Paper ID: FRC 100.008 | Version: v2.2

Computational Realization of Lambda-Field Coherence Dynamics v2.2

Author: Hadi Servat

Affiliation: Independent Researcher

Contact: publish@fractalresonance.com

Website: fractalresonance.com

Abstract

FRC 100.008 v2.2 corrects the executable Lambda engine and separates Lambda_obs, Lambda_eq, and optional Lambda_dyn. It also separates the canonical starred Boltzmann bridge k from the pilot's state-derived kernel-score normalization s_ref. The latter is selected once from a declared pre-evolution reference and frozen only to normalize the computational coherence readout; it is not k* and carries no reciprocity-law status. Lambda_obs is an instantaneous transform; Lambda_dyn is a latent model state relaxing toward Lambda_eq; a fundamental field remains a separate conjecture. The*

> *finite-difference and oscillator pilots demonstrate executable closures only and do not establish physical ontology.*

1. Role in the 100-series

FRC 100.007 introduces the Lambda-field as a scalar completion of coherence dynamics. FRC 100.009 then writes local open-system reciprocity as a transport law. This paper is the computational bridge: `100.007 Lambda field -> 100.008 executable Lambda dynamics -> 100.009 local reciprocity`.

The role of 100.008 is not to introduce another ontology. Its job is to type observed, target, and latent objects explicitly and test whether a latent closure adds anything beyond the observed transform.

An algebraic Λ_{obs} is legitimate as an observed transform. The error is assigning it independent dynamics while conflating it with a target or latent state. Version 2.2 also corrects a notation error: the pilot's frozen kernel-score normalization is not the reciprocity bridge k^*.

2. State, target, and field are separate objects

Let the computational state be a normalized density or probability mass:

$$\rho(x,t) \geq 0, \qquad \int_{\Omega} \rho(x,t)\, dx = 1.$$

The engine uses four separate objects:

1. the state field `rho`,
2. the operational coherence readout `C_obs[rho]`,
3. the equilibrium target field `Lambda_eq[rho]`,
4. the optional latent dynamical field `Lambda_dyn`.

The observed transform is

$$\Lambda_{\mathrm{obs}}(x,t) = \Lambda_0 \ln C_{\mathrm{obs}}(x,t).$$

The target supplied by the declared closure is $\Lambda_{\mathrm{eq}} = \Lambda_0 \ln C_{\mathrm{target}}$. Only Λ_{dyn} carries independent lag and diffusion.

Before evolution, declare $\mu = (\text{information-nat}, N, \ell, K_\ell, \epsilon, \rho_{\text{ref}})$ with $\rho_{\text{ref}} = \rho(t = 0)$ and select once the computational kernel-score scale

$$s_{\text{ref},\mu} = \max(\text{mean}_i\, \bar{s}_K[\rho_{\text{ref}}]_i, \epsilon). \tag{1}$$

The value is frozen for the trajectory and is never recomputed from the evolving state. It normalizes this pilot's kernel score; it is not the starred Boltzmann bridge and is not used to test reciprocity.

3. Kernel coherence target

The local target is mesoscopic, not pointwise. Define a smoothed score

$$\bar{s}_K(x,t) = \int_\Omega K_\ell(x-y)\, s_\gamma[\rho](y,t)\, dy,$$

with

$$s_\gamma[\rho] = -\rho \ln(\rho + \epsilon) + \frac{\gamma}{2}|\nabla\rho|^2.$$

The kernel coherence target is

$$C_{\text{obs}}[\rho](x,t) = \exp[-\bar{s}_K(x,t)/s_{\text{ref},\mu}],$$

and the corresponding observed Lambda field is

$$\Lambda_{\text{obs}}[\rho](x,t) = \Lambda_0 \ln C_{\text{obs}}[\rho](x,t).$$

The reference pilot uses the identity target closure $C_{\text{target}} = C_{\text{obs}}$, so $\Lambda_{\text{eq}} = \Lambda_{\text{obs}}$. Other targets require a predeclared calibration rule.

4. Dynamic Lambda equation

The optional latent field obeys

$$\partial_t \Lambda_{\text{dyn}} = D_\Lambda \nabla^2 \Lambda_{\text{dyn}} - \mu_\Lambda(\Lambda_{\text{dyn}} - \Lambda_{\text{eq}}[\rho]) + J_{\text{ext}}.$$

Here `D_Lambda` is a propagation or smoothing coefficient, `mu_Lambda` is the relaxation rate toward the target, and `J_ext` is an optional external or boundary source. This produces the computational loop

$$\rho \longrightarrow C_{\mathrm{obs}}[\rho] \longrightarrow \Lambda_{\mathrm{obs}} \longrightarrow \Lambda_{\mathrm{eq}} \longrightarrow \Lambda_{\mathrm{dyn}} \longrightarrow \rho.$$

The state evolves under diffusion and Lambda-guided drift:

$$\partial_t \rho = D_\rho \nabla^2 \rho - \alpha \nabla \cdot (\rho \nabla \Lambda_{\mathrm{dyn}}) + \Xi_\rho.$$

The two equations are reciprocally coupled but not algebraically identical.

