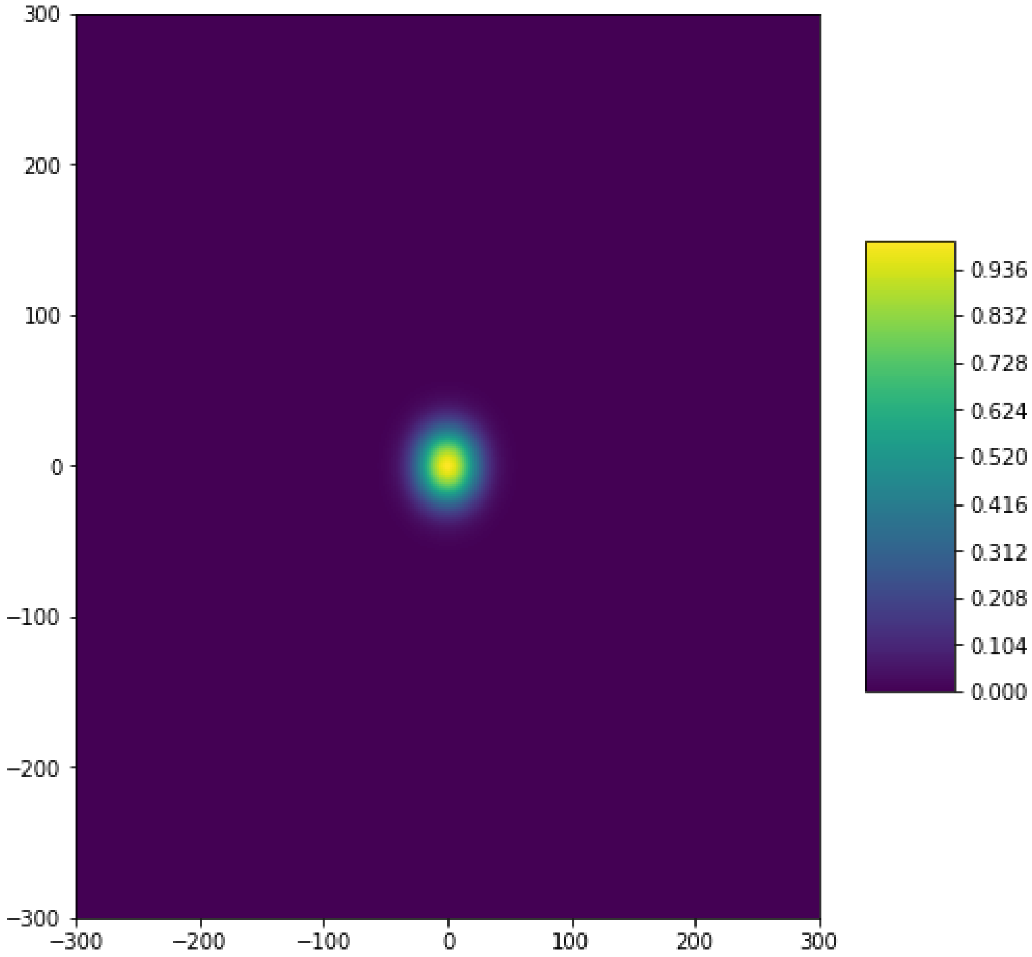
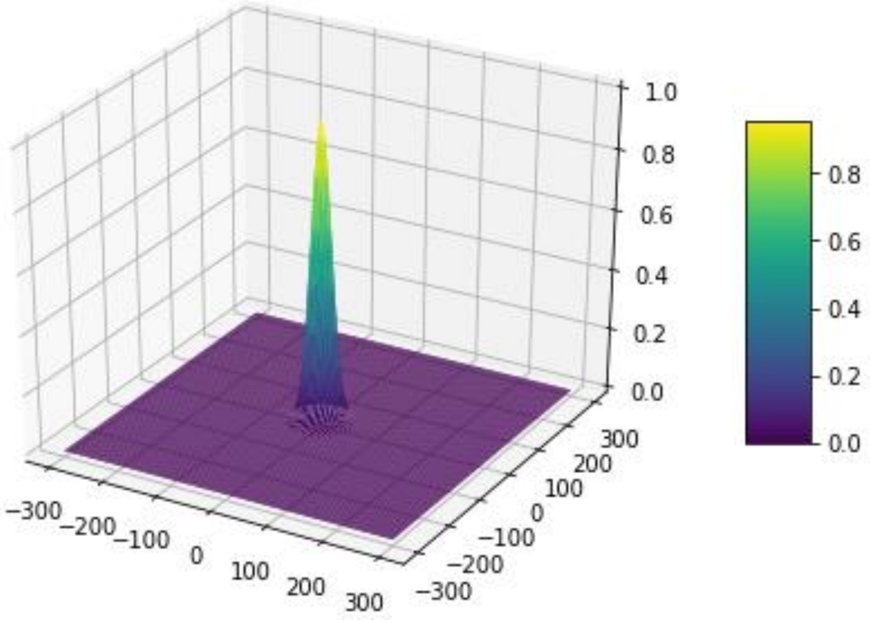
