**Overall V2 / Absolute Relativity — Core Engine (v0.9)**

**Scope.** This document specifies the **present‑act engine core** that all V2 simulations use. It describes the conceptual model, math interfaces, shared pipeline stages, configuration schema, I/O contracts, uncertainty treatment, gating/plateau detection, geometry options, and scene‑specific hooks for **Triad**, **T1**, **T2**, **T3**, and **T3‑B**. No large external datasets are required to understand or execute the engine logic.

**1) Conceptual model (engine invariants)**

**1.1 Present‑act engine**

The engine implements **feasibility geometry**: a gating mechanism over candidate evolutions that depends on a **container scale** and a universal amplitude . Gravity appears as **thinning** (suppression) structured by container boundaries.

* **Universal constant:** (fixed once via UGM→Earth calibration).
* **Container factor:** , a monotone function of container radius relative to UGM.
* **Scene observable:** An experiment‑specific mapping from thinning to a measured amplitude (deflection/delay/redshift; rotational plateau; RAR; lensing plateau).

**1.2 Decomposition used across scenes**

For any scene’s observable ,

where:

* is the “baseline” (e.g., Newton/GR‑like or instrument baseline).
* The **container term** is the scene’s readout of boundary thinning; collects local factors (e.g., mass proxy, geometry weight).
* In size‑controlled tests (T3/T3‑B), the **sign** of at fixed mass diagnoses whether the container term is active.

**2) Engine architecture (modules and flow)**

The core engine is a **small set of reusable stages** that all scenes call, with scene‑specific adapters.

**2.1 Configuration layer**

* **YAML study config** (flat keys; no framework dependency) defines:
  + **Data mode** (e.g., PRESTACKED for T3; direct tables for T1/T2/Triad).
  + **Bin edges** (size/mass or radial bins), **gates** (slope threshold, min bins), **geometry flags** (use\_geo\_in\_amplitude), **bootstrap\_n**, **random\_seed**.
  + **Paths** to input tables (or generators) and where to write outputs.

**Contract:** Every scene’s run\_\*.py reads a single YAML and writes standardized CSV/JSON outputs.

**2.2 Input & data access layer**

* **Readers** ingest lens/source/rotation/RAR/triad tables (FITS/CSV) into **pandas DataFrames** with strict column contracts (see §5).
* **Optional builders** generate:
  + **Random controls** (e.g., random lenses for T3 subtraction).
  + **Derived fields** (e.g., circularized size in kpc).
* **Caching** allowed but never required (engine runs from tables alone).

**2.3 Corrections & normalization**

* **Systematic corrections** exposed as toggles (e.g., **m‑correction** for shear).
* **Randoms subtraction** (T3): subtract additive signals using matched randoms.
* **Geometry normalization** (optional for T3): multiply/divide by or equivalent weight to compare stacks at different lens/source redshifts.

**2.4 Plateau / boundary detector (shared math)**

A single **flat‑window detector** is used wherever a plateau/boundary level must be read (T1 rotation plateau, T3 lensing plateau). It implements:

1. **Pre‑smoothing** (robust moving median, small window, e.g., 3–5 bins) on the plotted series **only** to stabilize gradients (level preserved).
2. **Gradient gate**: require over a **contiguous window** of at least min\_bins; is the scene’s abscissa (e.g., in T3).
3. **Window selection** in a **declared range** (e.g., – for T3).
4. **Amplitude**: the **median** of the **unsmoothed** series on the accepted window.
5. **Flatness QC**: RMSE and recorded for diagnostics.
6. **Claimable flag**: boolean pass/fail of the gate.

All sims that measure a boundary/plateau call this detector with scene‑specific .

**2.5 Uncertainty engine**

* **Bootstrap / MC** sampler with fixed random\_seed, default **percentiles (16–84%)** stored alongside amplitudes/slopes.
* **Weighted fits** derive from (84–16)/2; if undefined, a minimal positive fallback is used to keep weights finite.
* **Robust slopes** (Theil–Sen) available as alternative to WLS.

**2.6 Regression / contrast layer**

Common routines implement:

* **Within‑bin slopes** (e.g., vs at fixed ).
* **Contrasts** targeting boundary activation (e.g., outer–minus–mid for T3, inner–minus–mid for T1).
* **Model comparison** (AIC) between:
  + **size‑only** (baseline trend), and
  + **activation** (adds a term tied to container activation metric; see T3‑B).

