# MBT Superconductivity Transition (First Principles)

import numpy as np

import matplotlib.pyplot as plt

# Parameters

N = 32 # Lattice/grid size

timesteps = 150 # Iterations per temperature

temps = np.linspace(2.5, 0.05, 20) # "Cooling" schedule (high T → low T)

coupling = 1.0 # MBT interaction strength

# MBT Field Initialization

phase = np.random.uniform(0, 2\*np.pi, (N, N)) # Random phase field

memory = np.zeros((N, N)) # "Echo memory" (plasticity)

# Storage for plotting

order\_vs\_T = []

for T in temps:

for step in range(timesteps):

for i in range(N):

for j in range(N):

# Neighbour coordinates (periodic boundary)

nb = [

((i+1)%N, j), ((i-1)%N, j),

(i, (j+1)%N), (i, (j-1)%N)

]

# Local MBT phase update: weighted by neighbours & memory

sum\_sin = sum(np.sin(phase[nn]-phase[i,j]) for nn in nb)

dtheta = (coupling \* sum\_sin + 0.12 \* memory[i,j]) / (T + 0.2)

# Noise (temperature): hotter = more randomness

phase[i,j] += dtheta + np.random.normal(0, T\*0.1)

# Update memory field (MBT echo: favors stability)

memory[i,j] = 0.92\*memory[i,j] + 0.08\*np.cos(sum\_sin)

# Calculate MBT "order parameter" (coherence, like superconducting gap)

order = np.abs(np.mean(np.exp(1j\*phase)))

order\_vs\_T.append(order)

# Find critical temperature (where order rises sharply)

order\_vs\_T = np.array(order\_vs\_T)

Tc\_idx = np.argmax(np.gradient(order\_vs\_T))

Tc = temps[Tc\_idx]

# Final visualizations

fig, axes = plt.subplots(1, 3, figsize=(18, 4))

# Show final phase field (cosine: highlights domains)

axes[0].imshow(np.cos(phase), cmap='twilight', interpolation='nearest')

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

axes[0].axis('off')

# Show memory field (echo imprint)

axes[1].imshow(memory, cmap='inferno', interpolation='nearest')

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

axes[1].axis('off')

# Plot order parameter vs temperature

axes[2].plot(temps, order\_vs\_T, label='Coherence (Order)')

axes[2].axvline(Tc, color='k', linestyle='--', label='Critical Point?')

axes[2].set\_xlabel("Temperature (noise, ↓)")

axes[2].set\_ylabel("Order")

axes[2].set\_title("Disorder → Order (MBT Superconductivity)")

axes[2].legend()

plt.tight\_layout()

plt.show()