Phase 15 – ψ-Gravity as Unified Symbolism  
Part 2: ψ → Symbolic / Computational Substrate Mapping  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Goal**  
I formalize how ψ functions simultaneously as a physical field and as a computational/informational substrate. I provide precise mappings from the ψ-field to information measures, symbolic encodings, and simple dynamical computation primitives. All technical derivations and the example numerical code below are executed/produced by the AI as my computational collaborator; I remain the author of the conceptual mappings and interpretations.

### 1. Core definitions and preamble

I reuse the immutable core equation of the project as the physical anchor:

Plain text:  
Gravity(x,t) = (nabla^2 [ space(x,t) + current(x,t)^2 ]) \* psi(x,t)

This is the physical field that couples ψ to space/current. From ψ I define probability/information densities, modal encodings, and computational capacities.

### 2. Normalized probability density (ψ → probability)

To interpret ψ as an information-bearing probability amplitude, I define a normalized density:

Plain text:  
p(x,t) = |psi(x,t)|^2 / ( \_Omega |psi(x’,t)|^2 dx’ )

### 3. Local information density and total entropy

Local pointwise information density (Shannon differential form):

Plain text:  
s(x,t) = - p(x,t) log( p(x,t) )

Global information (differential entropy on domain Ω):

Plain text:  
S(t) = - \_Omega p(x,t) log( p(x,t) ) dx

### 4. Computational capacity density (heuristic scalar)

Plain text:  
C(x,t) = lambda1 \* |psi(x,t)|^2 + lambda2 \* |nabla psi(x,t)|^2 + lambda3 \* |Gravity(x,t)|

* λ\_i are tunable, positive coefficients setting the units and emphasis.
* Intuition: amplitude carries stored signal; gradients support differentiation and edge-like structure; gravity-like pressure provides energetic substrate for robust, energetically stable computation.

### 5. Modal decomposition and symbolic basis

Modal amplitude:

Plain text:  
a\_i(t) = \_Omega psi(x,t) \* phi\_i(x) dx

Symbolic activation (binary or analog):

Plain text:  
s\_i(t) = H( a\_i(t) - theta\_i )

where H is the Heaviside step (or a smooth sigmoid for analog states) and θ\_i is a threshold per channel. The set {s\_i} forms a discrete/quantized symbolic state extracted from the continuous ψ field.

### 6. Associative / energy-based symbolic dynamics

Plain text:  
H\_info({s\_i}) = -1/2 sum\_{i,j} w\_ij s\_i s\_j + sum\_i theta\_i s\_i

Weights w\_ij can be derived from stored modal patterns (Hebbian rule), and these weights can themselves be plastic functions of ψ (via local overlap integrals).

### 7. Mutual information and regional coupling

Plain text:  
I(A;B) = \_{A x B} p(x\_A,x\_B) log( p(x\_A,x\_B) / (p\_A(x\_A) p\_B(x\_B)) ) dx\_A dx\_B

### 8. Dynamics: mapping field evolution → symbolic computation

Physical update:

Plain text:  
d2/dt2 psi = c^2 nabla^2 psi - V’(psi) + F\_ext(x,t)

Symbol extraction: project ψ onto {ϕ\_i}, compute a\_i, apply thresholds → update {s\_i}.

### 9. Learning / plasticity (ψ as modifiable substrate)

Plain text:  
d/dt kappa(x,t) = eta \* ( Hebb(x,t) - gamma \* kappa(x,t) )

### 10. Example algorithm (step-by-step protocol)

* Choose domain Ω and basis {ϕ\_i}.
* On a grid, evaluate ψ(x). Compute p(x), local s(x), and global S.
* Project to modal amplitudes a\_i = ∫ψϕ\_i.
* Threshold to symbolic bits s\_i = H(a\_i - θ\_i).
* Compute associative/energy weights w\_ij.
* Optionally apply read/write forcing F\_sym.
* Repeat for time evolution and evaluate information flows.

### 11. Python example (1D minimal reproducible snippet)

# simulations/phase15\_part2\_symbolic\_mapping.py  
import numpy as np  
  
def laplacian(f, dx):  
 return np.gradient(np.gradient(f, dx), dx)  
  
# Grid  
L = 10.0  
N = 2048  
x = np.linspace(-L, L, N)  
dx = x[1] - x[0]  
  
# Example psi: sum of two Gaussian wells  
sigma1, sigma2 = 0.6, 1.2  
psi = 1.2 \* np.exp(- (x+2.0)\*\*2 / (2\*sigma1\*\*2)) + 0.9 \* np.exp(- (x-1.1)\*\*2 / (2\*sigma2\*\*2))  
  