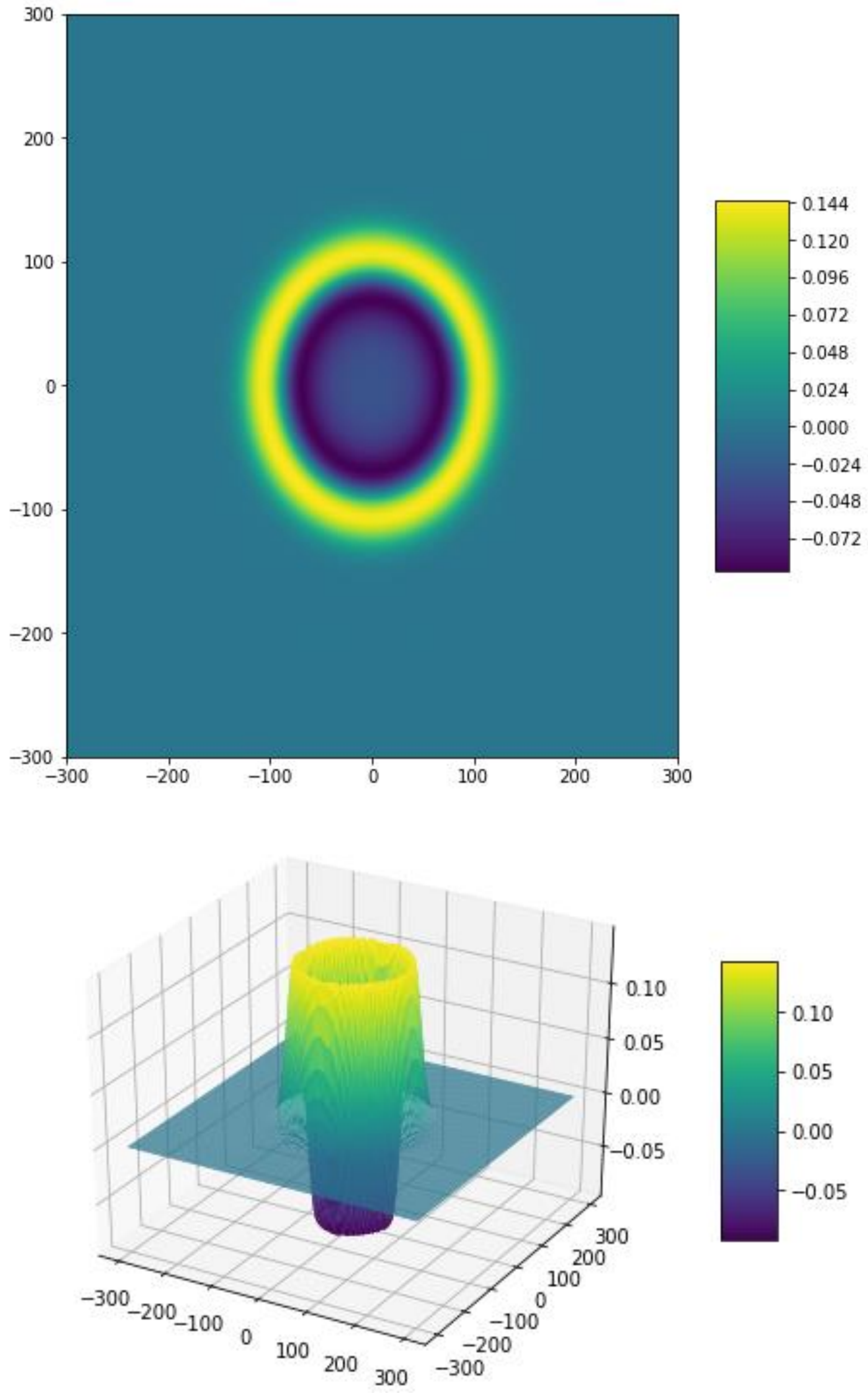
姓名 ： 徐昶亘 學號 ：r07525117

