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)

barrier\_left = (r > 50) & (r < 56) # Sharp bump

barrier\_right = (r > 62) & (r < 72) # Wide bump

V[barrier\_left] = 0.08

V[barrier\_right] = 0.04

# --- Breathing dispatch seed ---

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 with Breathing Internal Seed ---

def evolve\_packet(x0, k0, center\_seed, freq, base\_phase):

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

ψ\_total = []

for t in range(timesteps):

# Inject internal curvature seed with breathing phase

ψ\_internal = breathing\_seed(center=center\_seed, freq=freq, base\_phase=base\_phase, t=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

# Evolve ψ with internal curvature added

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

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

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

# Normalize

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 Forward (Left→Right): Favorable curvature seed ---

snap\_fwd, T\_fwd, R\_fwd = evolve\_packet(x0=25, k0=1.2, center\_seed=53, freq=0.06, base\_phase=0)

# --- Run Reverse (Right→Left): Phase-mismatched curvature seed ---

snap\_rev, T\_rev, R\_rev = evolve\_packet(x0=95, k0=-1.2, center\_seed=67, freq=0.06, base\_phase=np.pi/2)

# --- Plot Results ---

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

# Forward evolution

for ψs in snap\_fwd:

axs[0,0].plot(r, ψs, alpha=0.5)

axs[0,0].plot(r, V / np.max(V) \* np.max([np.max(p) for p in snap\_fwd]), 'k--', lw=2, label='Barrier')

axs[0,0].set\_title("Breathing Diode: Forward (Left→Right)")

axs[0,0].legend()

# Reverse evolution

for ψs in snap\_rev:

axs[0,1].plot(r, ψs, alpha=0.5)

axs[0,1].plot(r, V / np.max(V) \* np.max([np.max(p) for p in snap\_rev]), 'k--', lw=2, label='Barrier')

axs[0,1].set\_title("Breathing Diode: Reverse (Right→Left)")

axs[0,1].legend()

# Transmission bars

axs[1,0].bar(['→Forward', '←Reverse'], [T\_fwd, T\_rev], color=['lightgreen', 'lightblue'])

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

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

axs[1,0].set\_title("Directional Dispatch Tunneling")

# Reflection bars

axs[1,1].bar(['→Forward', '←Reverse'], [R\_fwd, R\_rev], color=['gold', 'red'])

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

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

axs[1,1].set\_title("Directional Dispatch Reflection")

plt.tight\_layout()

plt.show()