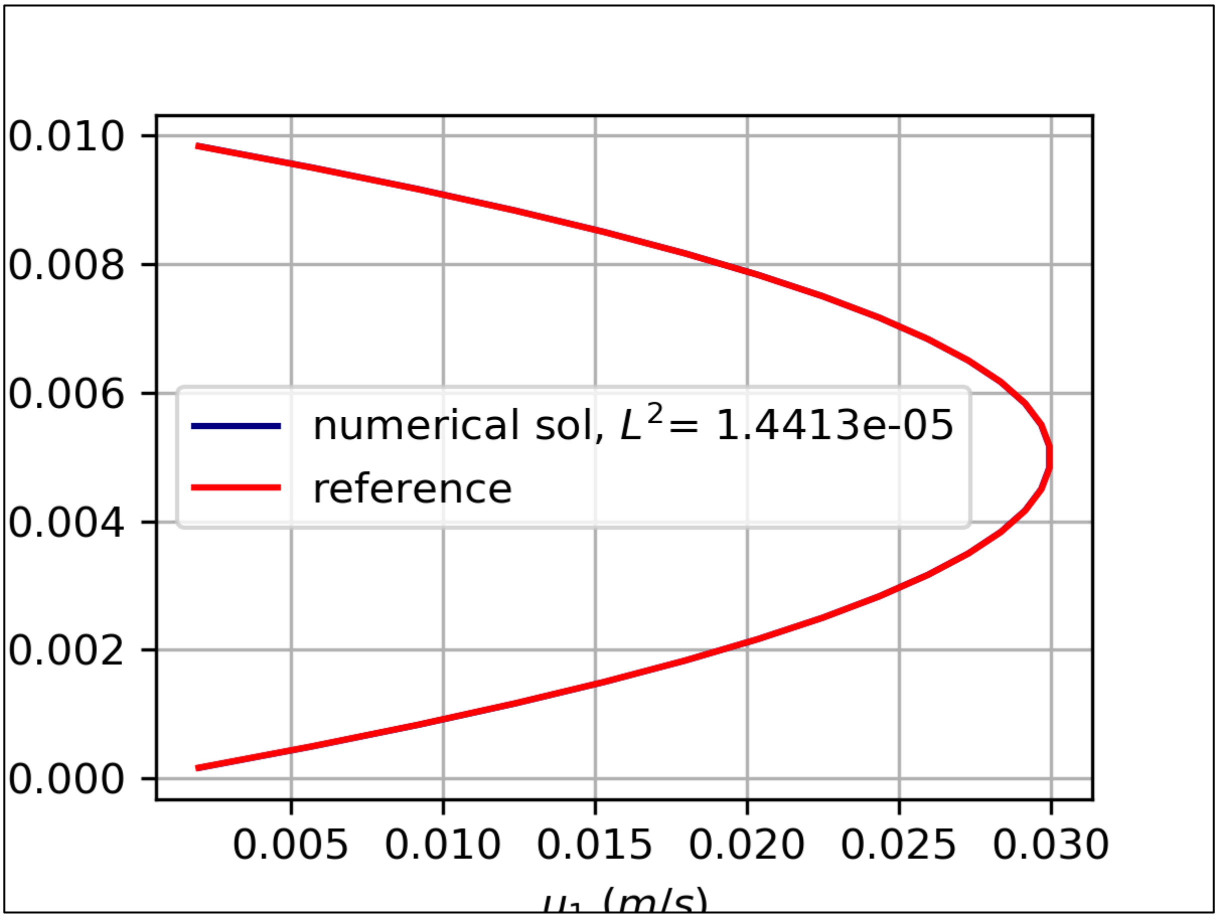
4.4 (a)

All initiated with uniform $v = 0.02\frac{m}{s}$. Code is attached behind.

