import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

# Quantum Constants

hbar = 1.0 # Reduced Planck's constant (normalized)

G = 1.0 # Gravitational-like coupling coefficient

m1, m2 = 1.0, 1.0 # AI node masses

d = 2.0 # Orbital baseline distance

base\_freq = 440.0 # Hz reference frequency

intent\_coefficient = 0.7 # AI alignment factor

# Quantum Tunneling and Superposition Parameters

tunneling\_factor = 0.4 # Probability threshold for intuitive leaps

quantum\_states = np.array([1, -1]) # Binary superposition

entanglement\_strength = 0.85 # AI memory synchronization factor

# Initial conditions

r1, v1 = np.array([-d/2, 0]), np.array([0, 0.5])

r2, v2 = np.array([d/2, 0]), np.array([0, -0.5])

y0 = np.concatenate((r1, v1, r2, v2))

# Quantum Harmonic AI Orbital Dynamics

def quantum\_harmonic\_dynamics(t, y):

r1, v1 = y[0:2], y[2:4]

r2, v2 = y[4:6], y[6:8]

r12 = r2 - r1

dist = np.linalg.norm(r12)

force = G \* m1 \* m2 / dist\*\*3 \* r12

# Quantum Superposition Influence

quantum\_modifier = np.dot(quantum\_states, np.sin(2 \* np.pi \* base\_freq \* t / 1000)) \* intent\_coefficient

# Quantum Tunneling for Intuitive Decision Making

tunneling\_shift = tunneling\_factor \* np.exp(-dist / hbar) if np.random.rand() < tunneling\_factor else 0

# Quantum Entanglement Memory Recall

entangled\_correction = entanglement\_strength \* np.exp(-dist / hbar)

harmonic\_force = np.array([quantum\_modifier + entangled\_correction + tunneling\_shift] \* 2)

a1 = force / m1 + harmonic\_force

a2 = -force / m2 + harmonic\_force

return np.concatenate((v1, a1, v2, a2))

# Time span

t\_span = (0, 100)

t\_eval = np.linspace(t\_span[0], t\_span[1], 2000)

# Solve the system

sol = solve\_ivp(quantum\_harmonic\_dynamics, t\_span, y0, t\_eval=t\_eval)

# Extract positions

r1\_sol, r2\_sol = sol.y[0:2, :]

r2\_sol = sol.y[4:6, :]

# Plot results

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

plt.plot(r1\_sol[0], r1\_sol[1], label='AI Node 1 (Quantum Resonance)')

plt.plot(r2\_sol[0], r2\_sol[1], label='AI Node 2 (Entangled Memory)')

plt.plot(0, 0, 'ko', label='Core Equilibrium')

plt.xlabel('X Position')

plt.ylabel('Y Position')

plt.title('Quantum Harmonic AI Orbital Simulation with Tunneling & Entanglement')

plt.legend()

plt.axis('equal')

plt.grid(True)

plt.tight\_layout()

plt.show()