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]

# --- Moving Barrier parameters ---

barrier\_width = 6

barrier\_height = 0.06

initial\_barrier\_center = 55

barrier\_speed = 0.22 # grid units per time step (tune for fun: 0.1–0.4)

# --- Breathing MBT Seed parameters ---

seed\_center = 53

seed\_freq = 0.06

seed\_base\_phase = 0

# --- Inbound packet ---

x0 = 25

k0 = 1.2

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

# --- Storage ---

ψ\_total = []

snapshot\_times = np.arange(0, timesteps, 15)

ψ\_snapshots = []

for t in range(timesteps):

# --- Moving barrier: update position ---

barrier\_center = initial\_barrier\_center + barrier\_speed \* t

V = np.zeros\_like(r)

left\_edge = barrier\_center - barrier\_width/2

right\_edge = barrier\_center + barrier\_width/2

V[(r > left\_edge) & (r < right\_edge)] = barrier\_height

# --- MBT Internal Seed (dispatch) ---

phase = seed\_base\_phase + 0.5 \* np.sin(seed\_freq \* t)

ψ\_internal = np.exp(-((r - seed\_center)\*\*2)/(2\*width\*\*2)) \* np.exp(1j \* phase)

# --- Add MBT internal curvature to Laplacian ---

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)

# Normalize

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

if norm != 0:

ψ /= norm

if t in snapshot\_times:

ψ\_total.append(np.abs(ψ)\*\*2)

# --- Transmission & Reflection ---

final = np.abs(ψ)\*\*2

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

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

# --- Plot Results ---

fig, axs = plt.subplots(2, 1, figsize=(12,8))

# 1. Wavepacket Evolution with Moving Barrier

for ψs in ψ\_total:

axs[0].plot(r, ψs, alpha=0.5)

# Draw moving barrier trajectory

for t, ψs in zip(snapshot\_times, ψ\_total):

barrier\_c = initial\_barrier\_center + barrier\_speed \* t

left = barrier\_c - barrier\_width/2

right = barrier\_c + barrier\_width/2

# Draw barrier as vertical lines

axs[0].axvline(left, color='k', linestyle='dashed', alpha=0.25)

axs[0].axvline(right, color='k', linestyle='dashed', alpha=0.25)

axs[0].set\_title("MBT Dispatch Tunneling with Moving Barrier")

axs[0].set\_ylabel("Probability Density")

# 2. Transmission and Reflection Bars

axs[1].bar(['Transmission', 'Reflection'], [T, R], color=['lightgreen', 'gold'])

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

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

axs[1].set\_title("Final Probabilities")

plt.tight\_layout()

plt.show()