import numpy as np

import matplotlib.pyplot as plt

# Parameters

N = 5000 # Number of trials (big enough for smooth stats)

theta\_A = 0 # Measurement setting A (angle, radians)

theta\_Ap = np.pi/4 # A'

theta\_B = np.pi/8 # B

theta\_Bp = 3\*np.pi/8 # B'

def mbt\_hidden\_phase():

"""Generate a shared field 'phase' as MBT would (here: truly random for test, but could use memory field)."""

return np.random.uniform(0, 2\*np.pi)

def measure(phase, theta):

"""Returns +1 or -1 depending on alignment of field phase and measurement angle."""

return 1 if np.cos(phase - theta) > 0 else -1

# Run Bell/CHSH test

def run\_bell(N, angles):

"""Run the full set of Bell tests for all settings."""

results = {k: [] for k in ['AB', 'ABp', 'ApB', 'ApBp']}

for \_ in range(N):

phi = mbt\_hidden\_phase() # Shared memory/phase

# All measurement settings

A = measure(phi, angles['A'])

Ap = measure(phi, angles['Ap'])

B = measure(phi, angles['B'])

Bp = measure(phi, angles['Bp'])

results['AB'].append(A\*B)

results['ABp'].append(A\*Bp)

results['ApB'].append(Ap\*B)

results['ApBp'].append(Ap\*Bp)

return results

angles = {'A': theta\_A, 'Ap': theta\_Ap, 'B': theta\_B, 'Bp': theta\_Bp}

res = run\_bell(N, angles)

# Compute averages (correlation for each setting)

E\_AB = np.mean(res['AB'])

E\_ABp = np.mean(res['ABp'])

E\_ApB = np.mean(res['ApB'])

E\_ApBp = np.mean(res['ApBp'])

# Bell-CHSH parameter

S = abs(E\_AB - E\_ABp + E\_ApB + E\_ApBp)

print(f"Bell-CHSH S = {S:.3f}")

print("(Classical limit = 2.0, Quantum max = 2.828)")

# Plot the correlation as function of angle

angles\_list = np.linspace(0, np.pi, 100)

corrs = [np.mean([measure(mbt\_hidden\_phase(), 0)\*measure(mbt\_hidden\_phase(), th) for \_ in range(200)]) for th in angles\_list]

plt.figure(figsize=(7,4))

plt.plot(angles\_list\*180/np.pi, corrs, label='MBT Field Correlation')

plt.axhline(0, color='gray', lw=0.8)

plt.xlabel('Angle between detectors (degrees)')

plt.ylabel('Correlation')

plt.title('MBT Bell Test Correlation vs. Angle')

plt.legend()

plt.tight\_layout()

plt.show()