Phase 8 – Part 6  
ψ–Metric Feedback and Self-Consistent Geometry

🎯 Goal  
To extend ψ-gravity into a two-way system:

* ψ(x, t) shapes the metric.
* The resulting curvature (via metric derivatives) feeds back into ψ’s evolution.  
  This mirrors the recursive structure of general relativity (geometry ↔ energy), but rephrased:  
  ψ ↔ geometry, with ψ as the substrate and metric as the emergent expression.

🔧 Setup  
From the upgraded ψ-gravity core equation:

Plain text:  
Gravity(x,t) = ( ∇²[ space(x) + current(x)² ] ) \* ψ(x,t)

Force law:

Plain text:  
F(x,t) = - ∇[Gravity(x,t)]

Metric embedding (from Part 5):

Plain text:  
g\_mu,nu(x,t) = η\_mu,nu + α \* ψ(x,t) \* h\_mu,nu(x,t)

🔁 Feedback Law  
We now introduce ψ evolution with curvature feedback.  
Define ψ as satisfying a Klein–Gordon-like equation with Ricci scalar coupling:

Plain text:  
□ψ(x,t) - m² ψ(x,t) = β \* R(x,t) \* ψ(x,t)

Where:

* □ = covariant wave operator built from .
* = Ricci scalar of the ψ-perturbed metric.
* = mass-like term for ψ.
* β = coupling constant controlling ψ–curvature feedback.

🌊 Desert Analogy Extension

* ψ = desert floor (still the foundation).
* Metric = dunes shaped by ψ.
* Curvature = dune sharpness, feeding back into the floor’s erosion/deposition.
* Feedback loop = ψ adjusts to the geometry it itself induced, like a desert that continuously reshapes under its own wind patterns.

📐 Coupled System

Metric from ψ:

Plain text:  
g\_mu,nu(x,t) = η\_mu,nu + α \* ψ(x,t) \* h\_mu,nu(x,t)

Curvature from metric:

Plain text:  
R(x,t) = g^μν R\_μν(x,t)

ψ evolution:

Plain text:  
□ψ - m²ψ = β R ψ

🔬 Sample Case: 1D Simplification

To illustrate, restrict to 1D ψ(x, t) with small metric perturbations.  
Approximate Ricci scalar as proportional to second derivative of ψ:

Plain text:  
R(x,t) ≈ γ \* ∂²\_x ψ(x,t)

Then the ψ evolution reduces to:

Plain text:  
∂²\_t ψ - c² ∂²\_x ψ - m² ψ = βγ (∂²\_x ψ) ψ

🖥️ Python Simulation

# simulations/phase8\_part6\_feedback.py  
import numpy as np  
import matplotlib.pyplot as plt  
  
# Parameters  
L = 10.0  
N = 200  
dx = L / N  
dt = 0.01  
steps = 500  
  
c = 1.0  
m = 0.1  
beta = 0.5  
gamma = 1.0  
  
# Grid  
x = np.linspace(-L/2, L/2, N)  
  
# Initial Gaussian psi  
psi = np.exp(-x\*\*2)  
psi\_old = psi.copy() # copy for time stepping (psi at t - dt)  
  
# Storage for evolution snapshots  
history = []  
  
for step in range(steps):  
 # Second derivative in space (finite difference, periodic BC via roll)  
 laplacian = (np.roll(psi, -1) - 2 \* psi + np.roll(psi, 1)) / dx\*\*2  
  
 # Effective Ricci scalar (approx)  
 R = gamma \* laplacian  
  
 # Discretized wave operator update (second-order in time)  
 psi\_new = (2 \* psi - psi\_old +  
 dt\*\*2 \* (c\*\*2 \* laplacian - m\*\*2 \* psi + beta \* R \* psi))  
  
 psi\_old, psi = psi, psi\_new  
  
 if step % 50 == 0:  
 history.append(psi.copy())  
  
# Plot snapshots  
plt.figure(figsize=(8, 6))  
for i, snapshot in enumerate(history):  
 t\_val = i \* 50 \* dt  
 plt.plot(x, snapshot, label=f"t={t\_val:.2f}")  
plt.legend()  
plt.title("ψ Evolution with Metric Feedback (1D)")  
plt.xlabel("x")  
plt.ylabel("ψ")  
plt.show()