**2.7 Output layer**

* Human‑parsable **CSV/JSON** files, one row per **stack** or per **scene summary**:
  + **Plateau table** with amplitude, CI, QC metrics, claimable flag.
  + **Regression/contrast** JSON with slopes, CIs, and goodness‑of‑fit.
  + **Scene‑specific summaries** (e.g., triad deflection/delay/redshift tuples).
* Figures are optional, never required for programmatic consumption.

**3) Shared math used by all V2 sims**

**3.1 Container amplitude & sign logic**

* The **container term** has **positive monotone coupling** to container scale : .
* Competing **baselines** generally **decrease** with “spread”/size at fixed mass (e.g., GR‑like lensing baseline ).
* The engine therefore tests **sign flips** or **contrast rises** that indicate the container term becoming dominant in the target regime.

**3.2 Geometry normalization (optional)**

* Distance factors (e.g., ) can be brought into the amplitude to compare stacks **fairly** across redshift distributions.
* Toggle is **per study** (use\_geo\_in\_amplitude) and recorded in outputs.

**3.3 Hinge/activation pattern (T3‑B generalization)**

* For any scene where a **reference container scale** is hypothesized (e.g., **Milky Way** for galaxies), the engine supports an **activation variable** and **cutoff tests**:

or hinge in . This is implemented as a **model‑comparison layer** (size‑only vs size+activation), not a change to core gating.

**4) Scene adapters (how each sim plugs into the engine)**

All scenes share the **same core services** (config → ingest → corrections → plateau/boundary → uncertainty → regression/contrast → outputs), differing in **what is**, **what bins exist**, and **what contrasts/regressions are evaluated**.

**4.1 Gravity Triad (Earth container)**

* **Observable(s):** deflection, Shapiro delay, gravitational redshift from Earth’s container scale.
* **Inputs:** constants from calibration (UGM length; Earth container radius ; ).
* **Core calls:** no large data; uses boundary readouts and deterministic formulas that invoke .
* **Outputs:** a small JSON/CSV with predicted/observed tuples and residuals.

**4.2 T1 — Rotation plateau**

* **:** rotation‑curve transform that produces a plateau region (scene‑specific definition).
* **Gates:** flat‑window detector over a declared -range; amplitude = median on the window.
* **Contrasts:** inner–mid / outer–mid depending on the rotation context.
* **Slopes:** within control bins (if applicable).
* **Outputs:** plateau table + contrasts/slopes.

**4.3 T2 — RAR (Radial Acceleration Relation)**

* **Observable:** relation between observed and baryonic acceleration; boundary imprint alters slope/intercept.
* **Core calls:** regression layer with uncertainty engine; optional geometry normalization if defined.
* **Outputs:** RAR coefficients with CIs; diagnostic contrasts if a plateau‑like readout is defined.

**4.4 T3 — Galaxy–galaxy lensing plateau**

* **.**
* **Corrections:** shear **m‑correction**, **randoms subtraction** (lenses vs random lenses).
* **Gates:** flat‑window in –; slope threshold; minimum bins; QC metrics.
* **Amplitudes:** per (mass bin, size bin) **stack**; CI via bootstrap/MC.
* **Regressions:** vs at fixed (WLS + Theil–Sen).
* **Outputs:** lensing\_plateau.csv, windows/QC JSONs; size‑slope JSON.

**4.5 T3‑B — Milky‑Way–anchored activation**

* **New variable:** (with small grid over ).
* **Linkage:** For each stack, compute frac\_x\_gt (fraction of lenses with ) from lens catalog.
* **Model compare:** per mass bin, **size‑only** vs **size+activation** (adds frac\_x\_gt) using WLS; sum ΔAIC across bins.
* **Outputs:** scan CSV over , JSON summary with best ΔAIC, and classic **P(out>mid)** contrasts (MC).

**5) Configuration & I/O contracts**

**5.1 Study config (YAML; shared keys)**

* study\_id: string
* mode: "PRESTACKED" | "DIRECT" | ...
* prestack\_csv: path (if used)
* prestack\_meta\_csv: path (if used)
* plateau\_slope\_abs\_max: float
* min\_bins: int
* min\_b: float, max\_b: float (abscissa range for window)
* bootstrap\_n: int
* random\_seed: int
* use\_geo\_in\_amplitude: bool

**5.2 Required columns (typical)**

**Lens‑level (when needed):**  
lens\_id, ra\_deg, dec\_deg, z\_lens, R\_G\_kpc, Mstar\_log10, R\_G\_bin, Mstar\_bin.

**Stack‑level (prestacked):**  
Mstar\_bin, R\_G\_bin, bin\_center (scene‑specific x), P(x), n\_pairs, ... (T3 has its own bin schema for ).