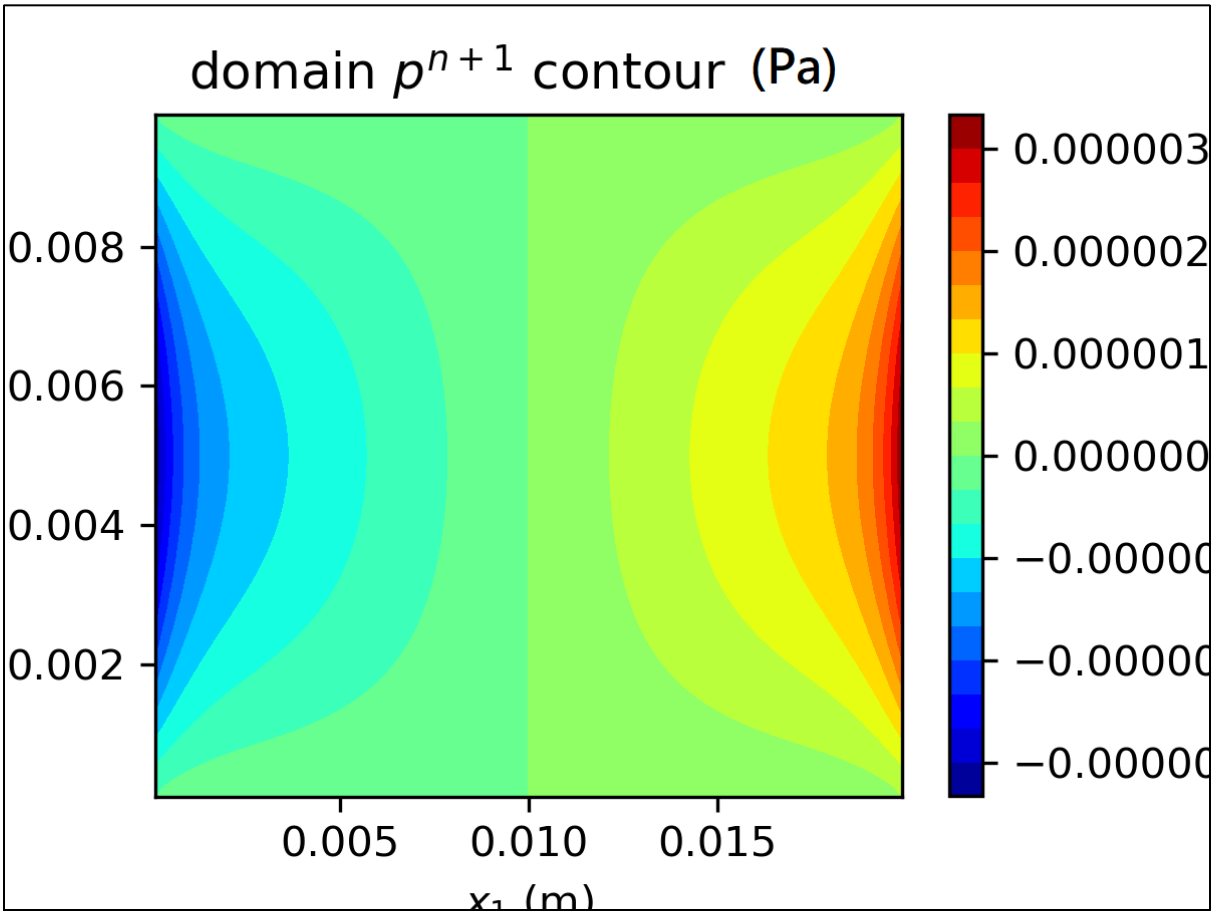
1. Domain with 30 pressure cells

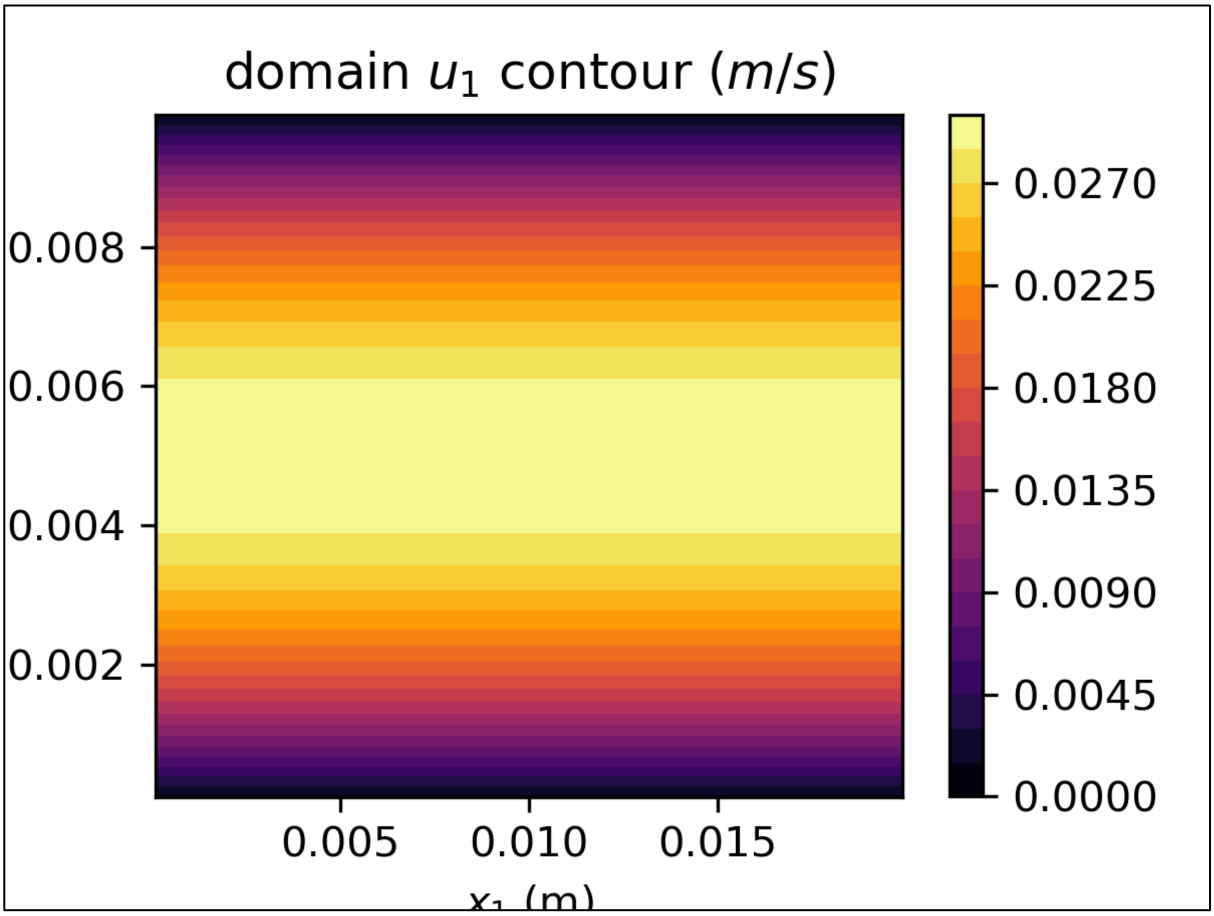


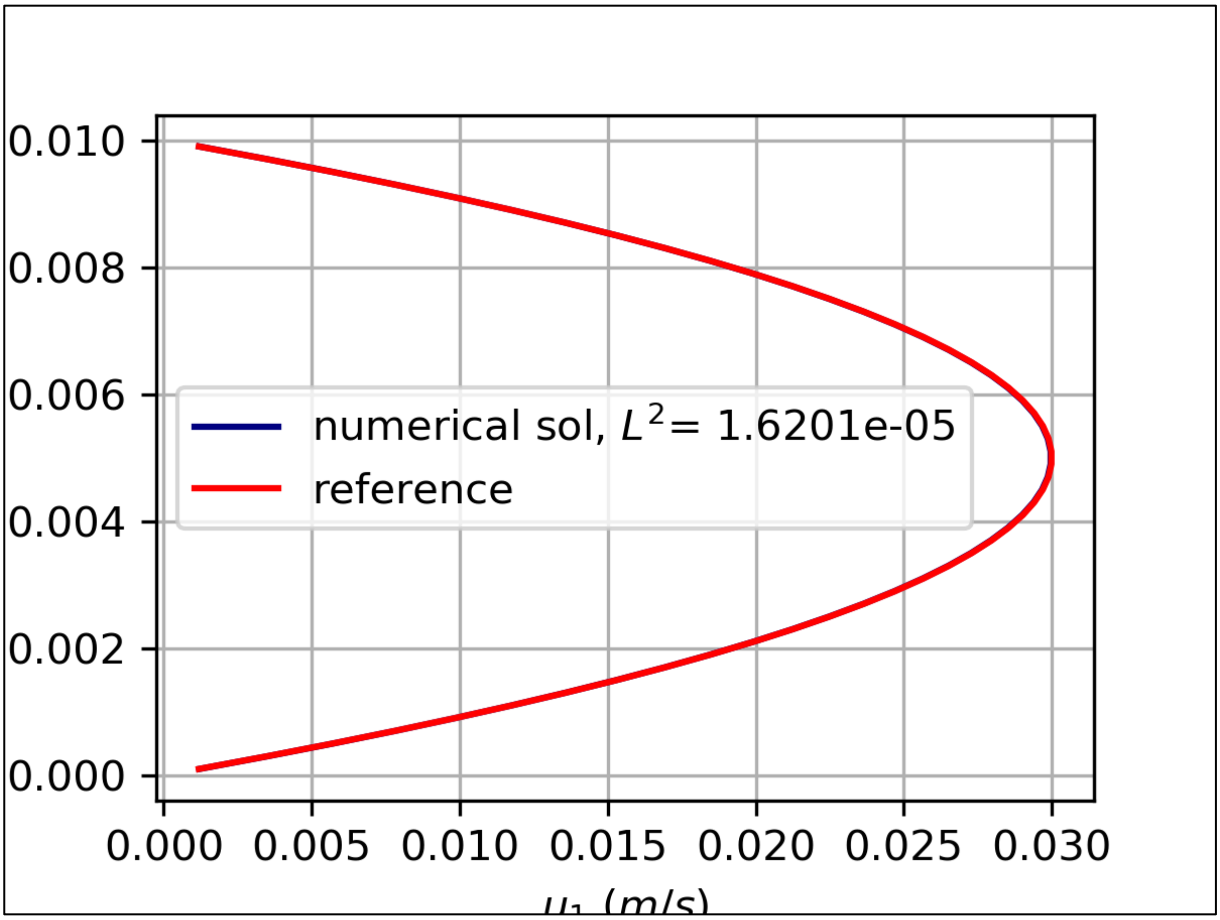




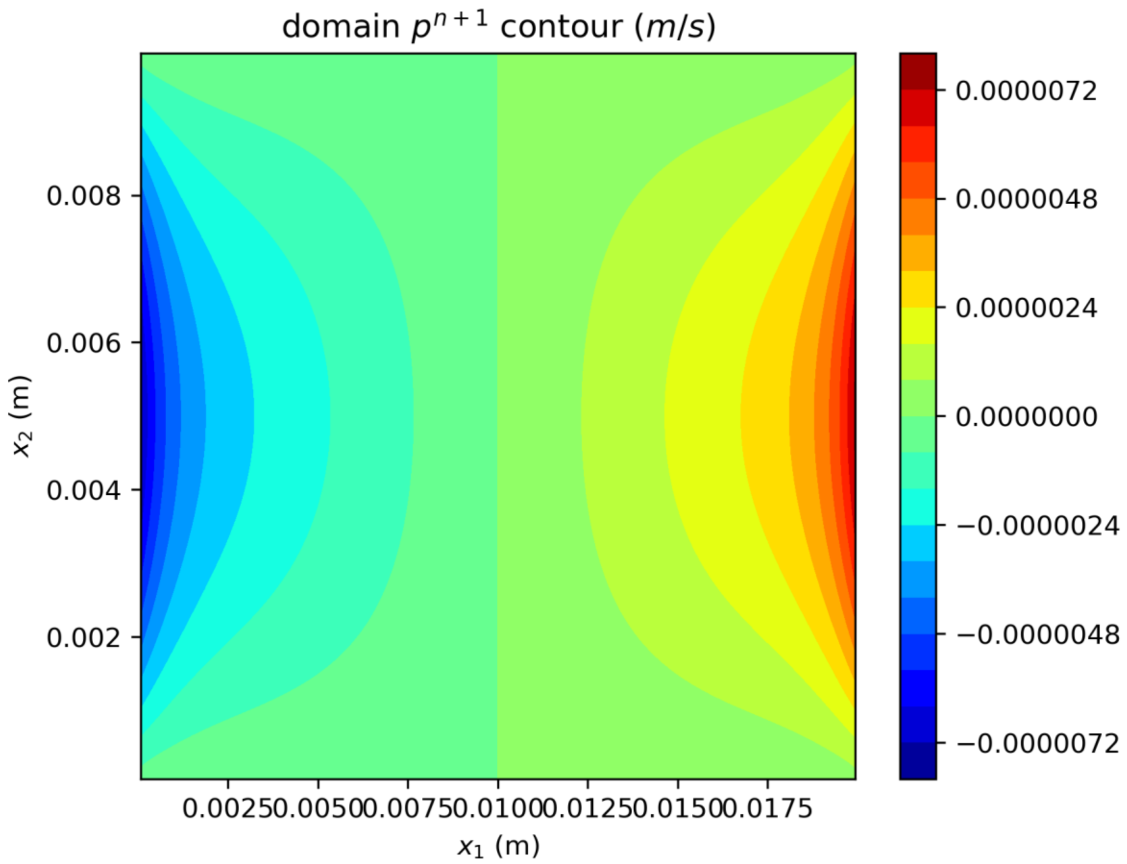
