

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Fri Feb 21 11:34:56 2020
4
5  @author: Wazowskai
6  """
7
8  import numpy as np
9  import math
10 import matplotlib.pyplot as plt
11
12
13
14 #problem constants
15 nu=1e-6
16 mu=1e-3
17 rho=1e+3
18 dt=0.0001
19 gradP=-2.4
20 n_iter=0
21 '''
22 node generation section
23 '''
24 #domain length
25
26 Lx1=0.02
27 Lx2=0.01
28
29 #number of cells on each direction
30 Nx1=60
31 Nx2=30
32
33 #mesh spacing
34 h=Lx1/Nx1
35
36 #uave
37 u1_ave=0.02
38
39 def Adv_x_n(i,j):
40     return
41     (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
42     _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
43     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]+c
44     ell_S_y_vn[i+1,j])))
45 def Dif_x_n(i,j):
46     return
47     (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-
48     1]-4*cell_S_x_un[i,j])
49 #def Adv_y_n(i,j):
50 #     return (1/h)*((0.5*(cell_S_y_vn[i,j]+cell_S_y_vn[i+1,j]))**2-(0.5*(cell_S_y_vn[i,j]+cell_S_y_vn[i-1,j]))**2+(0.5*(cell_S_y_vn[i,j+1]+cell_S_y_vn[i,j-1]))*(cell_S_x_un[i,j+1]+cell_S_x_un[i,j-1])-(0.5*(cell_S_y_vn[i,j]+cell_S_y_vn[i,j-1]))*(cell_S_x_un[i,j+1]+cell_S_x_un[i,j-1])))
51 def Dif_y_n(i,j):
52     return
53     (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])
54
55 def ref_vel_prof(x2):
56     '''
57     function returning reference analytic sol
58     '''
59     return -1200*((x2-0.005)**2)+0.03
60
61
62 #cell centroid coor
63 #the +2 stands for ghost cells on each direction
64 cell_cent_x=np.zeros([Nx1+2,Nx2+2])
65 cell_cent_y=np.zeros([Nx1+2,Nx2+2])
66

```

