import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import trapezoid

# --- Grid and Parameters ---

grid\_size = 120

timesteps = 180

dt = 0.12

width = 6

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

dr = r[1] - r[0]

# --- 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)

# --- Oscillating Barrier Function ---

def modulated\_barrier(t, freq, base\_amp=0.08, secondary\_amp=0.04):

modulation = 0.5 + 0.5 \* np.sin(freq \* t) # Ranges from 0 to 1

V = np.zeros\_like(r)

V[(r > 50) & (r < 56)] = base\_amp \* modulation # Left bump breathes

V[(r > 62) & (r < 72)] = secondary\_amp \* (1 - modulation) # Right bump anti-breathes

return V

# --- Evolution Function ---

def evolve\_resonant(seed\_center=67, x0=95, k0=-1.2, freq=0.06, base\_phase=np.pi):

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

ψ\_total = []

for t in range(timesteps):

ψ\_internal = breathing\_seed(seed\_center, freq, base\_phase, t)

V = modulated\_barrier(t, freq) # Oscillating barrier in sync

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

if t % 15 == 0:

ψ\_total.append(np.abs(ψ)\*\*2)

final = np.abs(ψ)\*\*2

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

R = trapezoid(final[r < 35], r[r < 35])

return ψ\_total, T, R

# --- Run the experiment ---

snap\_sync, T\_sync, R\_sync = evolve\_resonant()

# --- Plot Results ---

fig, axs = plt.subplots(2, 2, figsize=(15,8))

# Evolution snapshots

for ψs in snap\_sync:

axs[0,0].plot(r, ψs, alpha=0.5)

axs[0,0].plot(r, modulated\_barrier(0, 0.06) / 0.08 \* np.max([np.max(p) for p in snap\_sync]), 'k--', lw=2)

axs[0,0].set\_title("Dispatch Duet: Seed + Barrier in Rhythmic Sync")

# Transmission

axs[0,1].bar(['→Forward'], [T\_sync], color='mediumspringgreen')

axs[0,1].set\_ylim(0, 1)

axs[0,1].set\_ylabel("Transmission Probability")

axs[0,1].set\_title("Access with Harmonic Landscape")

# Reflection

axs[1,1].bar(['→Forward'], [R\_sync], color='orange')

axs[1,1].set\_ylim(0, 1)

axs[1,1].set\_ylabel("Reflection Probability")

axs[1,1].set\_title("Rejection with Harmonic Landscape")

axs[1,0].axis('off') # Unused subplot

plt.tight\_layout()

plt.show()