2. Domain with 50 pressure cells

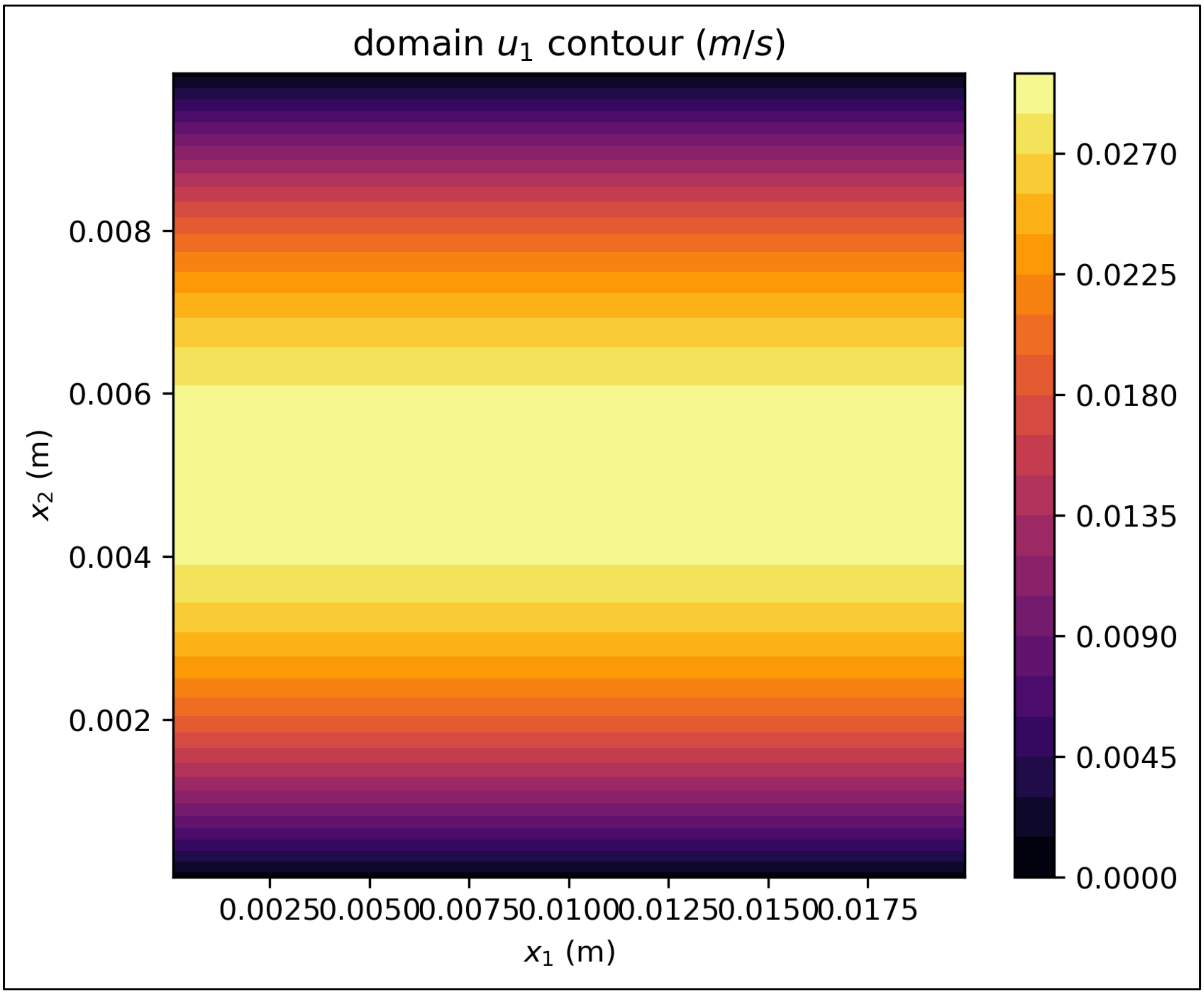


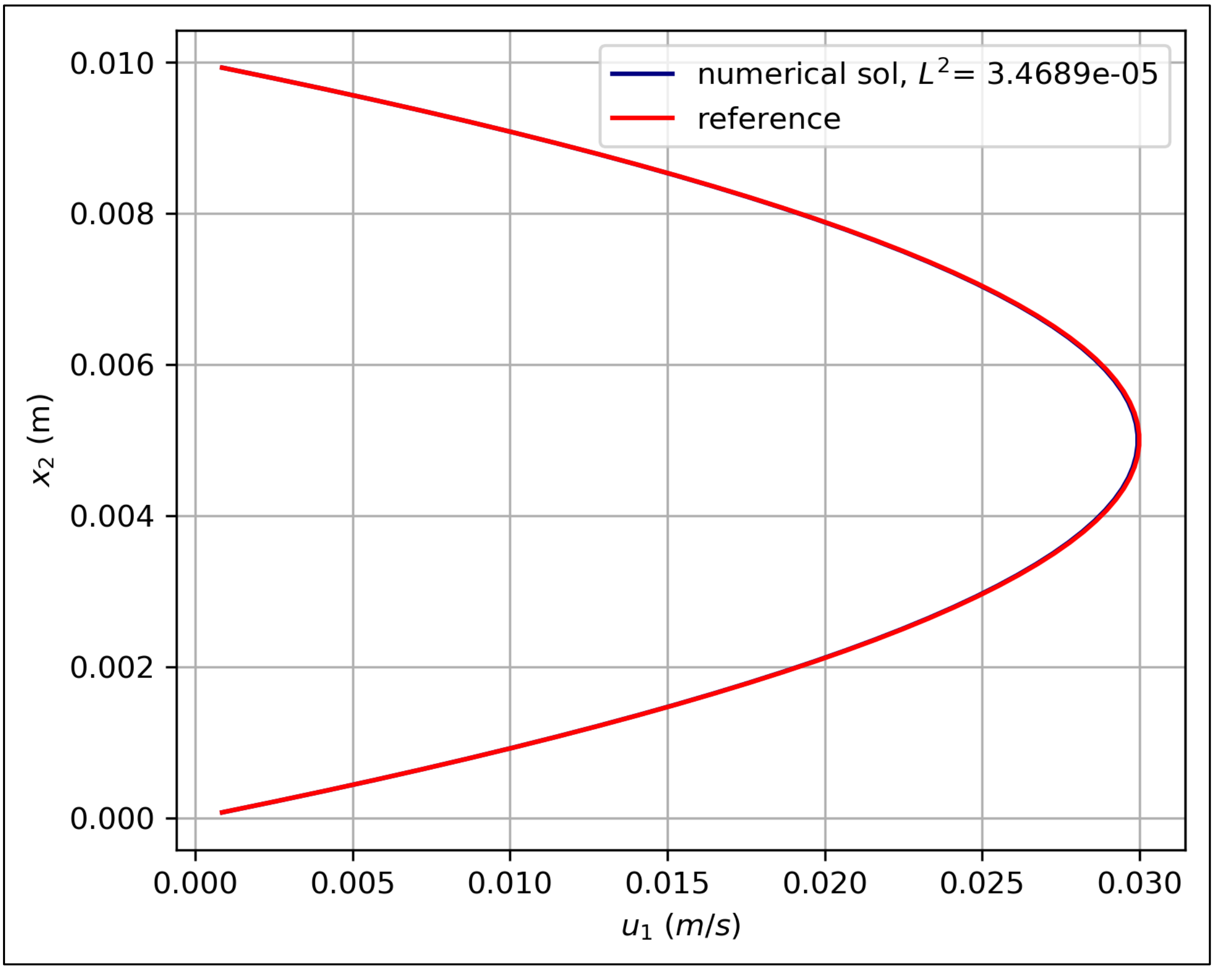

domain u_1 contour (m/s)
0.008
0.006
0.004
0.002
0.005
0.010
0.015
x_1 (m)
0.0270
0.0225
0.0180
0.0135
0.0090
0.0045
0.0000


0.010
0.008
0.006
0.004
0.002
0.000
0.000
0.005
0.010
0.015
0.020
0.025
0.030
u_1 (m/s)
numerical sol, L^2= 1.6201e-05
reference