Time = 0s



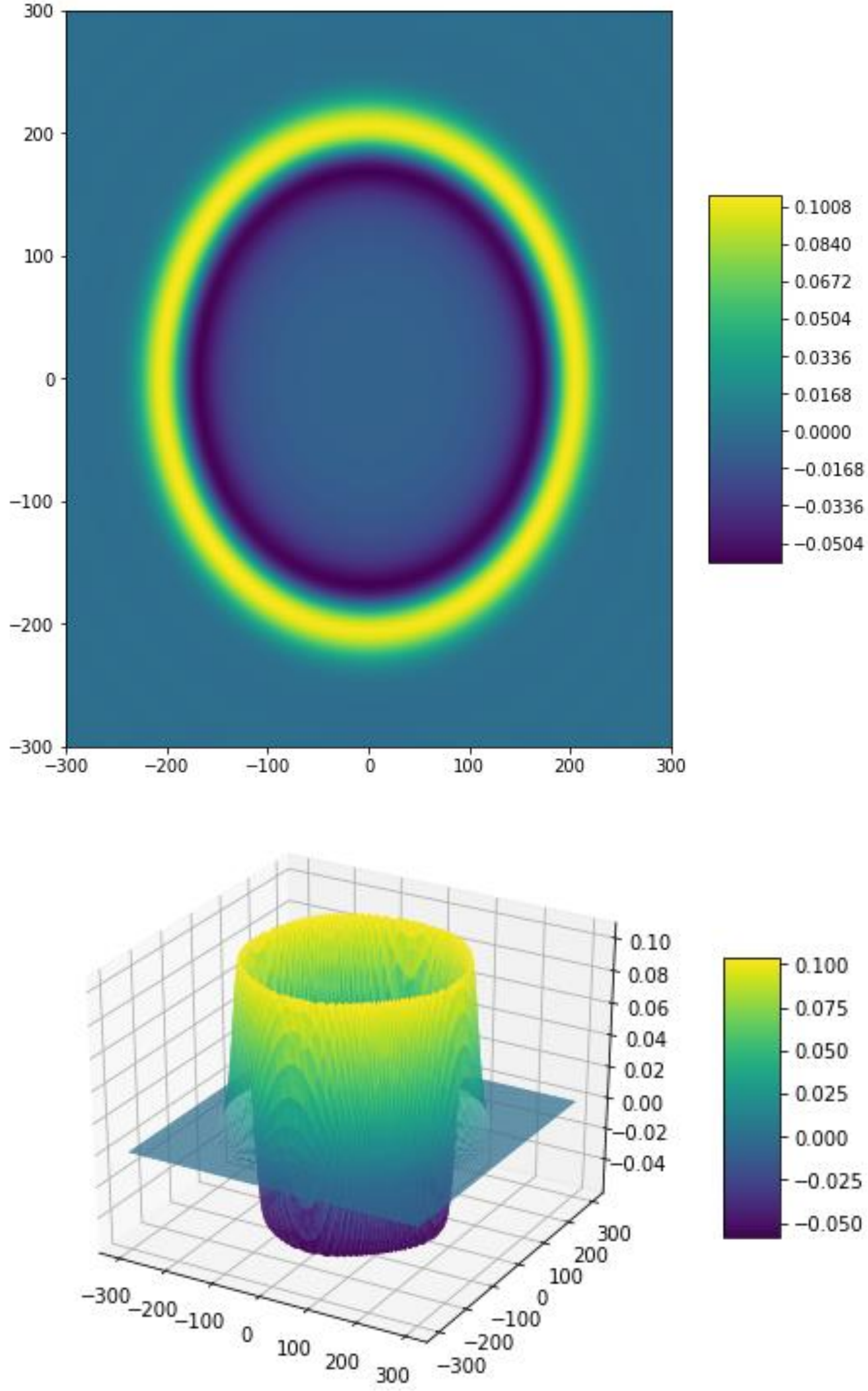


Time = 10s



300
200
100
0
−100
−200
−300
−300
−200
−100
0
100
200
300
0.144
0.120
0.096
0.072
0.048
0.024
0.000
−0.024
−0.048
−0.072
0.10
0.05
0.00
−0.05
300
200
100
0
−100
−200
−300
−300
−200
−100
0
100
200
300
0.10
0.05
0.00
−0.05

Time = 20s


300
200
100
0
−100
−200
−300
−300
−200
−100
0
100
200
300
0.1008
0.0840
0.0672
0.0504
0.0336
0.0168
0.0000
−0.0168
−0.0336
−0.0504
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−300
−200
−100
0
100
200
300
300
200
100
0
−100
−200
−300
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050

# Appendix

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 29 12:52:10 2020

@author: r07525117
"""

import os
os.chdir(r'D:\ShallowWaterComputation\HW4')
import numpy as np


def eta0(x,y,H=1,L=100):
    return H*np.exp( -18*( x / L )**2 )*np.exp( -18*( y /
L )**2 )

def U0(x,y):
    return 0.0

def V0(x,y):
    return 0.0


g = 9.806

xMax = 300
xMin = -300
yMax = 300
yMin = -300

tMax = 20
tMin = 0
```

```
deltaX = 5
deltaY = 5
deltaT = 0.05
# CFL = 0.85
# deltaT = CFL*( min(deltaX,deltaY) / np.sqrt(g*10) )

x = np.arange(xMin-2*deltaX, xMax+3*deltaX, deltaX)
y = np.arange(yMin-2*deltaY, yMax+3*deltaY, deltaY)
