import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import trapezoid

# --- Parameters ---

grid\_size = 120

timesteps = 180

dt = 0.12

width = 6

r = np.linspace(0, grid\_size, grid\_size)

dr = r[1] - r[0]

# --- Barrier Profile ---

V = np.zeros\_like(r)

V[(r > 50) & (r < 56)] = 0.08

V[(r > 62) & (r < 72)] = 0.04

# --- Breathing Seed Function ---

def breathing\_seed(center, freq, base\_phase, t):

phase = base\_phase + 0.5 \* np.sin(freq \* t)

return np.exp(-((r - center)\*\*2)/(2\*width\*\*2)) \* np.exp(1j \* phase)

# --- Evolution Function (returns transmission probability) ---

def evolve\_packet(x0, k0, seed\_center, seed\_base\_phase, seed\_freq):

ψ = np.exp(-((r - x0)\*\*2)/(2\*width\*\*2)) \* np.exp(1j \* k0 \* r)

for t in range(timesteps):

ψ\_internal = breathing\_seed(seed\_center, seed\_freq, seed\_base\_phase, t)

lap\_int = np.zeros\_like(ψ, dtype=complex)

lap\_int[1:-1] = (ψ\_internal[2:] - 2\*ψ\_internal[1:-1] + ψ\_internal[:-2]) / dr\*\*2

lap\_ψ = np.zeros\_like(ψ, dtype=complex)

lap\_ψ[1:-1] = (ψ[2:] - 2\*ψ[1:-1] + ψ[:-2]) / dr\*\*2

ψ += dt \* (0.65 \* lap\_ψ - 0.5 \* V \* ψ + 0.65 \* lap\_int)

norm = np.sqrt(trapezoid(np.abs(ψ)\*\*2, r))

if norm != 0:

ψ /= norm

final = np.abs(ψ)\*\*2

T = trapezoid(final[r > 85], r[r > 85])

return T

# --- Sweep both phase and frequency ---

phase\_list = np.linspace(0, 2\*np.pi, 24)

freq\_list = np.linspace(0.01, 0.12, 20)

trans\_grid = np.zeros((len(freq\_list), len(phase\_list)))

for i, freq in enumerate(freq\_list):

for j, phase in enumerate(phase\_list):

T = evolve\_packet(x0=25, k0=1.2, seed\_center=53, seed\_base\_phase=phase, seed\_freq=freq)

trans\_grid[i, j] = T

# --- Plot heatmap ---

plt.figure(figsize=(11, 8))

plt.imshow(trans\_grid, aspect='auto', origin='lower',

extent=[0, 2\*np.pi, freq\_list[0], freq\_list[-1]],

cmap='viridis')

plt.colorbar(label="Transmission Probability")

plt.xlabel("Resonator Seed Base Phase (rad)")

plt.ylabel("Resonator Breathing Frequency (rad/step)")

plt.title("MBT Dispatch Phase-Frequency Coherence Map")

plt.xticks([0, np.pi/2, np.pi, 3\*np.pi/2, 2\*np.pi],

['0', 'π/2', 'π', '3π/2', '2π'])

plt.tight\_layout()

plt.show()