3. Domain with 70 pressure cells

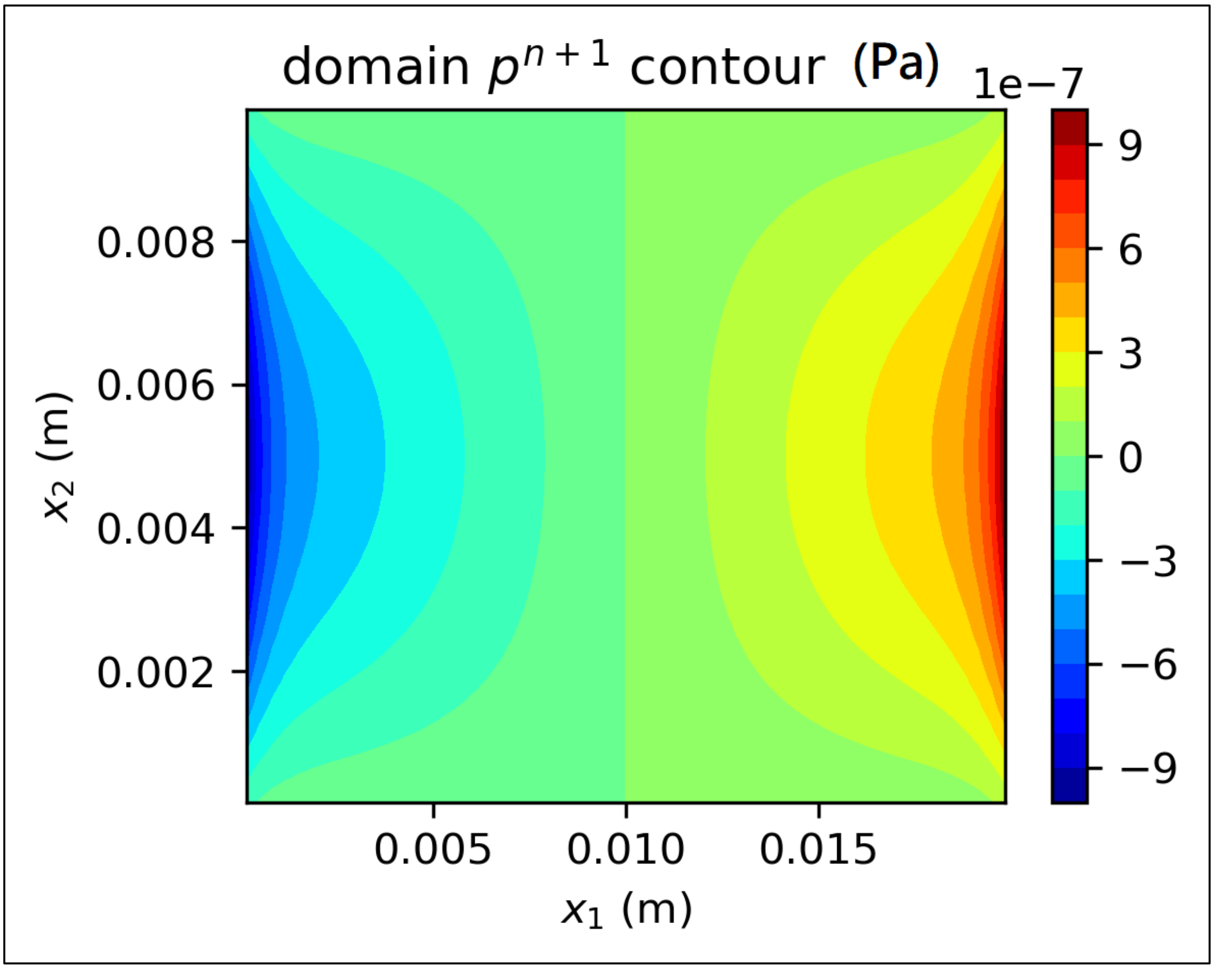




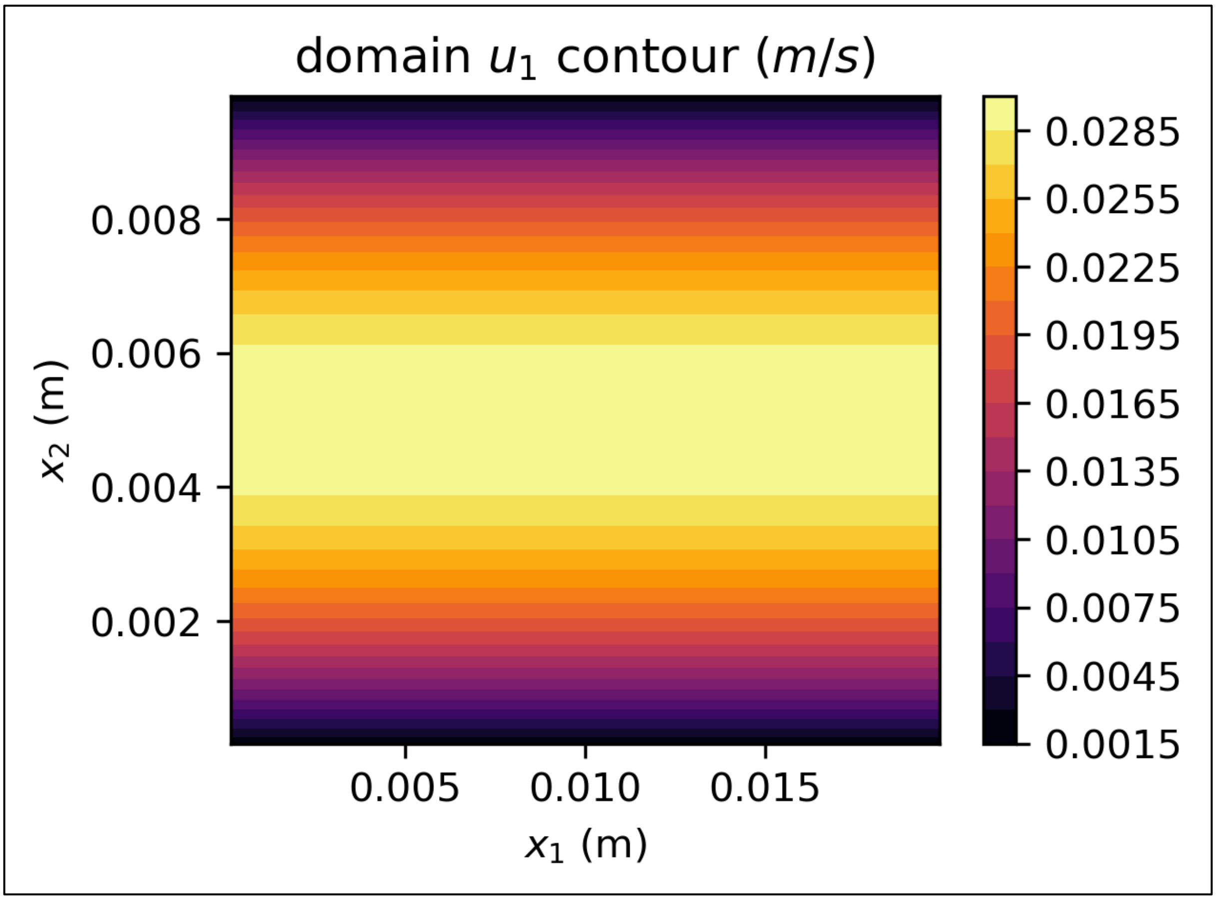


All cases eventually converged towards the reference parabolic velocity profile, with L^2 metric at order of 10^{-5}. The resulting pressure field is hourglass shaped with largest pressure gradient occurred at channel center level, where the maximum velocity appears. However, the result of pressure field is somewhat inconsistent with the analytical solution derived in previous problem 3.4, where the ∇P ought to be $-2.4\frac{Pa}{m}$. 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.

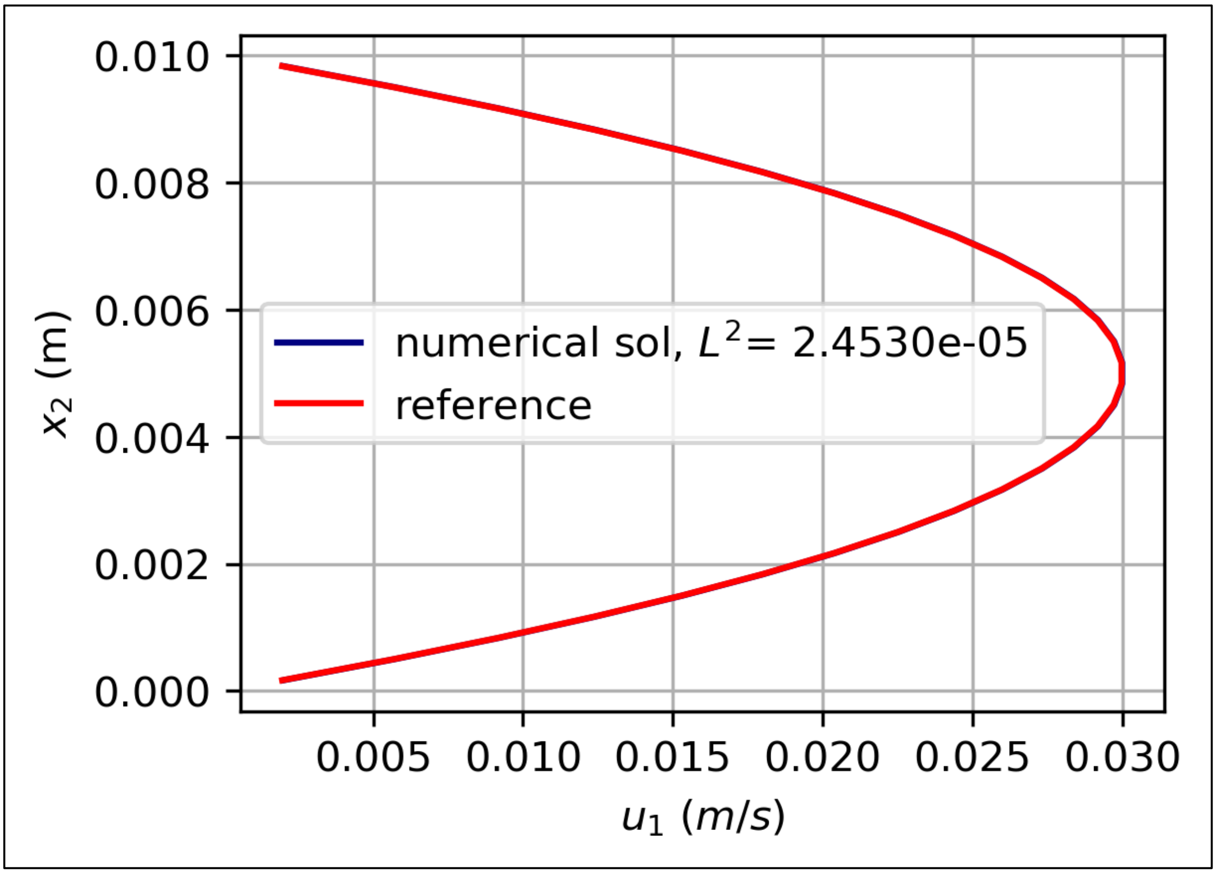
4.4 (b)



