import numpy as np

import matplotlib.pyplot as plt

# Parameters

N = 32 # grid size

timesteps = 150

noise = 0.15 # thermal noise / temperature

mag\_field = 0.8 # "magnetic field" strength

# MBT Phase field (theta), initial state = random

theta = np.random.uniform(0, 2\*np.pi, (N, N))

vortex\_memory = np.zeros\_like(theta)

# Artificial vector potential for a uniform "magnetic field"

def vector\_potential(x, y, B):

return (0, B \* x) # Landau gauge for simplicity

# Evolve MBT phase field with synthetic field (first-principles)

for t in range(timesteps):

for i in range(N):

for j in range(N):

# Neighbors with periodic BCs

ip, im = (i+1)%N, (i-1)%N

jp, jm = (j+1)%N, (j-1)%N

dtheta = 0

# x direction (vector potential shifts phase)

Ax\_i, Ay\_i = vector\_potential(i, j, mag\_field)

Ax\_ip, Ay\_ip = vector\_potential(ip, j, mag\_field)

dtheta += np.cos(theta[ip,j] - theta[i,j] - (Ay\_ip-Ay\_i))

dtheta += np.cos(theta[im,j] - theta[i,j] - (Ay\_i-vector\_potential(im,j,mag\_field)[1]))

# y direction

Ax\_jp, Ay\_jp = vector\_potential(i, jp, mag\_field)

dtheta += np.cos(theta[i,jp] - theta[i,j] + (Ax\_jp-Ax\_i))

dtheta += np.cos(theta[i,jm] - theta[i,j] + (Ax\_i-vector\_potential(i,jm,mag\_field)[0]))

# Update: MBT "motion" + memory

theta[i,j] += 0.05\*dtheta + noise\*np.random.randn()

vortex\_memory[i,j] += np.abs(dtheta)

# Vorticity detection (lattice curl of phase)

vorticity = np.zeros\_like(theta)

for i in range(N):

for j in range(N):

ip, jp = (i+1)%N, (j+1)%N

loop = (theta[i,jp] - theta[i,j] +

theta[ip,jp] - theta[i,jp] +

theta[ip,j] - theta[ip,jp] +

theta[i,j] - theta[ip,j])

vorticity[i,j] = np.sin(loop)

# Visualization

plt.figure(figsize=(14,4))

plt.subplot(1,3,1)

plt.title("Final MBT Phase Field (cos)")

plt.imshow(np.cos(theta), cmap='twilight')

plt.axis('off')

plt.subplot(1,3,2)

plt.title("Vortex Memory Field (MBT Order)")

plt.imshow(vortex\_memory, cmap='inferno')

plt.axis('off')

plt.subplot(1,3,3)

plt.title("MBT Vorticity (Abrikosov Lattice)")

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

plt.colorbar(label='Vorticity')

plt.axis('off')

plt.tight\_layout()

plt.show()