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: dual bumps ---

V = np.zeros\_like(r)

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

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

# --- 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 with two internal seeds ---

def evolve\_packet(x0, k0, centers, phases, freqs):

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

ψ\_total = []

for t in range(timesteps):

# Sum Laplacians of both seeds

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

for c, p, f in zip(centers, phases, freqs):

ψ\_seed = breathing\_seed(c, f, p, t)

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

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

lap\_total += lap

# Evolve ψ with barrier and seeds

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\_total)

# 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

# --- Twin seed parameters ---

centers = [53, 67]

phases = [0, np.pi/2]

freqs = [0.06, 0.06]

# --- Run Forward (L→R) and Reverse (R→L) ---

snap\_fwd, T\_fwd, R\_fwd = evolve\_packet(x0=25, k0=1.2, centers=centers, phases=phases, freqs=freqs)

snap\_rev, T\_rev, R\_rev = evolve\_packet(x0=95, k0=-1.2, centers=centers, phases=phases, freqs=freqs)

# --- Plot ---

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)

axs[0,0].set\_title("Twin Seed Dispatch Diode: Forward Evolution")

# 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)

axs[0,1].set\_title("Twin Seed Dispatch Diode: Reverse Evolution")

# Transmission comparison

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 Tunneling (Twin Seeds)")

# Reflection comparison

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 Reflection (Twin Seeds)")

plt.tight\_layout()

plt.show()