t = np.arange(0, tMax, deltaT)

[X,Y]=np.meshgrid(x,y)
Y = np.flip(Y) ## Make +Y direction be Upper

tlen = len(t)
xlen = len(x)
ylen = len(y)


ETA = np.zeros( ( tlen, ylen, xlen) )
U = np.zeros( ( tlen, ylen, xlen ) )
V = np.zeros( ( tlen, ylen, xlen ) )

ETAs1 = np.zeros( (  ylen, xlen) )
Us1 = np.zeros( (  ylen, xlen ) )
Vs1 = np.zeros( (  ylen, xlen ) )

ETAs2 = np.zeros( (  ylen, xlen) )
Us2 = np.zeros( (  ylen, xlen ) )
Vs2 = np.zeros( (  ylen, xlen ) )

h = np.zeros( ( ylen, xlen) )
h[:] = 10

GN = 2

##Initialization
ETA[0][GN:-GN, GN:-GN] = eta0( X[GN:-GN, GN:-GN], Y[GN:-GN,
GN:-GN] )
```

```
U[0][GN:-GN, GN:-GN] = U0( X[GN:-GN, GN:-GN], Y[GN:-GN, GN:-GN] )
V[0][GN:-GN, GN:-GN] = V0( X[GN:-GN, GN:-GN], Y[GN:-GN, GN:-GN] )

#Initialization BC
ETA[0][GN:-GN, GN:-GN][ : , [0,1,-1,-2]  ] = ETA[0][GN:-GN, GN:-GN][ : , [4,3,-5,-4] ]
ETA[0][GN:-GN, GN:-GN][ [0,1,-1,-2] , : ] = ETA[0][GN:-GN, GN:-GN][ [4,3,-5,-4] , : ]
U[0][GN:-GN, GN:-GN][:,[0,1,-1,-2]] = -U[0][GN:-GN, GN:-GN][:,[4,3,-5,-4]]  ##L boundary
U[0][GN:-GN, GN:-GN][[0,1,-1,-2],:] = U[0][GN:-GN, GN:-GN][[4,3,-5,-4],:] ##D boundary
V[0][GN:-GN, GN:-GN][:,[0,1,-1,-2]] = V[0][GN:-GN, GN:-GN][:,[4,3,-5,-4]]
V[0][GN:-GN, GN:-GN][[0,1,-1,-2],:] = -V[0][GN:-GN, GN:-GN][[4,3,-5,-4],:]