5. Discrete finite-difference engine

On a periodic one-dimensional grid, the reference implementation uses probability masses `rho_i` with `sum_i rho_i = 1`. At each step:

1. Compute the local score `s_i = -rho_i log(rho_i + eps)`.
2. Smooth it by FFT convolution with a Gaussian kernel `K_ell`.
3. Before the loop, select $s_{\mathrm{ref},\mu}$ once from the declared reference state using (1).
4. Compute `C_obs,i = exp(-sbar_i/score_scale)` without changing `score_scale`.
5. Compute `Lambda_obs` and the declared `Lambda_eq` target.
6. Update `Lambda_dyn` by the relaxation-diffusion equation.
7. Update `rho` by diffusion plus `-alpha div(rho grad Lambda_dyn)`.
8. Clip `rho_i >= 0` and renormalize.

The periodic spectral Laplacian is

$$\widehat{\nabla^2 \Lambda_{\mathrm{dyn}}}_q = -|q|^2 \widehat{\Lambda}_{\mathrm{dyn,q}},$$

so the spectral Lambda update is

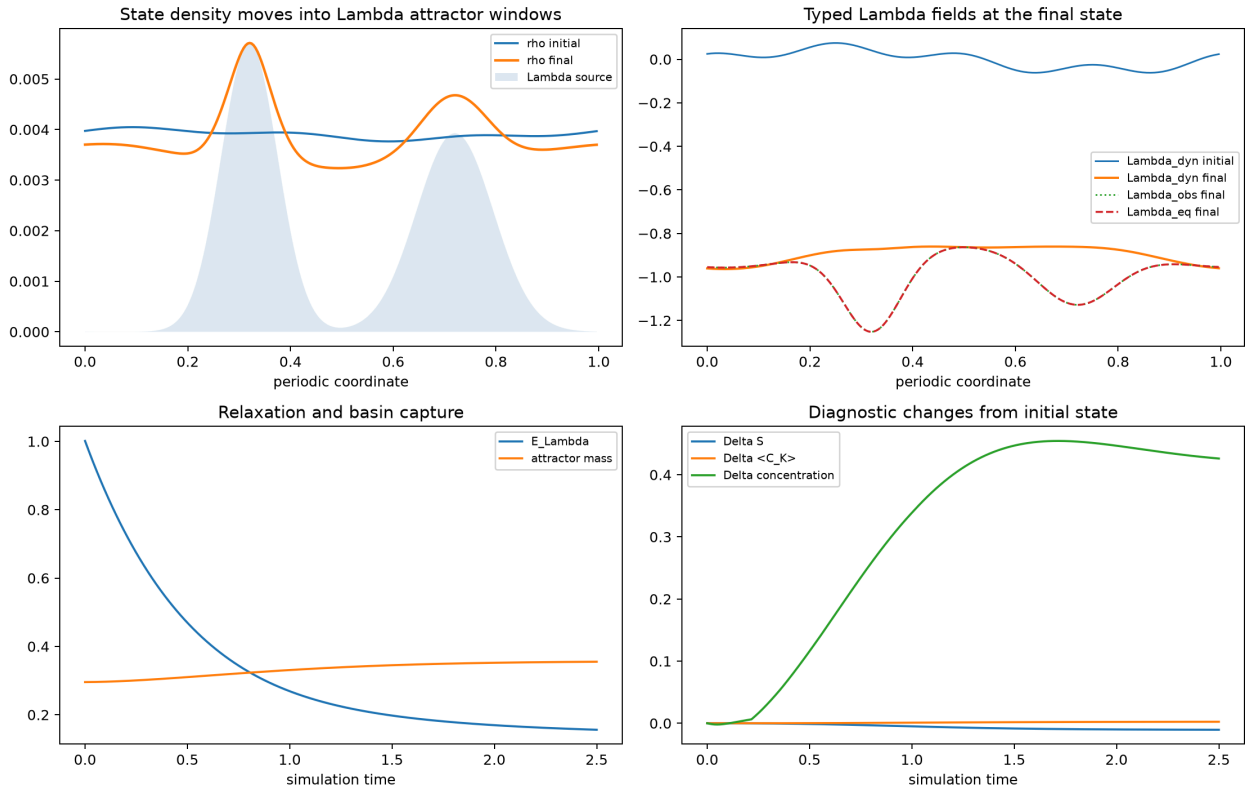
$$\partial_t \widehat{\Lambda}_{\mathrm{dyn,q}} = -D_\Lambda |q|^2 \widehat{\Lambda}_{\mathrm{dyn,q}} - \mu_\Lambda (\widehat{\Lambda}_{\mathrm{dyn,q}} - \widehat{\Lambda}_{\mathrm{eq},q}) + \widehat{J}_{\mathrm{ext},q}.$$

