# MBT Non-Abelian Double Braid with Error Injection & Correction

import numpy as np

import matplotlib.pyplot as plt

np.random.seed(42)

L = 32

phase = np.random.uniform(0, 2 \* np.pi, (L, L))

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

braid\_sites = [(6,6), (6,25), (25,6), (25,25)] # corners

def braid(phase, memory, order, chirality=1.0, steps=25, noise\_amp=0.0):

for step in range(steps):

for idx, site in enumerate(order):

i, j = braid\_sites[site]

dphi = chirality \* 0.15 \* (-1 if idx%2==0 else 1)

phase[i, :] += dphi

phase[:, j] -= dphi

# MBT memory update: "remembers" local phase info, heals errors

memory += np.cos(phase) \* 0.12

memory \*= 0.98 # gentle forgetting/healing

if noise\_amp > 0:

memory += np.random.normal(0, noise\_amp, memory.shape) # Inject errors

return phase, memory

# Run A then B (errors after A)

phase\_A, mem\_A = phase.copy(), memory.copy()

phase\_A, mem\_A = braid(phase\_A, mem\_A, [0,1], chirality=1.1, steps=20)

# Inject errors at midpoint

err\_idx = np.random.choice(L\*L, size=30, replace=False)

mem\_A.ravel()[err\_idx] += np.random.normal(0, 1.2, size=30)

phase\_A, mem\_A = braid(phase\_A, mem\_A, [2,3], chirality=1.1, steps=20)

# Run B then A (errors after B)

phase\_B, mem\_B = phase.copy(), memory.copy()

phase\_B, mem\_B = braid(phase\_B, mem\_B, [2,3], chirality=1.1, steps=20)

# Inject errors at midpoint

mem\_B.ravel()[err\_idx] += np.random.normal(0, 1.2, size=30)

phase\_B, mem\_B = braid(phase\_B, mem\_B, [0,1], chirality=1.1, steps=20)

# Memory values for logic states

memval\_A = np.tanh(mem\_A.mean())

memval\_B = np.tanh(mem\_B.mean())

difference = memval\_B - memval\_A

# Plot

fig, axs = plt.subplots(1, 3, figsize=(13, 4))

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

axs[0].set\_title(f"Final MBT Phase Field (A→B)\nMemory: {memval\_A:.3f}")

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

axs[1].set\_title(f"Final MBT Phase Field (B→A)\nMemory: {memval\_B:.3f}")

axs[2].bar(['A then B', 'B then A'], [memval\_A, memval\_B], color=['c', 'm'])

axs[2].set\_ylabel("MBT Memory (logic state)")

axs[2].set\_title(f"Difference: {difference:.3f}\n(non-Abelian logic, noisy)")

plt.tight\_layout()

plt.show()

print(f"Memory after A then B (noisy): {memval\_A:.4f}")

print(f"Memory after B then A (noisy): {memval\_B:.4f}")

print(f"Difference (robust logic): {difference:.4e}")