import numpy as np

import matplotlib.pyplot as plt

# Parameters

N = 32 # grid size

timesteps = 100

braid\_orders = [

['A', 'B', 'A'], # First Fibonacci braid

['B', 'A', 'B'], # Second order (should NOT commute)

]

# Fibonacci fusion "golden ratio"

phi = (1 + np.sqrt(5)) / 2

def braid\_phase(pattern, phi):

# Encodes a “Fibonacci” phase winding

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

centers = {

'A': (N//4, N//4),

'B': (3\*N//4, 3\*N//4)

}

for p in pattern:

cx, cy = centers[p]

for i in range(N):

for j in range(N):

dx, dy = i - cx, j - cy

angle = np.arctan2(dy, dx)

# Fibonacci phase: windings by multiples of golden ratio

field[i, j] += phi \* angle

# Normalize phase field

field = np.cos(field)

return field

def measure\_memory(field):

# “Memory” is average global coherence (absolute sum)

return np.mean(np.abs(field))

results = []

for braid in braid\_orders:

field = braid\_phase(braid, phi)

memory = measure\_memory(field)

results.append((braid, field, memory))

# Plot MBT Fibonacci Anyon Logic Demo

plt.figure(figsize=(12,4))

for idx, (braid, field, memory) in enumerate(results):

plt.subplot(1, 3, idx+1)

plt.imshow(field, cmap='twilight', interpolation='nearest')

plt.title(f"Braid: {'→'.join(braid)}\nMemory: {memory:.3f}")

plt.axis('off')

# Logic difference plot

plt.subplot(1, 3, 3)

memA = results[0][2]

memB = results[1][2]

plt.bar(['A→B→A', 'B→A→B'], [memA, memB], color=['orange', 'purple'])

plt.title(f"Logic Difference: {memA-memB:.3e}")

plt.ylabel("MBT Logic Memory")

plt.tight\_layout()

plt.show()