6. Reference implementation A: Lambda engine

The finite-difference pilot initializes a near-uniform state density on a periodic line and uses two Lambda source windows as attractor regions. The purpose is not to prove a physical law; it is to test the computational loop.

Measured canonical metrics:

Diagnostic	Initial	Final	Change
Entropy `S`	5.544985	5.534524	-0.010461
Kernel coherence `<C_obs>`	0.367922	0.370331	+0.002409
`Lambda_dyn`-target mismatch	1.001261	0.156089	-84.4%
Attractor-window mass	0.295593	0.355171	+0.059579
Concentration `max(rho)/mean(rho)`	1.036124	1.462226	+0.426102



Finite-difference Lambda-engine diagnostic. The dynamic Lambda field relaxes toward the kernel target while state density moves into the source-defined attractor windows. The pilot records Lambda-target mismatch, attractor mass, entropy, kernel coherence, and concentration from the same fixed-seed run.

The selected kernel-score scale is $s_{\mathrm{ref},\mu} = 0.0216600983$ for the declared reference state. The strongest software result is explicit type separation: the observed transform and target are computed from `rho`, the latent field relaxes with lag, and the state responds to the latent gradient. This is a closure demonstration, not physical evidence.

7. Oscillator implementation

Lambda can also be realized by oscillator dynamics. Let each computational unit have phase `phi_i`. The base coupling is

$$\dot{\phi}_i = \omega_i + \sum_j J_{ij} \sin(\phi_j - \phi_i).$$

A spatial vector `grad Lambda` cannot be added directly to a scalar phase rate. Three scalar channels are admissible:

Channel A - scalar frequency bias:

$$\dot{\phi}_i = \omega_i + \sum_j J_{ij} \sin(\phi_j - \phi_i) + \eta_\Lambda \Lambda(x_i, t).$$

Channel B - phase-reference coupling:

$$L(x,t) = C_\Lambda(x,t) e^{i\theta_\Lambda(x,t)},$$

$$\dot{\phi}_i = \omega_i + \sum_j J_{ij} \sin(\phi_j - \phi_i) + \eta C_\Lambda(x_i,t) \sin(\theta_\Lambda(x_i,t) - \phi_i).$$

Channel C - projected gradient drive, only when a physical orientation vector `n_i` exists:

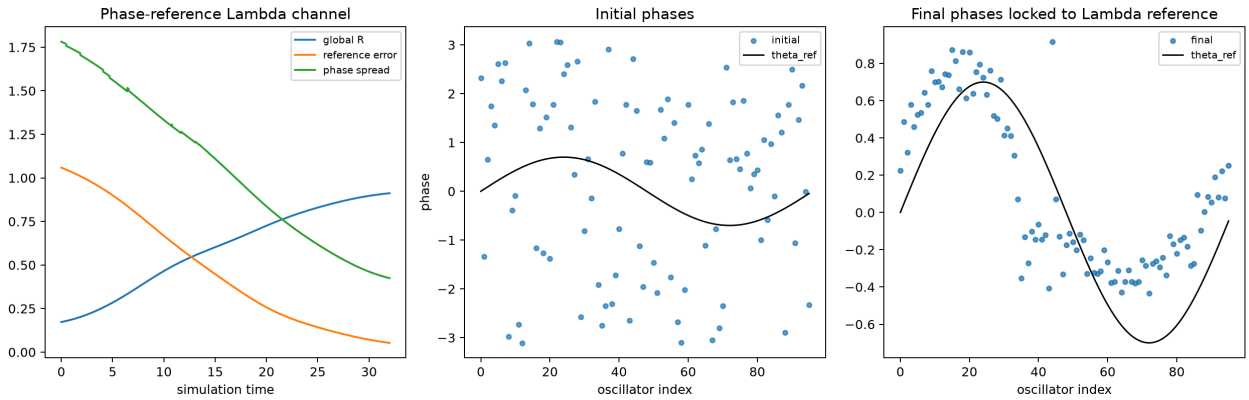
$$\dot{\phi}_i = \omega_i + \sum_j J_{ij} \sin(\phi_j - \phi_i) + \eta\, n_i \cdot \nabla\Lambda(x_i,t).$$

The phase-reference channel is the preferred computational realization because it makes Lambda a coherence clock rather than a dimensionally invalid vector injection.