**Plateau output (lensing\_plateau\*.csv):**  
Mstar\_bin, R\_G\_bin, A\_theta (or A), A\_theta\_CI\_low, A\_theta\_CI\_high, rmse\_flat, R2\_flat, win\_start, win\_stop, win\_nbins, claimable.

**Regression output:**  
Per mass bin: slope, intercept, CI\_16, CI\_84, n\_stacks, r2, method.

**T3‑B stack‑x table:**  
Mstar\_bin, R\_G\_bin, n\_lenses, x\_median, x\_p25, x\_p75, frac\_x\_gt, frac\_x\_around, R\_MW\_kpc, eta.

**6) Gating, QC, and uncertainty (shared behavior)**

* **Gate first, analyze later:** All selections (claimable stacks) are determined by boolean/ordinal gates (**no curve‑fitting to steer selection**).
* **Pre‑smoothing** is **only** to stabilize gradients for window finding; amplitudes are computed from the **unsmoothed** series.
* **QC metrics** (RMSE, , window width) accompany every accepted window.
* **Uncertainty:** 16/84% intervals via bootstrap/MC are propagated to all regression/contrast layers; WLS weights with safe fallbacks.

**7) Geometry & reproducibility**

* **Geometry toggle**: When enabled, amplitudes are normalized by the appropriate distance factor to make stacks comparable (recorded in outputs).
* **Seeds & determinism:** random\_seed ensures reproducibility of bootstrap/MC; configs freeze bin edges and gates.
* **Manifests:** Each run stores paths and parameters used (study ID, gate values, toggles) alongside results.

**8) Performance & scaling**

* **Parallel prestacking** (T3): multi‑process scan/accumulation over lenses/sources; CPU‑bound, memory‑frugal (chunked).
* **Vectorized analysis:** plateau detector and regressions are vectorized or batched per stack; MC draws use a single RNG instance for reproducibility.
* **I/O design:** long‑running downloads/prestacks are **out of band**; the engine consumes tables that can be rebuilt on another machine.

**9) Extensibility & scene reuse**

* New scenes plug in by defining:
  1. **Their** or boundary observable,
  2. **Their binning and valid window range**,
  3. **Which regressions/contrasts** to compute,
  4. **What geometry normalization** (if any) means for the scene.

Everything else (config, data access, plateau detection, uncertainty, outputs) is **shared**.

**10) Minimal “How it all fits” (one view)**

1. **Config in →** YAML declares data paths (or PRESTACKED), bins, gates, seeds, geometry toggle.
2. **Ingest & correct →** read tables, build randoms/corrections if requested.
3. **Plateau/boundary →** smooth‑for‑gradients → gate a flat window → amplitude from **unsmoothed** data → QC & claimable.
4. **Uncertainty →** bootstrap/MC 16/84% CI.
5. **Analysis →** per‑scene slopes/contrasts; for T3‑B, activation variable + model compare.
6. **Out →** CSV/JSON results; optional figures.

**11) Glossary of common symbols**

* : container radius (scene‑specific; Earth radius, galaxy structural size, etc.)
* : universal present‑act amplitude, fixed via UGM→Earth
* : container scaling factor (monotone in )
* , : measured amplitude (plateau level / boundary readout)
* : galaxy structural size (kpc)
* : stellar mass;
* : activation variable (size normalized to a reference container scale)
* : lensing geometry factor

**12) Scene‑neutral contracts (what verifiers need)**

* **No hidden steering.** All selection proceeds via **declared gates**; analysis layers only consume what gates pass.
* **One‑file inputs.** Each scene consumes **plain tables**; large external datasets are referenced by name, not bundled.
* **Portable outputs.** All results are **text CSV/JSON** with documented columns.
* **Deterministic switches.** Seeds, bin edges, and flags are persisted in configs and (optionally) small manifests.

**Appendix A — Typical folders in a V2 repo (for orientation)**

* src/ — engine and scene code (e.g., src/t3/plateau.py, src/t3/run\_t3.py, plus T1/T2/Triad).
* scripts/ — runnable helpers (download/build, prestack, regressions, T3‑B diagnostics).
* config/ — YAML studies (bin edges, gates, toggles, paths).
* data/ — **external** datasets (removed in archives; re‑fetched by scripts).
* outputs/ — small CSV/JSON results (kept selectively).
* figures/ — optional plots (kept selectively).