# Phase 5 — Part 5.10: Dimensional Extension (1D → 2D Test)

## Goal

I extend ψ–gravity from a one-dimensional lattice to a two-dimensional periodic grid.  
This allows me to probe how ψ-curvature feedback behaves when embedded in higher dimensions. The immediate objective is to see whether ψ forms stable “lumps,” ridges, or spreads smoothly under 2D Laplacian coupling. This lays the foundation for higher-dimensional analyses (Phase 7).

## Core ψ–Gravity Statement (2D)

I retain the definition:

Plain text:  
Gravity(x,y) = (∇²[space(x,y) + current(x,y)²]) × ψ(x,y)

The corresponding force field is:

Plain text:  
Force(x,y) = −∇[Gravity(x,y)]

## 2D Laplacian Operator

For a scalar field :

Plain text:  
∇²f(x,y) = ∂²f/∂x² + ∂²f/∂y²

## 2D Energy Landscape

I define the effective ψ energy density as:

Plain text:  
E(x,y) = 0.5 × |∇ψ(x,y)|² + 0.5 × Gravity(x,y) × ψ(x,y)

This provides a spatial map of ψ stability:  
- wells = confinement zones,  
- ridges = unstable boundaries,  
- plateaus = neutral spread regions.

## Numerical Simulation Setup

* Grid: square domain , lattice
* Boundary: periodic (wave wraps around domain)
* Initial ψ: Gaussian bump centered at
* Background:

Gravity is computed as Laplacian of , scaled by ψ.  
Force is the gradient of Gravity.  
Energy density combines ψ gradients and ψ–gravity coupling.

## Reference Python Implementation (2D Test)

# -----------------------------  
# Phase 5.10 — ψ extension into 2D lattice  
# -----------------------------  
import numpy as np  
import matplotlib.pyplot as plt  
  
# Grid setup  
N = 100  
x = np.linspace(0, 2\*np.pi, N)  
y = np.linspace(0, 2\*np.pi, N)  
X, Y = np.meshgrid(x, y)  
dx = x[1] - x[0]  
  
# Initial ψ field (Gaussian bump at center)  
psi = np.exp(-((X-np.pi)\*\*2 + (Y-np.pi)\*\*2) / 0.3)  
  
# Define background fields  
space = np.sin(X) \* np.cos(Y)  
current = np.cos(X) \* np.sin(Y)  
  
# 2D Laplacian (periodic boundaries)  
def laplacian(Z, dx):  
 return (  
 -4\*Z  
 + np.roll(Z, 1, axis=0) + np.roll(Z, -1, axis=0)  
 + np.roll(Z, 1, axis=1) + np.roll(Z, -1, axis=1)  
 ) / dx\*\*2  
  
# Compute Gravity field  
gravity = laplacian(space + current\*\*2, dx) \* psi  
  
# Force field (gradients of −Gravity)  
Fx, Fy = np.gradient(-gravity, dx, dx)  
  
# Energy density  
grad\_psi\_x, grad\_psi\_y = np.gradient(psi, dx, dx)  
energy = 0.5\*(grad\_psi\_x\*\*2 + grad\_psi\_y\*\*2) + 0.5\*gravity\*psi  
  
# Plot ψ, Gravity, and Energy  
fig, axs = plt.subplots(1, 3, figsize=(15,5))  
axs[0].imshow(psi, extent=[0,2\*np.pi,0,2\*np.pi], origin='lower')  
axs[0].set\_title("ψ Field (2D)")  
  
axs[1].imshow(gravity, extent=[0,2\*np.pi,0,2\*np.pi], origin='lower')  
axs[1].set\_title("Gravity(x,y)")  
  
axs[2].imshow(energy, extent=[0,2\*np.pi,0,2\*np.pi], origin='lower')  
axs[2].set\_title("Energy Density E(x,y)")  
  
plt.show()