```

60 cell_cent_un=np.zeros([Nx1+2,Nx2+2])
61 cell_cent_us=np.zeros([Nx1+2,Nx2+2])
62 cell_cent_unn=np.zeros([Nx1+2,Nx2+2])
63
64 #cell corner coor
65 cell_cor_x=np.zeros([Nx1+3,Nx2+3])
66 cell_cor_y=np.zeros([Nx1+3,Nx2+3])
67
68 #surf area of the cell
69 cell_S_x=np.zeros([Nx1+2,Nx2+2])
70 cell_S_y=np.zeros([Nx1+2,Nx2+2])
71
72 cell_S_x_coor_x=np.zeros([Nx1+2,Nx2+2])
73 cell_S_x_coor_y=np.zeros([Nx1+2,Nx2+2])
74 cell_S_y_coor_x=np.zeros([Nx1+2,Nx2+2])
75 cell_S_y_coor_y=np.zeros([Nx1+2,Nx2+2])
76
77 #normal vector of cell surfaces
78 cell_S_x_nx=np.zeros([Nx1+2,Nx2+2])
79 cell_S_x_ny=np.zeros([Nx1+2,Nx2+2])
80 cell_S_y_nx=np.zeros([Nx1+2,Nx2+2])
81 cell_S_y_ny=np.zeros([Nx1+2,Nx2+2])
82 #surface velocities
83 cell_S_x_un=np.zeros([Nx1+2,Nx2+2])
84 cell_S_x_us=np.zeros([Nx1+2,Nx2+2])
85 cell_S_x_unn=np.zeros([Nx1+2,Nx2+2])
86
87 cell_S_x_vn=np.zeros([Nx1+2,Nx2+2])
88 cell_S_x_vs=np.zeros([Nx1+2,Nx2+2])
89 cell_S_x_vnn=np.zeros([Nx1+2,Nx2+2])
90 #cell_S_x_v=np.zeros([Nx1+2,Nx2+2])
91 cell_S_y_un=np.zeros([Nx1+2,Nx2+2])
92 cell_S_y_us=np.zeros([Nx1+2,Nx2+2])
93 cell_S_y_unn=np.zeros([Nx1+2,Nx2+2])
94
95 cell_S_y_vn=np.zeros([Nx1+2,Nx2+2])
96 cell_S_y_vs=np.zeros([Nx1+2,Nx2+2])
97 cell_S_y_vnn=np.zeros([Nx1+2,Nx2+2])
98 #cell_S_y_v=np.zeros([Nx1+2,Nx2+2])
99 #reference velocity profile
100 ref_S_u=np.zeros([Nx2+2])
101 L_sq=np.array([1.0,1.0])
102
103 #corner coor initialization
104 for j in range(0,Nx2+3):
105     for i in range(0, Nx1+3):
106         cell_cor_x[i,j]=(Lx1/Nx1)*(i-1)
107         cell_cor_y[i,j]=(Lx2/Nx2)*(j-1)
108
109 #cell cent coor storage
110 for j in range(0, Nx2+2):
111     for i in range(0, Nx1+2):
112
113         cell_cent_x[i,j]='{:10.6e}'.format(0.25*(cell_cor_x[i,j]+cell_cor_x[i+1,j]+cell_c
or_x[i,j+1]+cell_cor_x[i+1,j+1]))
114
115         cell_cent_y[i,j]='{:10.6e}'.format(0.25*(cell_cor_y[i,j]+cell_cor_y[i+1,j]+cell_c
or_y[i,j+1]+cell_cor_y[i+1,j+1]))
116         cell_S_x_coor_x[i,j]=(cell_cor_x[i,j]+cell_cor_x[i,j+1])/2
117         cell_S_x_coor_y[i,j]=(cell_cor_y[i,j]+cell_cor_y[i,j+1])/2
118         cell_S_y_coor_x[i,j]=(cell_cor_x[i,j]+cell_cor_x[i+1,j])/2
119         cell_S_y_coor_y[i,j]=(cell_cor_y[i,j]+cell_cor_y[i+1,j])/2
120         #initial conditions
121         cell_S_x_un[i,j]=0.03
122         cell_S_y_un[i,j]=0.00
123
124         cell_S_x[i,j]=abs(cell_cor_y[i,j]-cell_cor_y[i,j+1])

```