8. Reference implementation B: phase-reference locking

The oscillator pilot uses a ring network with local Kuramoto coupling and a fixed Lambda phase-reference field. The result is a direct check of the corrected scalar channel.

Diagnostic	Initial	Final	Change
Global order `R`	0.172888	0.912092	5.28x
Reference error	1.059117	0.053094	-95.0%
Phase spread	1.781577	0.424640	-1.356936



Phase-reference oscillator diagnostic. Random initial phases lock to the Lambda reference channel under scalar phase-reference coupling. The run tracks global order, reference error, and phase spread.

This test is intentionally modest. It proves that the paper's preferred oscillator channel is executable and dimensionally consistent. It does not prove that physical measurement is oscillator locking.

9. Relation to 566.030 and 100.009

The canonical anchor is FRC 566.001 v2.2: $dS + k^* d\ln C = 0$, with indexed notation reserved for declared operational realizations. The engine does not establish a directional law or identify a thermodynamic residual.

100.008 therefore should be read as an engine, not as a thermodynamic proof. It supplies fields and fluxes that 100.009 can place into a local open-system law:

$$\partial_t s_\mu + k^*_\mu \partial_t \ln C_\mu + \nabla \cdot J_R = \Sigma_R.$$

The engine provides candidate transport objects: `rho`, `Lambda_obs`, `Lambda_eq`, `Lambda_dyn`, field mismatch, drift flux, convergence time, and oscillator order. An explicit environment model is still required for thermodynamic interpretation.

10. Kill conditions

This paper fails in its present computational role if:

1. the code conflates `Lambda_obs`, `Lambda_eq`, and `Lambda_dyn`;
2. `score_scale` is recomputed from the evolving state rather than frozen from the declared reference;
3. the finite-difference engine fails positivity or normalization;
4. `E_Lambda` does not decrease in the relaxation pilot;
5. attractor-window mass does not increase in the drift pilot;
6. oscillator coupling adds a spatial vector directly to a scalar phase rate;
7. the phase-reference oscillator pilot does not increase order or reduce reference error.

The shipped pilots pass these checks, but they are not enough to claim physical ontology.

11. Appendix program

The next paper or appendix should add:

1. a two-dimensional Lambda engine with attractor basin maps;
2. parameter sweeps for `ell`, `mu_Lambda`, `D_Lambda`, `alpha`, and the predeclared kernel-score normalization;
3. an explicit open export channel compatible with 100.005 v2;
4. coupling to the 100.003 stage-1/stage-2 collapse structure;
5. a comparison between oscillator locking time and the 787 coherence-lifetime criterion.

12. Source package

`FRC-100-008-v2.2.md`; `FRC-100-008-v2.2.pdf`; `fig1_lambda_engine.png`; `fig2_oscillator_locking.png`; `pilot/pilot_lambda_engine.py`; `pilot/lambda_metrics.json`; `pilot/compare_metrics.py`; `requirements.txt`; `README_RELEASE.md`; `render_frc_html_pdf.py`; `CHECKSUMS.txt`. The quoted numbers are generated by `pilot/pilot_lambda_engine.py` from fixed seeds and checked by `pilot/compare_metrics.py`.

13. Version note

Version 2.2 renames the state-derived normalization from k^* to $s_{\mathrm{ref},\mu}$ and restores the canonical/operational reciprocity hierarchy. The numerical pilot is unchanged. Version 2.1's separation of observed, target, latent, and fundamental Lambda meanings is retained.

References

1. H. Servat, *FRC 100.001 - Fractal Resonance Cognition*.
2. H. Servat, *FRC 100.002 v2.5 - Coherence in Chaos*.
3. H. Servat, *FRC 100.003 v3.3 - Collapse as Open-System Phase-Locking*.
4. H. Servat, *FRC 100.005 v2.3 - Thermodynamics of Locking*.
5. H. Servat, *FRC 100.007 - Lambda-Field: Scalar Completion of Coherence Dynamics*.
6. H. Servat, *FRC 100.009 - Local Reciprocity and Activation Dynamics*.
7. H. Servat, *FRC 566.030 - Reciprocity in Action*.
8. Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulence*, Springer.
9. J. A. Acebron et al., "The Kuramoto model: A simple paradigm for synchronization phenomena," *Reviews of Modern Physics* 77, 137-185 (2005).
10. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *PNAS* 79, 2554-2558 (1982).
11. H. Haken, *Synergetics*, Springer.