Phase 8 – Part 8  
Phase Diagram, Energy Diagnostics, and Invariants in the ψ–Metric System  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

🎯 Goal  
I map out the phase diagram of the ψ–metric feedback system described previously (Parts 5–7), identify approximate energy-like diagnostics and candidate invariants, and provide a numerical recipe to classify regimes (stable / oscillatory / chaotic).  
All analytic steps are presented as working hypotheses; the AI performed the numerical experiments and time-stepping used in examples.

🔧 Starting point (re-statement of the coupled system)  
From Part 6 / Part 7 we work with the 1D reduced, feedback-bearing evolution:

Plain text: ∂²\_t ψ(x,t) - c² ∂²\_x ψ(x,t) - m² ψ(x,t) = β γ ψ(x,t) ∂²\_x ψ(x,t)

Remarks:

* is wave speed, is the mass-like parameter, is the effective feedback strength.
* This PDE is nonlinear due to the term .

🔁 Nonlinear term as effective potential / modulational coupling  
I treat the right-hand-side nonlinear piece as a modulational coupling that can be rearranged (formally) by integrating by parts when constructing energy-like functionals. Formally, consider the Lagrangian density candidate (working ansatz):

Plain text: L = 1/2 (ψ̇² - c² (∂\_x ψ)² - m² ψ²) - (βγ/4) ∂\_x(ψ²) ∂\_x(ψ²)

Caveat:

* This Lagrangian is a working ansatz that captures a quartic gradient coupling producing a contribution like after variation; it is not unique. I flag this as provisional and consistent to leading order with the feedback structure.

From this ansatz, an energy density (Hamiltonian density) candidate reads:

Plain text: E = 1/2 (ψ̇² + c² (∂\_x ψ)² + m² ψ²) + (βγ/4) (∂\_x(ψ²))²

Interpretation:

* The first three terms are standard KG energy pieces (kinetic, gradient, mass).
* The last term penalizes sharp spatial modulation of and grows with feedback strength ; for large , energy can be stored in gradient-quartic channels, enabling localized excitation and possible turbulence-like cascades.

📐 Diagnostic quantities (computed in simulations)  
I use these diagnostics to classify long-term behaviour:

**Total energy**  — spatial integral of :

Plain text: E\_tot(t) = ∫\_{-L/2}^{L/2} E(ψ,ψ̇) dx

**Long-time RMS amplitude** :

Plain text: A\_rms = sqrt( (1/T) ∫\_{t0}^{t0+T} (1/L) ∫ ψ(x,t)² dx dt )

* Low : decay/stable.
* Moderate : persistent oscillations.
* Large, irregular : chaotic / energy cascade.

**Spectral entropy of spatial Fourier modes**  — measures energy spread over modes:

Plain text: S\_spec = -∑\_k p\_k log p\_k, p\_k = |ψ̂\_k|² / ∑\_j |ψ̂\_j|²

* Low entropy → energy concentrated in few modes (coherent).
* High entropy → broadband / turbulent.

**Finite-time Lyapunov proxy (practical)**: run two nearby initial conditions and ; compute log divergence of norm:

Plain text: λ\_FT(t) ≈ (1/t) ln( ||ψ¹(t)-ψ²(t)||\_2 / ||δψ||\_2 )

* Positive steady → chaos indicator.

🗺️ Phase diagrams: parameter axes and interpretation  
I focus on 2D slices through parameter space:

* at fixed .
* at fixed .

Classification metric (per grid point): compute long-time , , and . Use thresholds to label:

* **Stable:** low , low , .
* **Oscillatory:** moderate , low–moderate , but power concentrated in a few modes.
* **Chaotic:** large , high , .

This yields phase maps with sharp or fuzzy phase boundaries depending on nonlinearity.