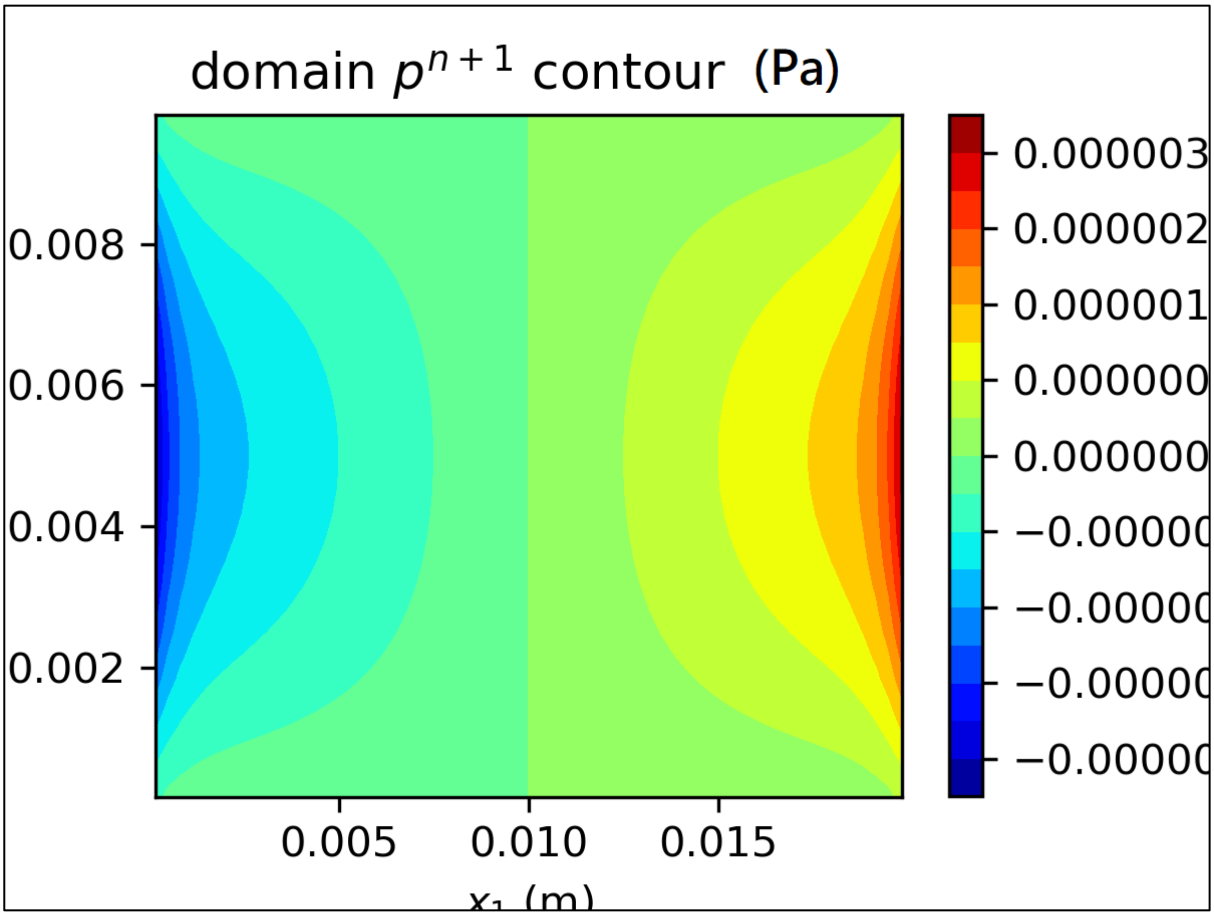
domain p^{n+1} contour (Pa)
1e−7
9
6
3
0
−3
−6
−9
0.008
0.006
0.004
0.002
x_2 (m)
0.005
0.010
0.015
x_1 (m)



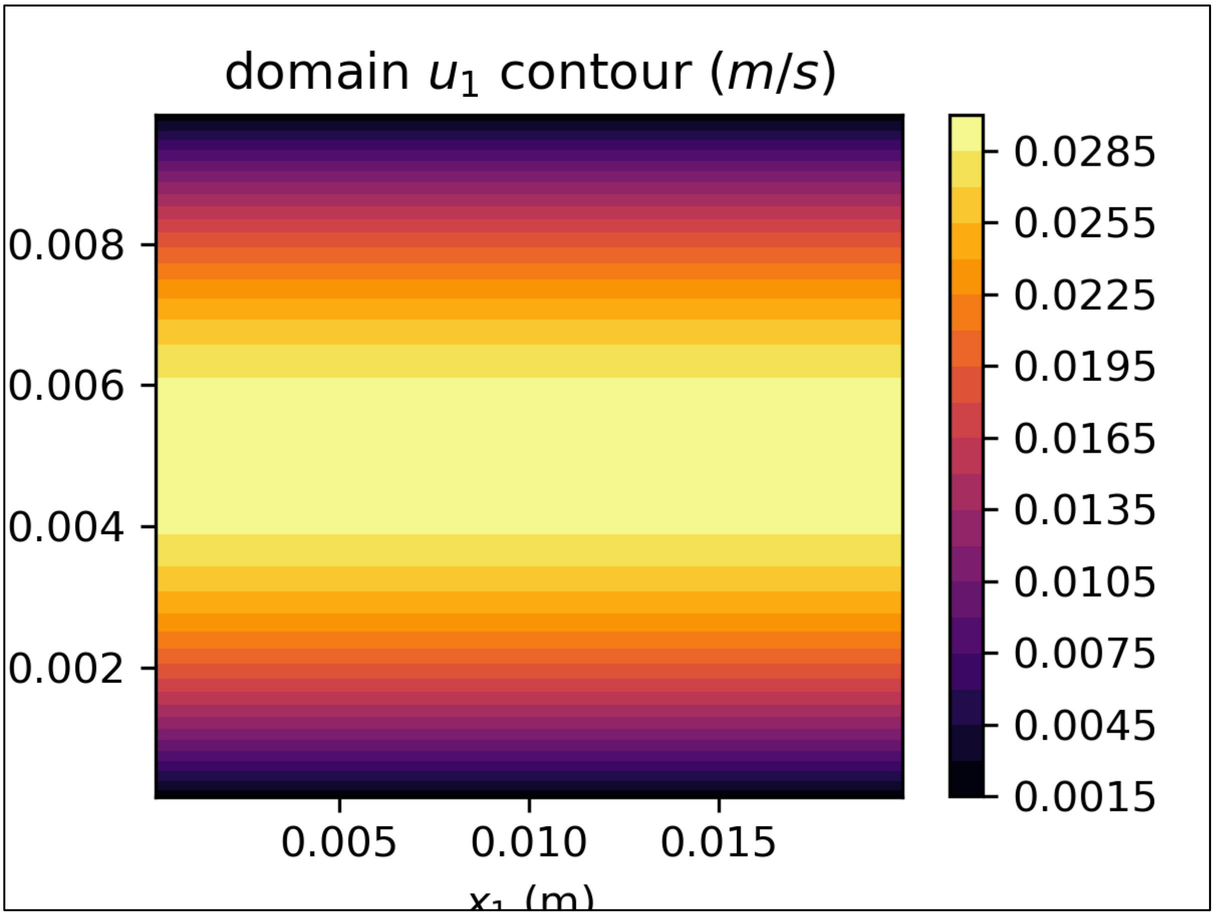
domain u_1 contour (m/s)
0.0285
0.0255
0.0225
0.0195
0.0165
0.0135
0.0105
0.0075
0.0045
0.0015
0.008
0.006
0.004
0.002
x_2 (m)
0.005
0.010
0.015
x_1 (m)

