Phase 10 – Part 4  
Simulation Recipes for ψ-Wave Propagation & Spectral Analysis

Goal  
The purpose of this part is to translate the linearized ψ-wave dynamics into explicit simulation schemes. I want to numerically evolve perturbations φ(x, t), observe their stability or instability, and perform spectral analysis to extract dispersion behavior directly from simulated data. This will bridge the analytic dispersion relation (Part 2) and classification framework (Part 3) with concrete computational demonstrations.

Governing Equation  
From Part 2, the perturbation field satisfies the wave-like equation:

Plain text:  
∂²φ/∂t² = −C₀ ∇²φ(x, t)

with dispersion relation:

Plain text:  
ω² = C₀ k²

Simulation Strategy

* **Discretize space:** Represent φ(x, t) on a 1D grid.
* **Initialize perturbation:** Choose sinusoidal, Gaussian, or random initial φ(x).
* **Time integration:** Use finite-difference or spectral methods to evolve φ under the wave equation.
* **Stability control:** Ensure numerical CFL (Courant–Friedrichs–Lewy) condition is satisfied:

Plain text:  
Δt < Δx / √(C₀)

* **Spectral analysis:** Compute Fourier transform of φ(x, t) to recover ω–k relation.

Finite Difference Implementation (1D)

# simulations/phase10\_part4\_wave\_propagation.py  
import numpy as np  
import matplotlib.pyplot as plt  
  
# Parameters  
L = 50.0 # domain length  
N = 500 # grid points  
dx = L / N  
x = np.linspace(0, L, N, endpoint=False)  
  
C0 = 1.0 # curvature factor (positive = stable waves)  
dt = 0.1 \* dx / np.sqrt(C0) # stable timestep  
T = 200 # number of time steps  
  
# Initialize perturbation: Gaussian ripple  
phi = np.exp(-(x - L/2)\*\*2 / 2.0)  
phi\_old = np.copy(phi) # initial condition at t - dt  
phi\_new = np.zeros\_like(phi)  
  
# Store history for spectral analysis  
history = []  
  
# Finite difference evolution  
for n in range(T):  
 # Laplacian (periodic BCs)  
 laplacian = (np.roll(phi, -1) - 2\*phi + np.roll(phi, 1)) / dx\*\*2  
 # Wave equation update  
 phi\_new = 2\*phi - phi\_old + (C0 \* dt\*\*2) \* laplacian  
   
 # Update steps  
 phi\_old = phi  
 phi = phi\_new  
   
 if n % 5 == 0:  
 history.append(np.copy(phi))  
  
# Plot final snapshot  
plt.plot(x, phi, label="φ(x, T)")  
plt.title("Phase 10 Part 4: ψ-Wave Propagation (Stable Case)")  
plt.xlabel("x")  
plt.ylabel("φ(x)")  
plt.legend()  
plt.show()