import numpy as np

import matplotlib.pyplot as plt

# --- Setup parameters ---

N = 32 # Lattice size

timesteps = 80

# --- Three sites for teleportation ---

sites = {'A': (8, 8), 'B': (24, 8), 'C': (16, 24)}

# --- Initialize phase and memory fields ---

phase = np.zeros((N, N))

memory = np.zeros((N, N))

# --- Step 1: "Entangle" Alice (A) and Bob (B) by a shared vortex ---

def entangle(a, b):

ax, ay = a

bx, by = b

for x in range(N):

for y in range(N):

rA = np.hypot(x-ax, y-ay) + 1e-3

rB = np.hypot(x-bx, y-by) + 1e-3

phase[x, y] += np.arctan2(y-ay, x-ax)

phase[x, y] -= np.arctan2(y-by, x-bx)

entangle(sites['A'], sites['B'])

# --- Step 2: Alice encodes a logic state (memory "twist") ---

def encode(alice\_site):

ax, ay = alice\_site

for x in range(N):

for y in range(N):

r = np.hypot(x-ax, y-ay) + 1e-3

memory[x, y] += np.exp(-r/2) \* np.cos(phase[x, y])

encode(sites['A'])

# --- Step 3: Local MBT "braid" at Alice (simulate measurement) ---

def braid(site):

# Add local field disturbance

bx, by = site

for x in range(N):

for y in range(N):

r = np.hypot(x-bx, y-by)

phase[x, y] += 0.5 \* np.exp(-r/2)

braid(sites['A'])

# --- Step 4: Teleport memory to Charlie (field swap, "classical" message) ---

def teleport(alice, charlie):

# Move memory pattern from Alice to Charlie, destroy at Alice

ax, ay = alice

cx, cy = charlie

for x in range(N):

for y in range(N):

dx, dy = x-ax, y-ay

# Move memory to new position

newx, newy = np.clip(cx+dx, 0, N-1), np.clip(cy+dy, 0, N-1)

memory[newx, newy] += memory[x, y]

memory[x, y] = 0

teleport(sites['A'], sites['C'])

# --- Step 5: Braid at Bob (needed for full teleport logic) ---

braid(sites['B'])

# --- Visualization ---

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

plt.subplot(1,3,1)

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

plt.imshow(np.cos(phase), cmap="twilight", origin="lower")

for k, (x, y) in sites.items():

plt.plot(x, y, "o", label=k)

plt.legend()

plt.subplot(1,3,2)

plt.title("Memory Field (Before/After Teleport)")

plt.imshow(memory, cmap="inferno", origin="lower")

for k, (x, y) in sites.items():

plt.plot(x, y, "o", label=k)

plt.legend()

plt.subplot(1,3,3)

plt.title("Memory Profile")

plt.plot([memory[x, y] for (x, y) in sites.values()], "o-", label=list(sites.keys()))

plt.xticks(range(3), list(sites.keys()))

plt.ylabel("Logic Memory")

plt.legend()

plt.tight\_layout()

plt.show()