**MATTER V2 Core Engine (Present‑Act / +1 Picture) — Specification for Defensive Publication**

**1) Purpose and Scope**

The MATTER V2 “core engine” is a discrete, deterministic simulation substrate designed to generate field‑like and structure‑like behavior **without introducing continuous fields, analog forces, or probabilistic control**. The engine’s job is to evolve a lattice of sites over discrete ticks using a **present‑act selection rule**: sites either “act” (+1) or do not act (0) at each tick, based only on **integer predicates**.

This engine underlies the V2 simulation suite (e.g., baseline kinematics, isotropy audits, invariance checks, relational maps, EM‑like proxies, gravity‑like feasibility, and matter‑addition tests). Individual simulations differ mainly in their **geometry**, **control schedule**, and **diagnostic readouts**, not in the engine ontology.

**2) Ontology: Sites, Ticks, and Acts**

* **Lattice:** A finite 2D grid (typically ) of sites indexed by integer coordinates .
* **Time:** Discrete ticks .
* **Act state:** Each tick produces a set of “active” sites. A site’s activity is binary: **0 or +1** (no analog magnitude in control).

Terminology used in reports:

* **“Present‑act”** means eligibility depends only on current tick predicates and declared memory flags (e.g., cooldown), not on continuous accumulated fields.
* The **“+1 picture”** refers to measured patterns of activity counts, overlaps, shells, sectors, loop edges, etc., which may resemble classical field behavior but are **readouts from discrete acts**, not imposed potentials.

**3) The Non‑Negotiable Contract (Across All V2 Sims)**

**3.1 Control is boolean/ordinal and integer-defined**

At each tick, eligibility is computed from **integer predicates** such as:

* membership in a **shell** (integer radius band),
* membership in a **sector** (integer angular partition),
* membership in a **loop/ring**,
* duty‑window phase (period/duty integers),
* **rails** parity (e.g., when explicitly enabled),
* neighborhood support thresholds (integer neighbor counts),
* cooldown counters (integer tick offsets),
* explicitly declared bootstrap moments (e.g., first tick of a duty window).

**Control never uses:**

* real‑valued weights, analog forces, “potential” values,
* probability thresholds inside eligibility,
* hidden randomization,
* floating comparisons that influence selection.

**3.2 Diagnostics never drive control**

Metrics such as:

* slopes, plateaus, coherence, circulation, re‑expression,
* correlation peaks, phase lags, segmentation fits,
* isotropy p‑values, CVs, RMSEs,

are computed **after** activity occurs. They are **readouts** and may not alter eligibility, periods, steps, thresholds, or geometry.

**3.3 Determinism and reproducibility**

* Control is deterministic by default.
* If randomness exists, it is allowed **only** for **tie-breaking** in explicitly declared tie events (PF/Born style), using a fixed seed, and must be auditable such that:
  + RNG is invoked **only** at ties,
  + control output is identical for the same seed and manifest.

Each run records:

* the full manifest used (or a hash),
* a diagnostics config (or a hash),
* engine entrypoint version identity,
* environment snapshot sufficient for replay.

**3.4 Mesh discipline for “field-like” claims**

If a claim could be contaminated by discretization artifacts (e.g., radial laws, plateau flatness), it must be checked at **two resolutions** (two meshes) with strict bounds on differences (e.g., slope deltas, CV deltas). Geometry/readout must be aligned to ensure comparisons are meaningful.

**4) Geometry Primitives Used by the Engine**

The engine reuses a small set of geometry partitions to define eligibility and to aggregate diagnostics:

**4.1 Shells (integer radial bands)**

Given a center , define integer radius:

A **shell** is the set of sites with radius . Shells are used for radial laws and “field proxy” tests.

**4.2 Sectors (integer angular bins)**

Given radius , define an angular index by mapping into integer bins. Sectors are used for anisotropy audits, phase/correlation tests, and loop directionality.

**4.3 Rings/Loops**

A ring is typically a narrow radius band optionally subdivided into sectors. Loops are used for circulation and directed-flow diagnostics.

**4.4 Boundary masks**

Certain sims restrict eligible cells by deterministic masks (e.g., checker keep/period patterns). These masks must be isotropic “in expectation” with testing windows chosen to avoid lattice symmetry artifacts.

**5) Control Building Blocks (Reusable Across Sims)**

Most sims implement control by combining these building blocks:

**5.1 Duty windows (period/duty)**

A site/group is eligible on tick if:

with integer period and duty . Used to define activity ladders across bands.

**5.2 Cooldown**

If a site acted at , it may be disallowed for the next ticks. This suppresses immediate re‑firing unless explicitly enabled in a sim (e.g., to measure re‑expression variance).

