Phase 13A – Mathematical Consolidation  
Part 1: Canonical Tensorial Notation of ψ-Gravity

Goal  
I now compactify ψ-gravity into a canonical tensorial formalism. The aim is to elevate the descriptive equation into a structure suitable for action principles, symmetry analysis, and eventual unification.

Core Equation Restated

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

Force is defined as:

Plain-text:  
F(x) = −∇[Gravity(x)]

Canonical Tensorial Setup

Let spacetime be described by coordinates with metric . I construct the following objects:

**ψ field**:  
A scalar (or scalar-density) field representing the substrate.

Plain-text:  
ψ: ℝ⁴ → ℝ

**Spatial contribution**:  
Encoded as , a scalar function.

**Current contribution**:  
Let be a vector field representing the dynamical current (wind in the analogy). Then:

Plain-text:  
current(x)² = g^{μν} J\_μ(x) J\_ν(x)

Compact Gravity Operator

Define the effective scalar potential:

Plain-text:  
Φ(x) = S(x) + g^{μν} J\_μ(x) J\_ν(x)

Then the compactified ψ-gravity equation becomes:

Plain-text:  
G(x) = (∇²Φ(x)) × ψ(x)

with the force law:

Plain-text:  
F\_μ(x) = −∇\_μ G(x)

Tensorial Expression

The Laplacian is expressed covariantly:

Plain-text:  
∇²Φ = g^{μν} ∇\_μ ∇\_ν Φ

Thus:

Plain-text:  
G(x) = (g^{μν} ∇\_μ ∇\_ν Φ(x)) × ψ(x)

Canonical Tensorial Summary

* Substrate scalar: ψ(x)
* Effective potential: Φ(x) = space(x) + g^{μν} J\_μ J\_ν
* Gravity density:

Plain-text:  
G(x) = (∇²Φ(x)) × ψ(x)

* Force density:

Plain-text:  
F\_μ(x) = −∇\_μ G(x)

This canonical tensorial notation sets the foundation for symmetry analysis and Noether derivations in the next part.

Python Symbolic Prototype

# simulations/phase13A\_part1\_tensorial\_notation.py  
import sympy as sp  
  
# Coordinates and metric  
t, x, y, z = sp.symbols('t x y z')  
coords = (t, x, y, z)  
g = sp.diag(-1, 1, 1, 1) # Minkowski for prototype  
  
# Fields  
psi = sp.Function('psi')(\*coords)  
S = sp.Function('S')(\*coords)  
J = [sp.Function(f'J{mu}')(\*coords) for mu in range(4)]  
  
# Current squared  
current\_sq = sum([g[mu, nu] \* J[mu] \* J[nu] for mu in range(4) for nu in range(4)])  
  
# Effective potential  
Phi = S + current\_sq  
  
# Laplacian in flat space  
laplacian = sum([g[mu, mu] \* sp.diff(Phi, coords[mu], 2) for mu in range(4)])  
  
# Gravity density  
G = laplacian \* psi  
  
# Force (gradient of Gravity)  
F = [sp.diff(G, c) for c in coords]  
  
print("Effective Potential Φ:", Phi)  
print("Gravity Density G:", G)  
print("Force Components Fμ:", F)