Phase 9 – Part 5  
Numerical Exploration of ψ-Thermodynamic Ensembles

Goal  
In this part, I numerically explore ψ as a thermodynamic medium by constructing ensembles of ψ-configurations and tracking their statistical behaviors. The focus is to see whether ψ, when viewed through the lens of entropy and free-energy analogues, supports coherent statistical equilibria or fluctuating turbulent states.  
This part provides computational experiments, testing ψ-entropy and ψ-energy across ensembles.

Setup  
Core equation (upgraded ψ-gravity form):

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

Force field:

Plaintext:  
Force(x) = −∇[Gravity(x)]

For the ψ-thermodynamic analogy, I define:

ψ-energy functional:

Plaintext:  
E[ψ] = ∫ [ 0.5 (∇ψ)² + V(ψ) ] dx

with chosen as a quadratic + quartic potential:

Plaintext:  
V(ψ) = (a/2) ψ² + (b/4) ψ⁴

ψ-entropy functional (Shannon-like):

Plaintext:  
S[ψ] = −∫ P(ψ) log(P(ψ)) dx

where is the normalized distribution of ψ values in space.

ψ-free energy analogue:

Plaintext:  
F[ψ] = E[ψ] − T S[ψ]

Here, is an effective ψ-temperature parameter controlling fluctuation strength.

Ensemble Construction  
To study ψ-statistics, I generate an ensemble of ψ-configurations:

* Start from Gaussian random fields ψ(x).
* Evolve under a Langevin-type update:

Plaintext:  
∂ψ/∂t = − δF/δψ + η(x,t)

where is Gaussian white noise of strength ∝ T.

* Collect snapshots of ψ, compute ensemble averages of E, S, and F.

Numerical Experiment

# simulations/phase9\_part5\_ensemble.py  
import numpy as np  
  
# Parameters  
N = 256 # grid size  
dx = 1.0 / N  
dt = 0.01  
steps = 2000  
a, b = 1.0, 1.0 # potential parameters  
T = 0.1 # effective temperature  
ensembles = 20  
  
def laplacian(field, dx):  
 return (np.roll(field,1) + np.roll(field,-1) - 2\*field) / dx\*\*2  
  
def energy(psi, dx):  
 grad = (np.roll(psi,-1) - psi) / dx  
 V = 0.5\*a\*psi\*\*2 + 0.25\*b\*psi\*\*4  
 return np.sum(0.5\*grad\*\*2 + V)\*dx  
  
def entropy(psi, bins=50):  
 hist, edges = np.histogram(psi, bins=bins, density=True)  
 P = hist + 1e-12  
 return -np.sum(P\*np.log(P)) \* (edges[1]-edges[0])  
  
energies, entropies, free\_energies = [], [], []  
  
for e in range(ensembles):  
 psi = np.random.normal(0, 1, N)  
 for t in range(steps):  
 lap = laplacian(psi, dx)  
 dF = -lap + a\*psi + b\*psi\*\*3  
 noise = np.sqrt(2\*T\*dt/dx) \* np.random.normal(0,1,N)  
 psi += -dt\*dF + noise  
   
 E = energy(psi, dx)  
 S = entropy(psi)  
 F = E - T\*S  
 energies.append(E)  
 entropies.append(S)  
 free\_energies.append(F)  
  
print("Average Energy:", np.mean(energies))  
print("Average Entropy:", np.mean(entropies))  
print("Average Free Energy:", np.mean(free\_energies))