**5.3 Rails (optional parity organizer)**

When enabled, rails partition the lattice by parity:

and alternate eligibility by tick parity. Rails are used only where explicitly required to sustain deterministic neighbor support and avoid dead starts; otherwise they are disabled to prevent aliasing.

**5.4 Neighbor support (K-gating)**

A predicate that requires at least supporting neighbor acts in a neighborhood at the appropriate time window. For cold starts under narrow duty windows, a declared bootstrap rule may bypass support check on the first tick of a window (as a control rule, not a diagnostic hack).

**5.5 DDA schedulers (deterministic frequency laws)**

A key engine pattern is a DDA‑style integer accumulator to enforce exact firing frequencies without analog weights. For example, for shell index :

* accumulator starts at 0,
* each tick: (integer),
* if : shell fires; .

This yields an exact firing rate proportional to (hence per cell when shell populations are accounted for), while remaining purely integer and deterministic.

**6) Diagnostics Standardization (Readout Discipline)**

The engine’s measurements follow strict rules to avoid mistaking instrumentation artifacts for physics:

**6.1 Windowed metrics under duty/rails**

Whenever eligibility depends on duty windows or rails, “last-tick coherence” can be meaningless. Coherence and overlap are computed over the **union of the last duty window** (“C2\_window”), not a single tick.

**6.2 Plateau vs slope binning rules**

Radial “E-like” tests distinguish:

* **Slope readout:** per-cell rate vs on **log-spaced bins**; fit a mid-band window in log–log.
* **Plateau readout:** totals per annulus on **equal-Δr (linear) bins**, dropping trailing partial bins and evaluating CV over an outer fraction.

This bin discipline is mandatory because mixing log bins into plateau tests breaks the intended geometric cancellation.

**6.3 Direction-aware edges for circulation**

For loop/sector flows, “CW” and “CCW” are defined relative to the loop’s declared forward step. Edges and overlap counts must be sign‑correct and consistent with the step convention.

**6.4 Hashes and single-read invariants**

Any simulation involving hinge/measure files must enforce “single physical read” with recorded hashes. Diagnostic changes may not change the underlying measure.

**6.5 Mesh equivalence: compare like with like**

Cross-mesh certification requires **physically aligned bins/windows**. Amplitude comparisons should use **width-invariant density** measures where appropriate so the fine mesh does not appear to “double counts” purely due to more cells.

**7) What “Pass” Means in This Framework**

A PASS does **not** mean “fit whatever is desired.” It means:

1. **Control contract satisfied**: present‑act, boolean/ordinal, deterministic; no diagnostics feedback.
2. **Readouts computed correctly** (right bins/windows, correct conventions).
3. **Invariants hold** under declared perturbations:
   * mesh changes,
   * isotropic masks,
   * center shifts (recentering),
   * duty/scheduling variations that are allowed by the sim.
4. Any amplitude scaling induced by geometry is **predicted** by a simple container fraction and verified within tight tolerance, while **form** remains invariant.

**8) Theory Interpretation (Why This Engine Matters)**

The MATTER V2 engine is intentionally minimal: it demonstrates how behaviors commonly attributed to continuous fields can emerge as **count‑laws** from discrete eligibility constraints:

* “Field-like” radial laws arise from deterministic frequency schedulers (e.g., DDA ) and geometric aggregation.
* “Flow-like” behavior arises from directed eligibility along loops and stable overlap structures.
* “Phase/polarization” and correlation structure arise from sector schedules and consistent lag conventions, not introduced wave equations.
* “No-signalling” arises because remote settings are not present in local eligibility predicates, and schedule balance ensures unbiased sampling.

In all cases, the engine maintains the same ontological stance: **there are only present acts and their relational traces**; anything resembling a classical field is a measurement of coordinated eligibility, not an assumed potential.

**9) Minimal Reproducibility Requirements (For Each Published Run)**

A published run should archive:

* the exact manifest (or cryptographic hash),
* engine source version identifier,
* diagnostics configuration (or hash),
* seeds (if any, and tie-only policy),
* the result line (PASS/FAIL + key metrics),
* metrics CSV/JSON outputs sufficient to rerun audits,
* environment snapshot (Python version, platform).

**10) Implementation Summary (One Paragraph)**

The MATTER V2 core engine evolves a discrete lattice over discrete ticks by selecting a set of acting sites each tick using only integer, boolean/ordinal eligibility predicates (shell/sector membership, duty windows, optional parity rails, neighbor support, cooldown, and deterministic DDA schedulers). It is deterministic by default; randomness is permitted only for explicit tie-breaking and is isolated and audited. All physics-like behavior is evaluated through post-hoc diagnostics computed from counts and overlaps using fixed, artifact-resistant binning and window rules, with mesh checks required for claims sensitive to discretization. This separation—**form in control, numbers in the container, diagnostics as readout only**—is the core design invariant across the V2 simulation suite.