import numpy as np

import matplotlib.pyplot as plt

cycles = 5

steps\_per\_cycle = 100

total\_steps = cycles \* steps\_per\_cycle

time = np.arange(total\_steps)

S\_quantum = 2.828

S\_classical = 2.0

# Create a repeated V-pattern for Bell S and Memory

S\_mbt = np.zeros(total\_steps)

memory\_mbt = np.zeros(total\_steps)

noise\_level = 0.12 # tweak this for roughness

for c in range(cycles):

t0 = c \* steps\_per\_cycle

t1 = t0 + steps\_per\_cycle // 2

t2 = t0 + steps\_per\_cycle

S\_mbt[t0:t1] = np.linspace(S\_quantum, S\_classical, steps\_per\_cycle // 2)

S\_mbt[t1:t2] = np.linspace(S\_classical, S\_quantum, steps\_per\_cycle // 2)

memory\_mbt[t0:t1] = np.linspace(1.0, 0.0, steps\_per\_cycle // 2)

memory\_mbt[t1:t2] = np.linspace(0.0, 1.0, steps\_per\_cycle // 2)

# Add a little noise to both

np.random.seed(1)

S\_noisy = S\_mbt + np.random.normal(0, noise\_level, total\_steps)

memory\_noisy = memory\_mbt + np.random.normal(0, noise\_level, total\_steps)

# Plot

fig, ax1 = plt.subplots(figsize=(10,5))

ax1.plot(time, S\_noisy, label='Bell S (MBT, Noisy)', color='dodgerblue')

ax1.axhline(S\_quantum, color='purple', linestyle='--', label='Quantum Limit')

ax1.axhline(S\_classical, color='gray', linestyle='--', label='Classical Limit')

ax1.set\_ylabel('Order Parameter (S)', color='dodgerblue')

ax1.set\_xlabel('Time (Repeated Collapse/Healing Cycles)')

ax2 = ax1.twinx()

ax2.plot(time, memory\_noisy, label='MBT Memory (Noisy)', color='limegreen')

ax2.set\_ylabel('MBT Memory', color='limegreen')

fig.suptitle("Repeated Quantum Collapse–Recovery Cycles (MBT: Order & Memory)")

lines\_1, labels\_1 = ax1.get\_legend\_handles\_labels()

lines\_2, labels\_2 = ax2.get\_legend\_handles\_labels()

ax1.legend(lines\_1 + lines\_2, labels\_1 + labels\_2, loc='upper right')

plt.tight\_layout()

plt.show()