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]

# --- Sweep Ranges ---

phase\_list = np.linspace(0, 2\*np.pi, 20)

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

transmission\_matrix = np.zeros((len(freq\_list), len(phase\_list)))

# --- SCANNER LOOP ---

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

for pi, phase\_offset in enumerate(phase\_list):

# --- ψ₁: Inbound packet ---

x1, k1 = 25, 1.4

ψ1 = np.exp(-((r - x1)\*\*2)/(2\*width\*\*2)) \* np.exp(1j \* k1 \* r)

# --- ψ₂: Resonator inside barrier ---

x2 = 60

ψ2 = np.exp(-((r - x2)\*\*2)/(2\*width\*\*2)) \* np.exp(1j \* phase\_offset)

ψ = np.array([ψ1, ψ2])

n = len(ψ)

# --- Barrier ---

barrier\_center = 60

barrier\_width, barrier\_height = 6, 0.06

V = np.zeros\_like(r)

V[(r > barrier\_center - barrier\_width/2) & (r < barrier\_center + barrier\_width/2)] = barrier\_height

# --- Evolution Loop ---

for t in range(timesteps):

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

ψ[1] = np.exp(-((r - x2)\*\*2)/(2\*width\*\*2)) \* np.exp(1j \* current\_phase)

for i in range(n):

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

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

ψ[i] += dt \* (0.65 \* lap - 0.5 \* V \* ψ[i])

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

if norm != 0:

ψ[i] /= norm

# --- Transmission ---

final = np.abs(np.sum(ψ, axis=0))\*\*2

trans\_region = r > (barrier\_center + barrier\_width/2)

P\_trans = trapezoid(final[trans\_region], r[trans\_region])

transmission\_matrix[fi, pi] = P\_trans

# --- Plot Heatmap ---

fig, ax = plt.subplots(figsize=(10, 8))

im = ax.imshow(transmission\_matrix, origin='lower', aspect='auto',

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

cmap='viridis')

ax.set\_xlabel("Resonator Phase Offset (rad)")

ax.set\_ylabel("Resonator Frequency (rad/step)")

ax.set\_title("MBT Dispatch Tunneling — Coherence Surface")

cbar = plt.colorbar(im, ax=ax)

cbar.set\_label("Transmission Probability")

plt.tight\_layout()

plt.show()