# import matplotlib.pyplot as plt
# from mpl_toolkits.mplot3d import Axes3D

# fig = plt.figure(figsize=(8, 8))
# plt.contourf( X[2:-2,2:-2] ,Y[2:-2,2:-2] , ETA[0][2:-2,2:-2] )
# plt.xlim([-40, 40])
# plt.ylim([-40, 40])
# plt.show()





n = 0
while( n < (tlen-1) ):

    #Cell-first round
    ETAs1[GN:-GN, GN:-GN] = ETA[n][GN:-GN, GN:-GN]\
    - ( (deltaT) / (12*deltaX) )*\
    ( -U[n][GN:-GN, (GN+2):]*h[GN:-GN, (GN+2):]\
    + 8*U[n][GN:-GN, (GN+1):(-GN+1)]*h[GN:-GN, (GN+1):(-GN+1)]\
    -8*U[n][GN:-GN, (GN-1):(-GN-1)]*h[GN:-GN, (GN-1):(-GN-1)] \
    + U[n][GN:-GN, (GN-2):(-GN-2)]*h[GN:-GN, (GN-2):(-GN-2)] )\
    -( (deltaT) / (12*deltaY) )*\
    ( -V[n][(GN+2):, GN:-GN]*h[(GN+2):, GN:-GN] \
    +8*V[n][(GN+1):(-GN+1), GN:-GN]*h[(GN+1):(-GN+1), GN:-GN]\
    -8*V[n][(GN-1):(-GN-1), GN:-GN]*h[(GN-1):(-GN-1), GN:-GN] \
    + V[n][(GN-2):(-GN-2), GN:-GN]*h[(GN-2):(-GN-2), GN:-GN] )

    Us1[GN:-GN, GN:-GN] = U[n][GN:-GN, GN:-GN] - ( (deltaT*g) /
(12*deltaX) )*\
    (-ETA[n][GN:-GN, (GN+2):]
    +8*ETA[n][GN:-GN, (GN+1):(-GN+1)]\
    -8*ETA[n][GN:-GN, (GN-1):(-GN-1)]\
    +ETA[n][GN:-GN, (GN-2):(-GN-2)] )

    Vs1[GN:-GN, GN:-GN] = V[n][GN:-GN, GN:-GN] - ( (deltaT*g) /
(12*deltaY) )*\
    (-ETA[n][(GN+2):, GN:-GN]\
    +8*ETA[n][(GN+1):(-GN+1), GN:-GN]\
    -8*ETA[n][(GN-1):(-GN-1), GN:-GN]\
    +ETA[n][(GN-2):(-GN-2), GN:-GN] )

    #BC-first round
    ETAs1[GN:-GN, GN:-GN][ : , [0,1,-1,-2]  ] = ETAs1[GN:-GN,
GN:-GN][ : , [4,3,-5,-4] ]
    ETAs1[GN:-GN, GN:-GN][ [0,1,-1,-2] , : ] = ETAs1[GN:-GN,
GN:-GN][ [4,3,-5,-4] , : ]

    Us1[GN:-GN, GN:-GN][:,[0,1,-1,-2]] = -Us1[GN:-GN, GN:-
GN][:,[4,3,-5,-4]]  ##L boundary
    Us1[GN:-GN, GN:-GN][[0,1,-1,-2],:] = Us1[GN:-GN, GN:-
GN][[4,3,-5,-4],:] ## D boundary

    Vs1[GN:-GN, GN:-GN][:,[0,1,-1,-2]] = Vs1[GN:-GN, GN:-
```

```
GN][:,[4,3,-5,-4]] ##L boundary
    Vs1[GN:-GN, GN:-GN][[0,1,-1,-2],:] = -Vs1[GN:-GN, GN:-
GN][[4,3,-5,-4],:] ##D boundary