# Example space and current fields  
space = 0.1 \* np.sin(0.4 \* x)  
current = 0.5 \* np.exp(-x\*\*2 / (2\*1.5\*\*2)) \* np.sin(0.8 \* x)  
  
# Compute Gravity  
A = laplacian(space + current\*\*2, dx)  
Gravity = A \* psi  
  
# Normalize to get p(x)  
psi2 = np.abs(psi)\*\*2  
norm = np.trapz(psi2, x)  
p = psi2 / (norm + 1e-30)  
  
# Local info density and total entropy  
eps = 1e-12  
s\_local = - p \* np.log(p + eps)  
S\_total = np.trapz(s\_local, x)  
  
# Computational capacity density C(x)  
lambda1, lambda2, lambda3 = 1.0, 0.5, 0.2  
grad\_psi = np.gradient(psi, dx)  
C = lambda1 \* np.abs(psi)\*\*2 + lambda2 \* grad\_psi\*\*2 + lambda3 \* np.abs(Gravity)  
  
# Modal basis (localized Gaussians)  
num\_modes = 16  
centers = np.linspace(-6, 6, num\_modes)  
width\_mode = 0.9  
phi = np.array([np.exp(- (x - c)\*\*2 / (2\*width\_mode\*\*2)) for c in centers])  
  
# Orthonormalize (Gram-Schmidt)  
for i in range(num\_modes):  
 for j in range(i):  
 proj = np.trapz(phi[i]\*phi[j], x)  
 phi[i] -= proj \* phi[j]  
 norm\_phi = np.sqrt(np.trapz(phi[i]\*\*2, x))  
 if norm\_phi > 0:  
 phi[i] /= norm\_phi  
  
# Modal amplitudes and symbolic bits  
a = np.array([np.trapz(psi \* phi\_i, x) for phi\_i in phi])  
theta = 0.1 \* np.max(np.abs(a))  
s\_bits = (a > theta).astype(int)  
  
# Print diagnostics  
print("Total entropy S =", S\_total)  
print("Top 5 modal amplitudes:", np.sort(np.abs(a))[-5:])  
print("Symbolic bits (first 16):", s\_bits)

### 12. Validation criteria and expected signatures

* Localized high-C(x) regions correlate with modal concentration.
* Low entropy pockets near localized ψ wells, high entropy at boundaries.
* Mutual information peaks across regions with modal overlap.
* Associative recall observed under H\_info-like dynamics.
* Gravity(x) aligns with energetically stable symbolic loci.

### 13. Deliverables (Part 2)

* Formal mappings from ψ → p(x), S, C(x), a\_i, s\_i.
* Python script for diagnostics.
* Protocol for read/write cycles and plasticity.
* Validation criteria for ψ as a computational substrate.

### 14. Continuity and next steps

Part 3 will:  
(a) run targeted numerical experiments,  
(b) test associative storage/retrieval,  
(c) measure scaling of computational capacity.

### Quick reference: essential equations

**Probability density**

p(x,t) = |ψ(x,t)|² / ∫\_Ω |ψ(x′,t)|² dx′

Plain text:  
p(x,t) = |psi(x,t)|^2 / ( \_Omega |psi(x’,t)|^2 dx’ )

**Local entropy density**

s(x,t) = - p(x,t) log p(x,t)

Plain text:  
s(x,t) = - p(x,t) log( p(x,t) )

**Total entropy**

S(t) = - ∫\_Ω p(x,t) log p(x,t) dx

Plain text:  
S(t) = - \_Omega p(x,t) log( p(x,t) ) dx

**Capacity density**

C(x,t) = λ₁ |ψ|² + λ₂ |∇ψ|² + λ₃ |Gravity(x,t)|

Plain text:  
C(x,t) = lambda1 \* |psi(x,t)|^2 + lambda2 \* |nabla psi(x,t)|^2 + lambda3 \* |Gravity(x,t)|

**Modal amplitudes**

a\_i(t) = ∫\_Ω ψ(x,t) ϕ\_i(x) dx

Plain text:  
a\_i(t) = \_Omega psi(x,t) \* phi\_i(x) dx

**Binary symbolic readout**

s\_i(t) = H(a\_i(t) - θ\_i)

Plain text:  
s\_i(t) = H( a\_i(t) - theta\_i )

**Symbolic Hamiltonian**

H\_info({s\_i}) = -½ ∑\_(i,j) w\_ij s\_i s\_j + ∑\_i θ\_i s\_i

Plain text:  
H\_info({s\_i}) = -1/2 sum\_{i,j} w\_ij s\_i s\_j + sum\_i theta\_i s\_i

End of Phase 15 — Part 2 (ψ → Symbolic / Computational Substrate Mapping).