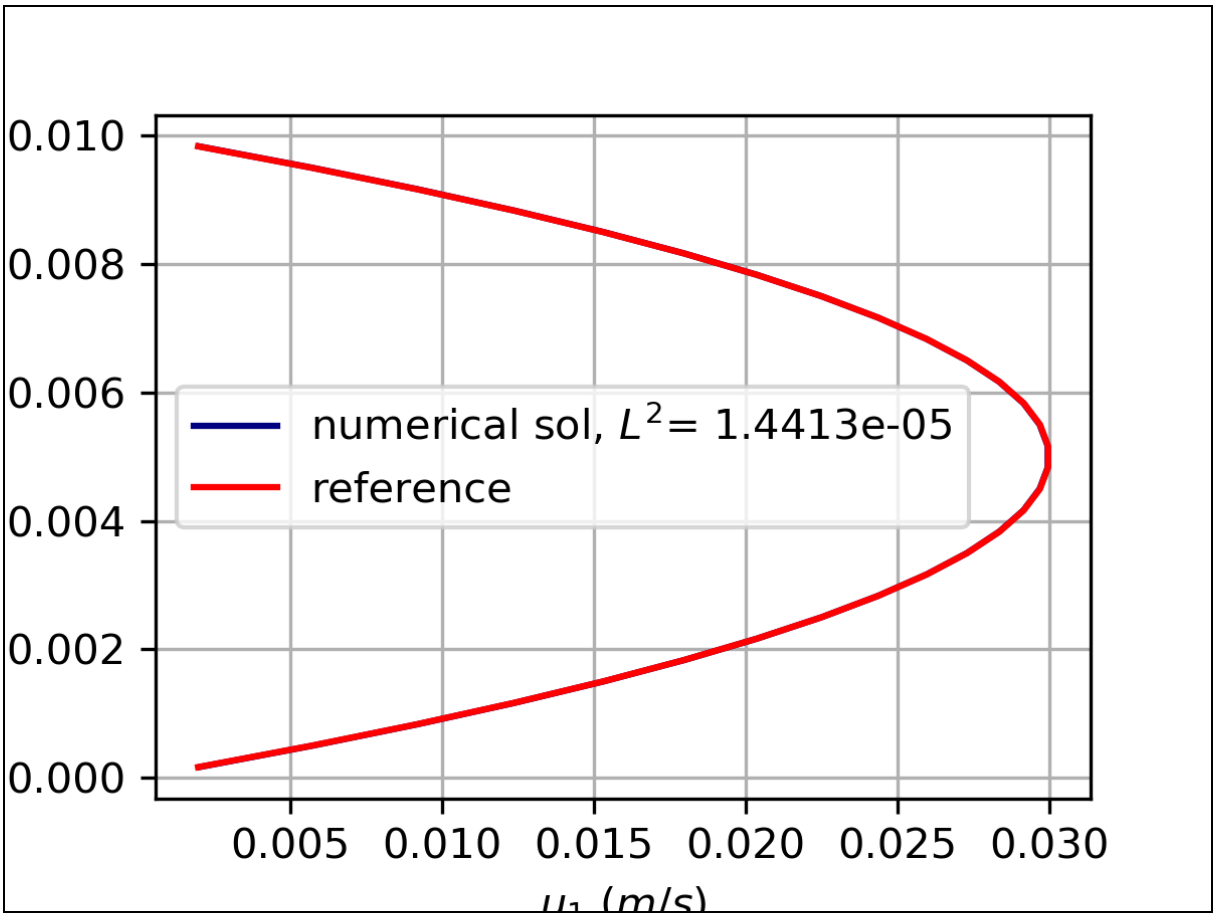


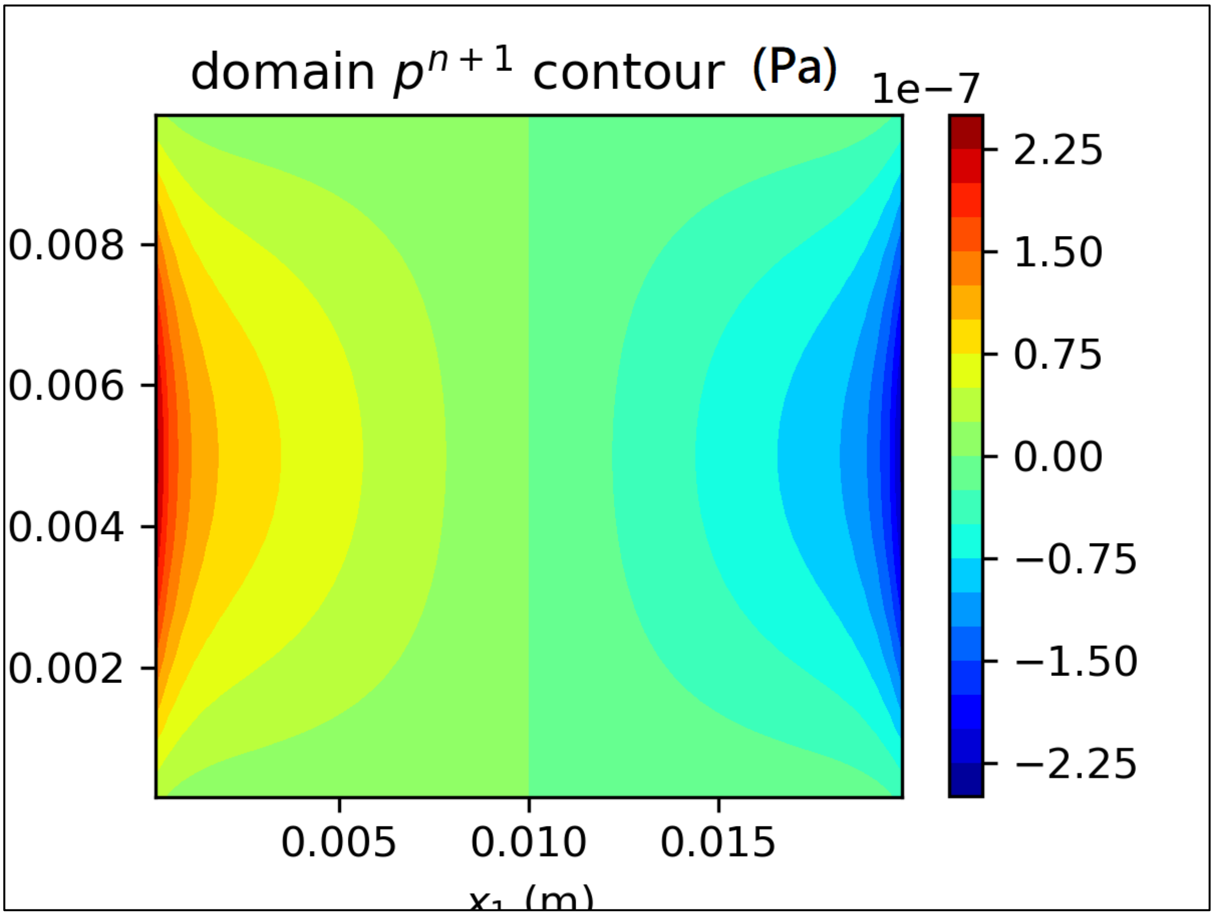
0.010
0.008
0.006
0.004
0.002
0.000
x_2 (m)
numerical sol, L^2= 2.4530e-05
reference
0.005
0.010
0.015
0.020
0.025
0.030
u_1 (m/s)