    #Cell-Second round
    ETAs2[GN:-GN, GN:-GN] = (3.0/4.0)*ETA[n][GN:-GN, GN:-GN] +
(1.0/4.0)*ETAs1[GN:-GN, GN:-GN]\
    -( deltaT/(4.0*12*deltaX) )*\
    (-Us1[GN:-GN, (GN+2):]*h[GN:-GN, (GN+2):]\
    +8*Us1[GN:-GN, (GN+1):(-GN+1)]*h[GN:-GN, (GN+1):(-GN+1)]\
    -8*Us1[GN:-GN, (GN-1):(-GN-1)]*h[GN:-GN, (GN-1):(-GN-1)]\
    +Us1[GN:-GN, (GN-2):(-GN-2)]*h[GN:-GN, (GN-2):(-GN-2)] )\
    -( deltaT/(4.0*12*deltaY) )*\
    (-Vs1[(GN+2):, GN:-GN ]*h[(GN+2):, GN:-GN]\
    +8*Vs1[(GN+1):(-GN+1), GN:-GN]*h[(GN+1):(-GN+1), GN:-GN]\
    -8*Vs1[(GN-1):(-GN-1), GN:-GN]*h[(GN-1):(-GN-1), GN:-GN]\
    +Vs1[(GN-2):(-GN-2), GN:-GN]*h[(GN-2):(-GN-2), GN:-GN])

    Us2[GN:-GN, GN:-GN] = (3.0/4.0)*U[n][GN:-GN, GN:-GN] +
(1.0/4.0)*Us1[GN:-GN, GN:-GN]\
    - ( (deltaT*g) / (4.0*12*deltaX) )*\
    ( -ETAs1[GN:-GN, (GN+2):]\
     +8*ETAs1[GN:-GN, (GN+1):(-GN+1)]\
     -8*ETAs1[GN:-GN, (GN-1):(-GN-1)]\
     +ETAs1[GN:-GN, (GN-2):(-GN-2)] )

    Vs2[GN:-GN, GN:-GN] = (3.0/4.0)*V[n][GN:-GN, GN:-GN] +
(1.0/4.0)*Vs1[GN:-GN, GN:-GN]\
    - ( (deltaT*g) / (4.0*12*deltaY) )*\
    ( -ETAs1[(GN+2):, GN:-GN]\
     +8*ETAs1[(GN+1):(-GN+1), GN:-GN]\
     -8*ETAs1[(GN-1):(-GN-1), GN:-GN]\
     +ETAs1[(GN-2):(-GN-2), GN:-GN] )

    #BC-Second round
    ETAs2[GN:-GN, GN:-GN][ : , [0,1,-1,-2]  ] = ETAs2[GN:-GN,
GN:-GN][ : , [4,3,-5,-4] ]
    ETAs2[GN:-GN, GN:-GN][ [0,1,-1,-2] , : ] = ETAs2[GN:-GN,
GN:-GN][ [4,3,-5,-4] , : ]


    Us2[GN:-GN, GN:-GN][:,[0,1,-1,-2]] = -Us2[GN:-GN, GN:-
GN][:,[4,3,-5,-4]]  ##L boundary
    Us2[GN:-GN, GN:-GN][[0,1,-1,-2],:] = Us2[GN:-GN, GN:-
GN][[4,3,-5,-4],:] ## D boundary

    Vs2[GN:-GN, GN:-GN][:,[0,1,-1,-2]] = Vs2[GN:-GN, GN:-
GN][:,[4,3,-5,-4]] ##L boundary
    Vs2[GN:-GN, GN:-GN][[0,1,-1,-2],:] = -Vs2[GN:-GN, GN:-
GN][[4,3,-5,-4],:] ##D boundary

