# MBT Superconductor: Persistent Current Simulation (Colab-friendly)

import numpy as np

import matplotlib.pyplot as plt

# --- Parameters ---

N = 32 # Grid size (NxN)

steps = 400 # Time steps

noise\_levels = np.linspace(0.0, 2.5, 16) # 'Temperature' range

twist\_phase = np.pi/2 # Injected phase twist (persistent current)

decay\_times = []

def mbt\_sim(noise, plot\_example=False):

# Initialize phases randomly

phase = np.random.rand(N, N) \* 2 \* np.pi

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

# Inject phase twist ("current") at the left edge

phase[:, 0] += twist\_phase

# Track average phase difference across system (as a proxy for current)

avg\_phase\_diff = []

for t in range(steps):

# MBT phase update: nearest-neighbor alignment + memory + noise

nbrs = (

np.roll(phase, 1, axis=0) + np.roll(phase, -1, axis=0) +

np.roll(phase, 1, axis=1) + np.roll(phase, -1, axis=1)

)

phase\_force = 0.22 \* np.sin(nbrs - 4 \* phase)

noise\_term = noise \* np.random.randn(N, N)

mem = 0.88 \* mem + 0.12 \* np.cos(phase)

phase = (phase + phase\_force + noise\_term + 0.04 \* mem) % (2 \* np.pi)

# Track phase difference left vs right

left = np.mean(np.cos(phase[:, 0]))

right = np.mean(np.cos(phase[:, -1]))

avg\_phase\_diff.append(np.abs(left - right))

# Optional plot

if plot\_example:

fig, axs = plt.subplots(1, 3, figsize=(16, 4))

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

axs[0].set\_title("Final MBT Phase Field (cos)")

axs[1].imshow(mem, cmap='inferno')

axs[1].set\_title("Final MBT Memory Field")

axs[2].plot(avg\_phase\_diff, label="Current Persistence")

axs[2].set\_title("Current Decay vs Time")

axs[2].set\_xlabel("Timestep")

axs[2].set\_ylabel("Avg Phase Difference")

axs[2].legend()

plt.tight\_layout()

plt.show()

# Determine how long the current survives above half its initial value

init = avg\_phase\_diff[0]

decay\_idx = next((i for i, v in enumerate(avg\_phase\_diff) if v < 0.5\*init), steps-1)

return decay\_idx

# --- Main Sweep: Persistent current lifetime vs temperature ---

for i, noise in enumerate(noise\_levels):

if i == 2: # Plot a low-T example

print(f"Example: Low T={noise:.2f}")

t\_decay = mbt\_sim(noise, plot\_example=True)

elif i == len(noise\_levels)-2: # Plot a high-T example

print(f"Example: High T={noise:.2f}")

t\_decay = mbt\_sim(noise, plot\_example=True)

else:

t\_decay = mbt\_sim(noise)

decay\_times.append(t\_decay)

# --- Plot: Lifetime of persistent current vs temperature ---

plt.figure(figsize=(8,4))

plt.plot(noise\_levels, decay\_times, marker='o', label="Current Lifetime")

plt.axvline(noise\_levels[np.argmax(np.diff(decay\_times))], color='k', ls='--', label='Critical Point?')

plt.title("MBT Persistent Current Lifetime (Superconductivity Test)")

plt.xlabel("Temperature (noise, ↑)")

plt.ylabel("Current Lifetime (timesteps)")

plt.legend()

plt.tight\_layout()

plt.show()