import numpy as np

import matplotlib.pyplot as plt

np.random.seed(42)

N = 32 # Grid size

steps = 100

# 1. Initialize MBT phase field

field = np.random.uniform(-np.pi, np.pi, (N, N))

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

# Anyon positions (corners)

anyonA = [3, 3]

anyonB = [N-4, 3]

anyonC = [N//2, N-4]

def braid\_path(start, end, T=steps):

x, y = np.linspace(start[0], end[0], T), np.linspace(start[1], end[1], T)

return list(zip(np.round(x).astype(int), np.round(y).astype(int)))

def apply\_braid(field, memory, path):

# Braiding = phase twist along the path

for (x, y) in path:

if 0 <= x < N and 0 <= y < N:

field[x, y] += np.pi/2 # Apply phase twist (fractional)

memory[x, y] += 1

return field, memory

def braid\_sequence(order):

# Start fresh memory

field\_braid = field.copy()

mem\_braid = np.zeros\_like(field)

paths = {

'A': braid\_path(anyonA, anyonB),

'B': braid\_path(anyonB, anyonC),

'C': braid\_path(anyonC, anyonA)

}

for label in order:

field\_braid, mem\_braid = apply\_braid(field\_braid, mem\_braid, paths[label])

return field\_braid, mem\_braid

# Run two different non-commuting braid orders

order1 = ['A', 'B', 'C']

order2 = ['C', 'B', 'A']

field1, mem1 = braid\_sequence(order1)

field2, mem2 = braid\_sequence(order2)

# Calculate topological "memory" (logic state) = total winding in memory field

memory1 = np.sum(np.cos(field1) \* mem1)

memory2 = np.sum(np.cos(field2) \* mem2)

diff = memory2 - memory1

# --- PLOTS ---

fig, axs = plt.subplots(1, 3, figsize=(14,4))

axs[0].imshow(np.cos(field1), cmap='twilight', vmin=-1, vmax=1)

axs[0].set\_title(f'Braiding: {"→".join(order1)}\nMemory: {memory1:.3f}')

axs[0].axis('off')

axs[1].imshow(np.cos(field2), cmap='twilight', vmin=-1, vmax=1)

axs[1].set\_title(f'Braiding: {"→".join(order2)}\nMemory: {memory2:.3f}')

axs[1].axis('off')

axs[2].bar(['→'.join(order1), '→'.join(order2)], [memory1, memory2], color=['cyan','magenta'])

axs[2].set\_ylabel('MBT Logic State')

axs[2].set\_title(f'Difference: {diff:.4f}\n(Non-Abelian logic)')

plt.tight\_layout()

plt.show()

print(f"Memory after {order1}: {memory1:.4f}")

print(f"Memory after {order2}: {memory2:.4f}")

print(f"Difference (non-Abelian logic): {diff:.4e}")