    #Cell-Third round
    ETA[n+1][GN:-GN, GN:-GN] = (1.0/3.0)*ETA[n][GN:-GN, GN:-GN]
+ (2.0/3.0)*ETAs2[GN:-GN, GN:-GN]\
    -( (2*deltaT) / (3.0*12*deltaX) )*\
    (-Us2[GN:-GN, (GN+2):]*h[GN:-GN, (GN+2):]\
    +8*Us2[GN:-GN, (GN+1):(-GN+1)]*h[GN:-GN, (GN+1):(-GN+1)]\
    -8*Us2[GN:-GN, (GN-1):(-GN-1)]*h[GN:-GN, (GN-1):(-GN-1)]\
    +Us2[GN:-GN, (GN-2):(-GN-2)]*h[GN:-GN, (GN-2):(-GN-2)] )\
    -( (2*deltaT) / (3.0*12*deltaY) )*\
    ( -Vs2[(GN+2):, GN:-GN]*h[(GN+2):, GN:-GN]\
     +8*Vs2[(GN+1):(-GN+1), GN:-GN]*h[(GN+1):(-GN+1), GN:-GN]\
     -8*Vs2[(GN-1):(-GN-1), GN:-GN]*h[(GN-1):(-GN-1), GN:-GN]\
     +Vs2[(GN-2):(-GN-2), GN:-GN]*h[(GN-2):(-GN-2), GN:-GN] )

    U[n+1][GN:-GN, GN:-GN] = (1.0/3.0)*U[n][GN:-GN, GN:-GN] +
(2.0/3.0)*Us2[GN:-GN, GN:-GN]\
    - ( (2*deltaT*g)/(3.0*12*deltaX) )*\
    ( -ETAs2[GN:-GN, (GN+2):]\
    +8*ETAs2[GN:-GN, (GN+1):(-GN+1)]\
    -8*ETAs2[GN:-GN, (GN-1):(-GN-1)]\
    +ETAs2[GN:-GN, (GN-2):(-GN-2)] )

    V[n+1][GN:-GN, GN:-GN] = (1.0/3.0)*V[n][GN:-GN, GN:-GN] +
(2.0/3.0)*Vs2[GN:-GN, GN:-GN]\
    -( (2*deltaT*g)/(3.0*12*deltaY) )*\
    ( -ETAs2[(GN+2):, GN:-GN]\
    +8*ETAs2[(GN+1):(-GN+1), GN:-GN]\
    -8*ETAs2[(GN-1):(-GN-1), GN:-GN]\
    +ETAs2[(GN-2):(-GN-2), GN:-GN] )

    #BC-Third round
    ETA[n+1][GN:-GN, GN:-GN][ : , [0,1,-1,-2]  ] =
ETA[n+1][GN:-GN, GN:-GN][ : , [4,3,-5,-4] ]
    ETA[n+1][GN:-GN, GN:-GN][ [0,1,-1,-2] , : ] = ETA[n+1][GN:-
GN, GN:-GN][ [4,3,-5,-4] , : ]

    U[n+1][GN:-GN, GN:-GN][:,[0,1,-1,-2]] = -U[n+1][GN:-GN,
GN:-GN][:,[4,3,-5,-4]]  ##L boundary
    U[n+1][GN:-GN, GN:-GN][[0,1,-1,-2],:] = U[n+1][GN:-GN, GN:-
GN][[4,3,-5,-4],:] ##D boundary

    V[n+1][GN:-GN, GN:-GN][:,[0,1,-1,-2]] = V[n+1][GN:-GN, GN:-
GN][:,[4,3,-5,-4]]
    V[n+1][GN:-GN, GN:-GN][[0,1,-1,-2],:] = -V[n+1][GN:-GN,
GN:-GN][[4,3,-5,-4],:]
    n = n+1
#return x,t,ETA,U,V


import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

fig = plt.figure(figsize=(8, 8))
surf = plt.contourf( X[2:-2,2:-2], Y[2:-2,2:-2], ETA[0][2:-
2,2:-2], levels=250)
# plt.xlim([-300, 300])
# plt.ylim([-300, 300])

fig.colorbar(surf, shrink=0.5, aspect=5.0)
plt.show()
```

```
fig = plt.figure()
ax = Axes3D(fig)
surf = ax.plot_surface(X[2:-2, 2:-2], Y[2:-2, 2:-2],
ETA[0][2:-2, 2:-2], rstride=1, cstride=1, cmap=cm.viridis)
fig.colorbar(surf, shrink=0.5, aspect=5.0)
plt.show()
```