import numpy as np

import matplotlib.pyplot as plt

from scipy.ndimage import laplace

# Parameters

N = 60 # grid size (NxN)

timesteps = 120 # number of time steps

dt = 0.19 # time step

width = 3.5 # initial packet width

# Quantum percolation barrier grid (randomized)

np.random.seed(42)

V = np.zeros((N, N))

barrier\_prob = 0.27 # fraction of sites with a barrier

# Generate mask ONCE, and random barrier heights to match

barrier\_mask = np.random.rand(N, N) < barrier\_prob

random\_heights = 1.2 + 0.6 \* np.random.rand(np.sum(barrier\_mask))

V[barrier\_mask] = random\_heights

# Initial wavepacket (start on left edge)

psi = np.zeros((N, N), dtype=complex)

x0 = 6

y0 = N // 2

for i in range(N):

for j in range(N):

psi[i, j] = np.exp(-((i - x0)\*\*2 + (j - y0)\*\*2) / (2 \* width\*\*2)) \* np.exp(1j \* 0.7 \* i)

# Main time evolution loop

for t in range(timesteps):

lap = laplace(psi)

psi += dt \* (0.7 \* lap - 0.65 \* V \* psi)

# Normalize at each step

norm = np.sqrt(np.sum(np.abs(psi)\*\*2))

if norm != 0:

psi /= norm

# Plot: Final probability density

plt.figure(figsize=(8,7))

plt.imshow(np.abs(psi)\*\*2, origin='lower', cmap='viridis', extent=[0,N,0,N])

plt.title("2D MBT Quantum Percolation — Final Probability Density")

plt.xlabel("X")

plt.ylabel("Y")

plt.colorbar(label="|ψ|²")

# Overlay barriers (as black dots)

by, bx = np.where(V > 0.1)

plt.scatter(bx, by, c='black', s=8, label='Barrier')

plt.legend()

plt.tight\_layout()

plt.show()

# Optional: Calculate transmission (prob. on rightmost edge)

T = np.sum(np.abs(psi[-1,:])\*\*2)

print(f"Transmission probability (right edge): {T:.4f}")