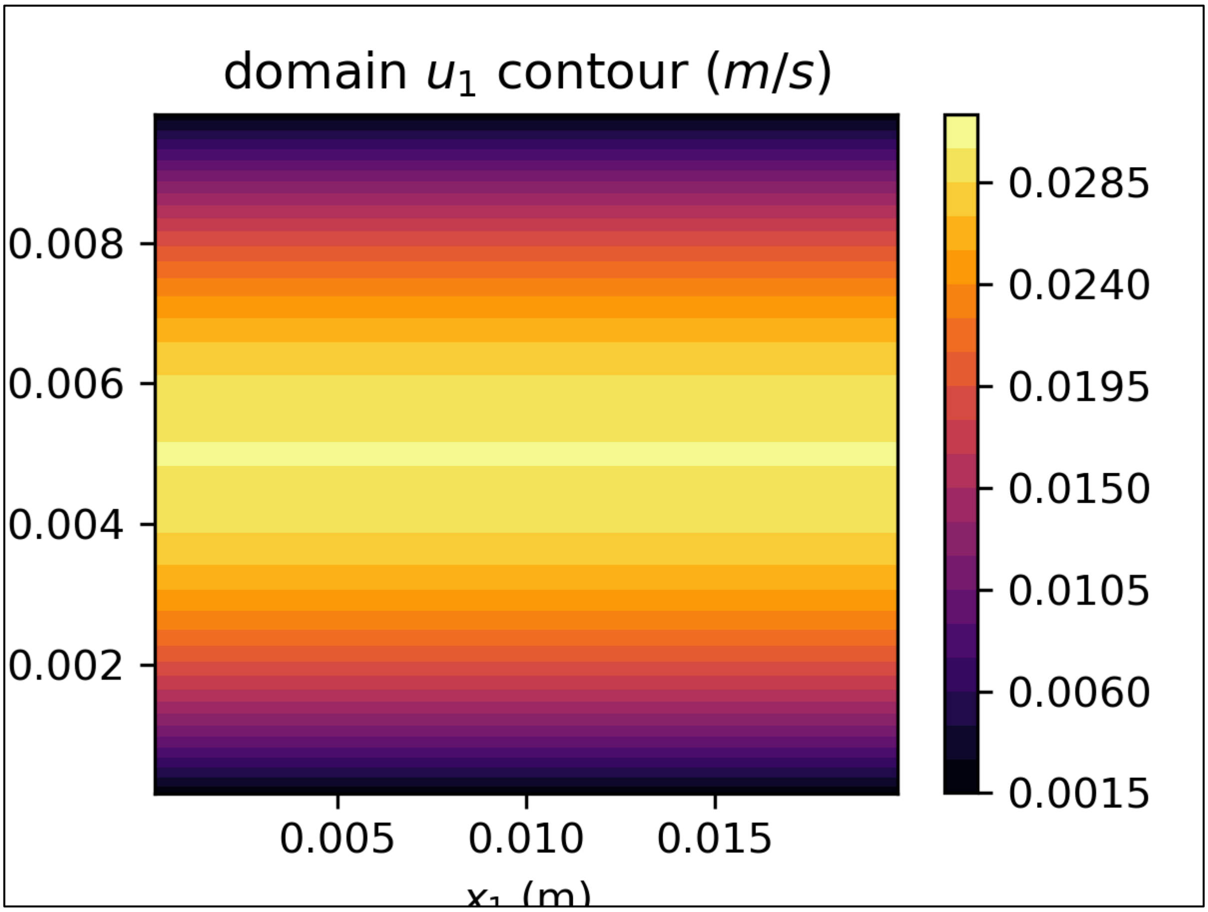


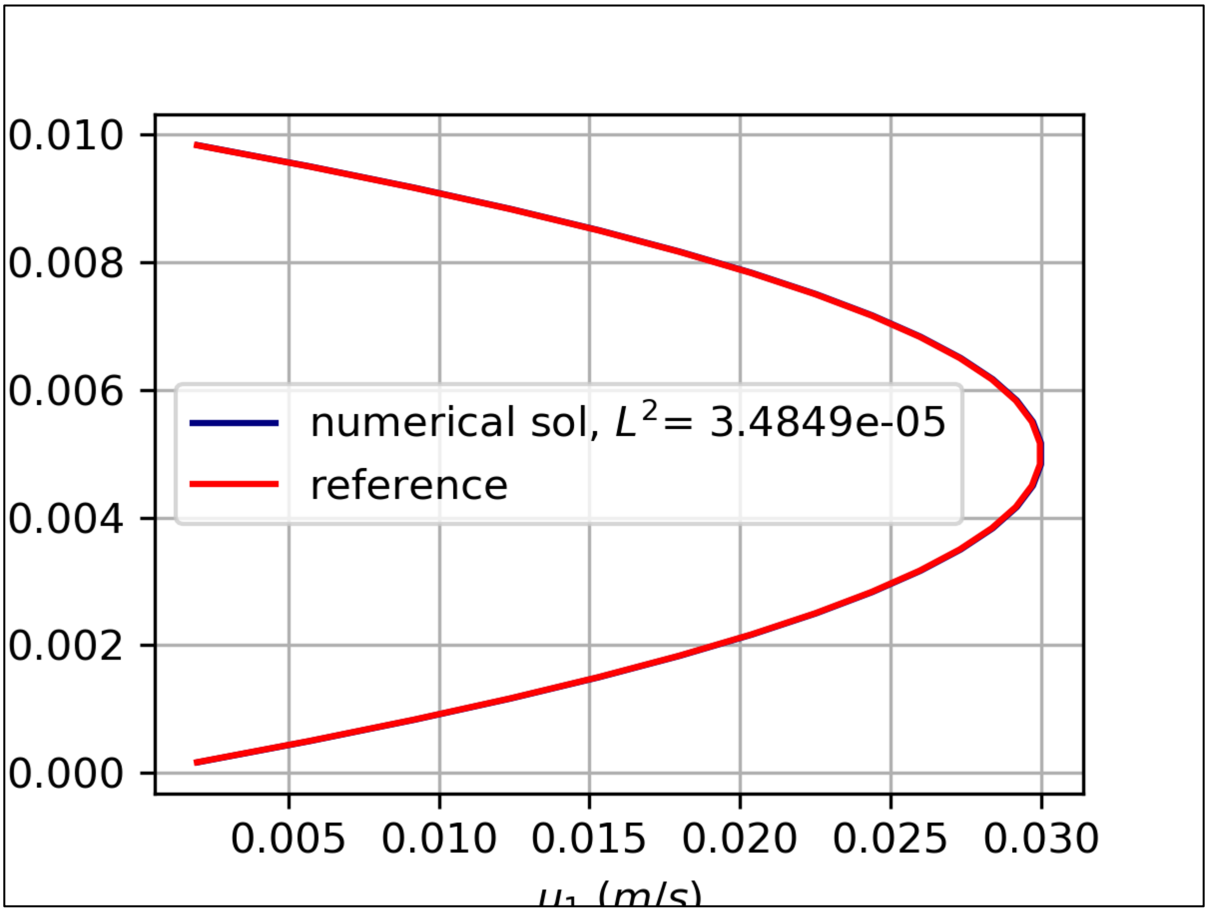
domain p^{n+1} contour (Pa)
0.008
0.006
0.004
0.002
0.005
0.010
0.015
0.000003
0.000002
0.000001
0.000000
0.000000
−0.00000
−0.00000
−0.00000
−0.00000


domain u_1 contour (m/s)
0.008
0.006
0.004
0.002
0.005
0.010
0.015
0.0285
0.0255
0.0225
0.0195
0.0165
0.0135
0.0105
0.0075
0.0045
0.0015


0.010
0.008
0.006
0.004
0.002
0.000
0.005
0.010
0.015
0.020
0.025
0.030
numerical sol, L^2= 1.4413e-05
reference


domain p^{n+1} contour (Pa)
1e−7
2.25
1.50
0.75
0.00
−0.75
−1.50
−2.25
0.008
0.006
0.004
0.002
0.005
0.010
0.015
x_1 (m)


domain u_1 contour (m/s)
0.0285
0.0240
0.0195
0.0150
0.0105
0.0060
0.0015
0.008
0.006
0.004
0.002
0.005
0.010
0.015
x_1 (m)



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:

$$u^* = u^n + \Delta t(\nu \boldsymbol{D}(u^n) - \frac{1}{\rho}\nabla \mathrm{P})$$

And the corrector step keeps the u^{n+1} 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 $0.00\ m/s$ initial velocity would have almost opposite pressure field distribution trend compared to case with $0.03\ m/s$.

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]+cel
l_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]+ce
ll_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()
```