```

123         cell_S_y[i,j]=abs(cell_cor_x[i,j]-cell_cor_x[i+1,j])
124
125         cell_S_x_nx[i,j]=(cell_cor_y[i,j+1]-cell_cor_y[i,j])/cell_S_x[i,j]
126         cell_S_x_ny[i,j]=(cell_cor_x[i,j+1]-cell_cor_x[i,j])/cell_S_x[i,j]
127         cell_S_y_nx[i,j]=(cell_cor_y[i+1,j]-cell_cor_y[i,j])/cell_S_y[i,j]
128         cell_S_y_ny[i,j]=(cell_cor_x[i+1,j]-cell_cor_x[i,j])/cell_S_y[i,j]
129
130 plt.contourf(cell_cent_x[1:Nx1+1, 1:Nx2+1], cell_cent_y[1:Nx1+1, 1:Nx2+1],
131 cell_S_x_un[1:Nx1+1, 1:Nx2+1], 20, cmap='inferno')
132 plt.colorbar()
133 plt.show()
134
135 L_sq_r=L_sq[1]/L_sq[0]
136 for i in range(1, Nx2+1):
137     ref_S_u[i]=ref_vel_prof(cell_S_x_coar_y[0,i])
138
139 while L_sq_r<1.01:
140     L_sq[0]=L_sq[1]
141     #predictor step:
142     for j in range(1, Nx2+1):
143         for i in range(1, Nx1+1):
144             #cell_S_x_us[i,j]=cell_S_x_un[i,j]+dt*(-Adv_x_n(i,j)+nu*Dif_x_n(i,j))
145             cell_S_x_us[i,j]=cell_S_x_un[i,j]+dt*(nu*Dif_x_n(i,j))
146 # B.C. update
147 for j in range(0, Nx2+2):
148     cell_S_x_us[0,j]=cell_S_x_us[-2,j]
149     cell_S_x_us[-1,j]=cell_S_x_us[1,j]
150 for i in range(0, Nx1+2):
151     cell_S_x_us[i,0]=-cell_S_y_us[i,1]
152     cell_S_x_us[i,-1]=-cell_S_y_us[i,-2]
153
154 #corrector step:
155 for j in range(1, Nx2+1):
156     for i in range(1, Nx1+1):
157         cell_S_x_unn[i,j]=cell_S_x_us[i,j]-(1/rho)*(dt)*(gradP)
158
159 #B.C. update
160 for j in range(0, Nx2+2):
161     cell_S_x_unn[0,j]=cell_S_x_unn[-2,j]
162     cell_S_x_unn[-1,j]=cell_S_x_unn[1,j]
163 for i in range(0, Nx1+2):
164     cell_S_x_unn[i,0]=-cell_S_y_unn[i,1]
165     cell_S_x_unn[i,-1]=-cell_S_y_unn[i,-2]
166
167 for j in range(1, Nx2+1):
168     for i in range(1, Nx1+1):
169         cell_S_x_un[i,j]=cell_S_x_unn[i,j]
170
171 for j in range(0, Nx2+2):
172     cell_S_x_un[0,j]=cell_S_x_un[-2,j]
173     cell_S_x_un[-1,j]=cell_S_x_un[1,j]
174 for i in range(0, Nx1+2):
175     cell_S_x_un[i,0]=-cell_S_y_un[i,1]
176     cell_S_x_un[i,-1]=-cell_S_y_un[i,-2]
177
178 sq_sum_error=0
179
180 for i in range(1,Nx2+1):
181     sq_sum_error+=(ref_S_u[i]-cell_S_x_un[50,i])**2
182 L_sq[1]=math.sqrt(sq_sum_error/(Nx2+1))
183
184 if n_iter%10000==0:
185     print('iter= '+str(n_iter)+' , L_sq= {:.4e}'.format(L_sq[0]))
186     plt.plot(cell_S_x_un[50,1:Nx2+1],cell_S_x_coar_y[50,1:Nx2+1], color='navy',
187 label='numerical sol, $L^2$= {:.10e}'.format(L_sq[0]))
188     plt.plot(ref_S_u[1:Nx2+1] ,cell_S_x_coar_y[50,1:Nx2+1], color='red',
189 label='reference')
190     plt.xlabel('$u_1$ ($m/s$)')

```

```

187         plt.ylabel('$x_2$ (m)')
188         plt.legend()
189
190         plt.grid()
191         plt.show()
192
193         plt.contourf(cell_cent_x[1:Nx1+1, 1:Nx2+1], cell_cent_y[1:Nx1+1, 1:Nx2+1],
194                      cell_S_x_un[1:Nx1+1, 1:Nx2+1], 20, cmap='inferno')
195         plt.colorbar()
196         plt.xlabel('$x_1$ (m)')
197         plt.ylabel('$x_2$ (m)')
198         plt.title('domain $u_1$ contour ($m/s$)')
199         plt.show()
200         L_sq_r=L_sq[1]/L_sq[0]
201         n_iter+=1
202
203
204         print('iter= '+str(n_iter)+' , L_sq= {:.4e}'.format(L_sq[0]))
205         plt.plot(cell_S_x_un[50,1:Nx2+1],cell_S_x_coar_y[50,1:Nx2+1], color='navy',
206                label='numerical sol, $L^2$= {:.104e}'.format(L_sq[0]))
207         plt.plot(ref_S_u[1:Nx2+1] ,cell_S_x_coar_y[50,1:Nx2+1], color='red', label='reference')
208         plt.xlabel('$u_1$ ($m/s$)')
209         plt.ylabel('$x_2$ (m)')
210         plt.legend()
211         plt.grid()
212         plt.show()
213
214         plt.contourf(cell_cent_x[1:Nx1+1, 1:Nx2+1], cell_cent_y[1:Nx1+1, 1:Nx2+1],
215                      cell_S_x_un[1:Nx1+1, 1:Nx2+1], 20, cmap='inferno')
216         plt.colorbar()
217         plt.xlabel('$x_1$ (m)')
218         plt.ylabel('$x_2$ (m)')
219         plt.title('domain $u_1$ contour ($m/s$)')
220         plt.show()

```