import matplotlib.pyplot as plt

import numpy as np

def main():

""" Finite Volume simulation """

# Simulation parameters

Nx = 400 # resolution x-dir

Ny = 100 # resolution y-dir

rho0 = 100 # average density

tau = 0.6 # collision timescale

Nt = 4000 # number of timesteps

plotRealTime = True # switch on for plotting as the simulation goes along

# Lattice speeds / weights

NL = 9

idxs = np.arange(NL)

cxs = np.array([0, 0, 1, 1, 1, 0,-1,-1,-1])

cys = np.array([0, 1, 1, 0,-1,-1,-1, 0, 1])

weights = np.array([4/9,1/9,1/36,1/9,1/36,1/9,1/36,1/9,1/36]) # sums to 1

# Initial Conditions

F = np.ones((Ny,Nx,NL)) #\* rho0 / NL

np.random.seed(42)

F += 0.01\*np.random.randn(Ny,Nx,NL)

X, Y = np.meshgrid(range(Nx), range(Ny))

F[:,:,3] += 2 \* (1+0.2\*np.cos(2\*np.pi\*X/Nx\*4))

rho = np.sum(F,2)

for i in idxs:

F[:,:,i] \*= rho0 / rho

# Cylinder boundary

X, Y = np.meshgrid(range(Nx), range(Ny))

cylinder = (X - Nx/4)\*\*2 + (Y - Ny/2)\*\*2 < (Ny/4)\*\*2

# Prep figure

fig = plt.figure(figsize=(4,2), dpi=80)

# Simulation Main Loop

for it in range(Nt):

print(it)

# Drift

for i, cx, cy in zip(idxs, cxs, cys):

F[:,:,i] = np.roll(F[:,:,i], cx, axis=1)

F[:,:,i] = np.roll(F[:,:,i], cy, axis=0)

# Set reflective boundaries

bndryF = F[cylinder,:]

bndryF = bndryF[:,[0,5,6,7,8,1,2,3,4]]

# Calculate fluid variables

rho = np.sum(F,2)

ux = np.sum(F\*cxs,2) / rho

uy = np.sum(F\*cys,2) / rho

# Apply Collision

Feq = np.zeros(F.shape)

for i, cx, cy, w in zip(idxs, cxs, cys, weights):

Feq[:,:,i] = rho \* w \* ( 1 + 3\*(cx\*ux+cy\*uy) + 9\*(cx\*ux+cy\*uy)\*\*2/2 - 3\*(ux\*\*2+uy\*\*2)/2 )

F += -(1.0/tau) \* (F - Feq)

# Apply boundary

F[cylinder,:] = bndryF

# plot in real time - color 1/2 particles blue, other half red

if (plotRealTime and (it % 10) == 0) or (it == Nt-1):

plt.cla()

ux[cylinder] = 0

uy[cylinder] = 0

vorticity = (np.roll(ux, -1, axis=0) - np.roll(ux, 1, axis=0)) - (np.roll(uy, -1, axis=1) - np.roll(uy, 1, axis=1))

vorticity[cylinder] = np.nan

cmap = plt.cm.bwr

cmap.set\_bad('black')

plt.imshow(vorticity, cmap='bwr')

plt.clim(-.1, .1)

ax = plt.gca()

ax.invert\_yaxis()

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

ax.set\_aspect('equal')

plt.pause(0.001)

# Save figure

plt.savefig('latticeboltzmann.png',dpi=240)

plt.show()

return 0

if \_\_name\_\_== "\_\_main\_\_":

main()