# Phase 5.13 — Numerical Phase Diagram

In this phase, I construct a numerical phase diagram for ψ–gravity dynamics.  
The goal is to map how ψ behavior changes across parameter space: amplitude , width , and coupling strength .  
This allows me to distinguish **stable**, **collapsing**, and **dispersive** regimes for emergent ψ structures.

## Core ψ-Gravity Equation (Reminder)

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

Force derived as:

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

## Parameters of Interest

* **Amplitude :** Controls the strength of ψ perturbations.
* **Width :** Determines localization vs delocalization.
* **Coupling :** Effective strength of ψ–gravity feedback.

## Classification of Regimes

* **Stable lumps:** ψ remains localized with finite width.
* **Collapse:** ψ focuses into a singular peak (runaway confinement).
* **Dispersion:** ψ spreads out and decays.

## Energy Balance Criterion

The total energy is defined as:

Plain text:  
E\_total = ∫ [ ½|∇ψ|² + (g/2)·Gravity(x)·ψ ] dx

* If gradient energy dominates → ψ **disperses**.
* If gravity dominates → ψ **collapses**.
* If they balance → ψ forms a **stable lump**.

## Numerical Simulation: Parameter Scan

I perform a parameter sweep over , , and to construct the phase diagram.

# -----------------------------  
# simulations/phase5\_part5\_13\_ψ-Phase\_Diagram.py  
# Phase 5.13 — ψ-gravity phase diagram  
# -----------------------------  
  
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]  
  
# Define background fields  
space = np.sin(X) \* np.cos(Y)  
current = np.cos(X) \* np.sin(Y)  
  
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  
  
# Phase classification function  
def classify(A, sigma, g):  
 # Initial Gaussian lump  
 psi = A \* np.exp(-((X-np.pi)\*\*2 + (Y-np.pi)\*\*2) / sigma\*\*2)  
   
 # Gravity field  
 gravity = laplacian(space + current\*\*2, dx) \* psi  
   
 # Gradient energy  
 grad\_x, grad\_y = np.gradient(psi, dx, dx)  
   
 # Total energy density  
 energy = 0.5\*(grad\_x\*\*2 + grad\_y\*\*2) + 0.5\*g\*gravity\*psi  
 E\_total = np.sum(energy)  
   
 # Heuristic classification  
 if E\_total > 50:  
 return "collapse"  
 elif E\_total < -5:  
 return "dispersion"  
 else:  
 return "stable"  
  
# Parameter scan  
A\_values = [0.5, 1.0, 1.5, 2.0]  
sigma\_values = [0.3, 0.5, 0.8, 1.2]  
g\_values = [0.5, 1.0, 1.5, 2.0]  
  
phase\_map = {}  
  
for A in A\_values:  
 for sigma in sigma\_values:  
 for g in g\_values:  
 phase\_map[(A, sigma, g)] = classify(A, sigma, g)  
  
# Visualization of phase diagram for fixed sigma  
sigma\_fix = 0.5  
results = np.zeros((len(A\_values), len(g\_values)))  
  
for i, A in enumerate(A\_values):  
 for j, g in enumerate(g\_values):  
 results[i, j] = {  
 "stable": 0,  
 "collapse": 1,  
 "dispersion": -1  
 }[phase\_map[(A, sigma\_fix, g)]]  
  
plt.imshow(  
 results,  
 cmap="bwr",  
 extent=[min(g\_values), max(g\_values), min(A\_values), max(A\_values)],  
 origin="lower",  
 aspect="auto"  
)  
plt.colorbar(label="Phase (-1: dispersion, 0: stable, 1: collapse)")  
plt.xlabel("Coupling g")  
plt.ylabel("Amplitude A")  
plt.title(f"ψ Phase Diagram (σ={sigma\_fix})")  
plt.show()