🖥️ Python script — phase diagram scan + diagnostics

# simulations/phase8\_part8\_phase\_diagram.py  
import numpy as np  
import matplotlib.pyplot as plt  
from numpy.fft import rfft  
  
# --- Parameters for evolution (match previous parts for consistency) ---  
L = 20.0  
N = 512  
dx = L / N  
dt = 0.005  
steps = 2000  
c = 1.0  
m\_vals = np.linspace(0.05, 0.5, 6) # sample mass slice  
beta\_gamma\_vals = np.linspace(0.0, 6.0, 25) # effective feedback strength  
x = np.linspace(-L/2, L/2, N)  
  
# Initial condition: small Gaussian + small random noise  
def initial\_psi(x):  
 return np.exp(-x\*\*2) + 1e-3 \* np.random.randn(\*x.shape)  
  
# Diagnostics  
def compute\_energy(psi, psi\_dot, beta\_gamma, c, m, dx):  
 grad = np.gradient(psi, dx)  
 E\_density = 0.5 \* (psi\_dot\*\*2 + c\*\*2 \* grad\*\*2 + m\*\*2 \* psi\*\*2)  
 E\_density += 0.25 \* beta\_gamma \* (np.gradient(psi\*\*2, dx))\*\*2  
 return np.sum(E\_density) \* dx  
  
def spectral\_entropy(psi):  
 spec = np.abs(rfft(psi))\*\*2  
 p = spec / (spec.sum() + 1e-16)  
 S = -np.sum(p \* np.log(p + 1e-16))  
 return S  
  
# Time-stepping: simple leapfrog-style (matching earlier examples)  
def evolve\_single(beta\_gamma, m, steps=steps):  
 psi = initial\_psi(x)  
 psi\_old = psi.copy()  
 psi\_dot = np.zeros\_like(psi)  
 snapshots = []  
 for step in range(steps):  
 lap = (np.roll(psi, -1) - 2\*psi + np.roll(psi, 1)) / dx\*\*2  
 R\_approx = lap  
 psi\_new = (2\*psi - psi\_old + dt\*\*2 \* (c\*\*2 \* lap - m\*\*2 \* psi + beta\_gamma \* psi \* R\_approx))  
 psi\_old, psi = psi, psi\_new  
 if step >= steps - 200: # collect last 200 frames for long-time diagnostics  
 psi\_dot = (psi - psi\_old) / dt  
 snapshots.append((psi.copy(), psi\_dot.copy()))  
 return snapshots  
  
# Grid scan  
phase\_label = np.zeros((len(m\_vals), len(beta\_gamma\_vals))) # 0=stable,1=osc,2=chaos  
for i, m in enumerate(m\_vals):  
 for j, bg in enumerate(beta\_gamma\_vals):  
 snaps = evolve\_single(bg, m, steps=steps)  
 # Aggregate diagnostics over snapshots  
 if len(snaps) == 0:  
 phase\_label[i, j] = 0  
 continue  
 Ps = np.array([s[0] for s in snaps])  
 Pdot = np.array([s[1] for s in snaps])  
 A\_rms = np.sqrt(np.mean(Ps\*\*2))  
 S\_spec = np.mean([spectral\_entropy(p) for p in Ps])  
 # Finite-time Lyapunov proxy: perturb initial cond and compare divergence  
 # here we use a cheap proxy: temporal variance growth  
 temporal\_var = np.var(Ps)  
 # crude classifier thresholds (calibrated experimentally)  
 if (bg > 3.0 and S\_spec > 5.0) or temporal\_var > 1e-2:  
 label = 2 # chaotic  
 elif A\_rms > 0.3:  
 label = 1 # oscillatory  
 else:  
 label = 0 # stable  
 phase\_label[i, j] = label  
 print(f"Completed m={m:.3f}")  
  
# Plot phase diagram (m vs beta\_gamma)  
plt.figure(figsize=(8,5))  
plt.imshow(phase\_label, origin='lower', aspect='auto',  
 extent=[beta\_gamma\_vals[0], beta\_gamma\_vals[-1], m\_vals[0], m\_vals[-1]])  
cbar = plt.colorbar(ticks=[0,1,2])  
cbar.ax.set\_yticklabels(['Stable','Oscillatory','Chaotic'])  
plt.xlabel("βγ (feedback strength)")  
plt.ylabel("m (mass-like parameter)")  
plt.title("Phase diagram: ψ–metric feedback (coarse scan)")  
plt.show()