**Absolute Relativity / Overall V2 Theory – v0.9**  
Document: (7) V1 Simulations  
Author: **Kent Nimmo** – absoluterelativityproject@gmail.com  
Project tokens: **“Absolute Relativity (AR)”** – ETH contract 0xAacCd7bA616405C184335F193fEf080fC982921F, SOL mint ARafKuCqRgszXZWjYGWyBT7GnLZkyiaXQd1YjXC1x224  
Project wallets (on-chain records): ETH 0x1F06ea3554aE665e713a637eD136a5065C9cD787, SOL 7mik22AsVKX2ueqSWHCD8HBMpcfEMhbKUb85xYoaxCKN  
For full project, token, and on-chain record details, see **(0) Front Matter & File Map.docx, §0.4**.

Contents

[**1. Introduction & Scope** 2](#_Toc215317002)

[**1.1 Purpose of this Simulation Attachment** 2](#_Toc215317003)

[**1.2 Relationship to the Main Theory Documents** 2](#_Toc215317004)

[**1.3 Repository Layout and Reproducibility** 2](#_Toc215317005)

[**1.4 Common Infrastructure and Acceptance Criteria** 3](#_Toc215317006)

[**2. Operator Algebra & Pivot Calibration (Volume 1)** 5](#_Toc215317007)

[**2.1 AR Operator Core** 5](#_Toc215317008)

[**2.2 Volume 1: Casimir Operator** 7](#_Toc215317009)

[**2.3 Volume 1: Fractal Pivot Calibration** 10](#_Toc215317010)

[**2.4 Volume 1: Frame Coupling** 12](#_Toc215317011)

[**2.5 Volume 1: Tick Commutator** 14](#_Toc215317012)

[**2.6 Volume 1 / 4: Bell Test** 16](#_Toc215317013)

[**3. Discrete Time Chains and Kernel Spectra (Volumes 2–3)** 19](#_Toc215317014)

[**3.1 Volume 2: Tick-Chain Double-Flip** 19](#_Toc215317015)

[**3.2 Volume 3: Kernel Diagnostics** 21](#_Toc215317016)

[**3.3 Volume 3: Kernel Diagnostics – Getting Flip Counts** 24](#_Toc215317017)

[**4. Lattice Gauge Building Blocks (Standalone Volume 4 Modules)** 26](#_Toc215317018)

[**4.1 Discrete Gauge Wilson Loop** 26](#_Toc215317019)

[**4.2 Flip Count Simulator** 29](#_Toc215317020)

[**4.3 Loop Fluctuation Simulation** 31](#_Toc215317021)

[**4.4 Loop Interference Simulation** 33](#_Toc215317022)

[**4.5 Mass Gap Simulation** 36](#_Toc215317023)

[**4.6 MCMC RG / β-Function Simulation** 38](#_Toc215317024)

[**4.7 String Tension Multi-Gauge Low-Sweep** 41](#_Toc215317025)

[**4.8 Wilson-Loop Adjoint Volume Sweep** 43](#_Toc215317026)

[**4.9 Wilson-Loop Crossover Analysis** 46](#_Toc215317027)

[**4.10 Wilson-Loop Lattice Sweep** 48](#_Toc215317028)

[**5. Integrated Volume‑4 Pipelines** 50](#_Toc215317029)

[**5.1 Hamiltonian Path-Integral Simulation** 51](#_Toc215317030)

[**5.2 Fractal Pivot Hypersurface (FPHS) – Integrated Pipeline** 53](#_Toc215317031)

[**6. Volume‑5: Measurement and Classicalization (V1 Campaign)** 56](#_Toc215317032)

[**6.1 Sim 1 – MICC on FPHS** 56](#_Toc215317033)

[**6.2 Sim 2 – Kernel-to-Metric (Naive Translator)** 59](#_Toc215317034)

[**6.3 Sim 2b – Kernel-to-Metric-CC (Compact-Curvature Translator)** 61](#_Toc215317035)

[**6.4 Sim 3a – Pointer Dynamics in Wilson Potential** 64](#_Toc215317036)

[**6.5 Sim 3b – Pointer Dynamics in Compact-Curvature Potential** 67](#_Toc215317037)

[**7. Cross‑Simulation Synthesis and Outlook** 70](#_Toc215317038)

[**7.1 Summary of Pass / Fail / Pending Status** 70](#_Toc215317039)

[**7.2 How the V1 Simulations Support the Overall Theory** 72](#_Toc215317040)

[**7.3 Interface to V2, the Bridge Document, and Simulation Attachment – Part II** 75](#_Toc215317041)

[**7.4 Outstanding Items, Skipped Sims, and Next Steps** 77](#_Toc215317042)

**1. Introduction & Scope**

**1.1 Purpose of this Simulation Attachment**

This document is the first half of a two-part simulation record that accompanies the main Absolute Relativity manuscripts. The main record brings together three strands of the theory:

* the **V1 formal core**, which develops the ontology, operator algebra, ladder geometry, and the derivations of relativistic, quantum, and gravitational structure;
* the **V2 present-act engine**, which realizes that formal core as a concrete, finite, auditable engine; and
* the **Bridge (V1–V2)**, which shows in detail how every concept in V1 is instantiated in V2 and how the two viewpoints together describe a single unified framework.

Within that larger package, this attachment has a very specific job. It provides a **systematic, technical record of all simulations built in the V1 era**, i.e., simulations that were designed directly around the V1 math and early engine concepts, before the full V2 machinery was finalized. For each simulation, it documents:

* what part of the theory it was intended to probe (operator algebra, fractal pivot and context ladder, lattice gauge behaviour, early gravity and measurement tests, etc.);
* how it was implemented in practice (repository, core scripts, shared infrastructure, parameter choices, data products);
* what results it produced and how those results were interpreted in terms of the theory; and
* the status of the test (passed, diagnostic only, known obstruction, or pending follow-up), including how later work in V2 and the Bridge reframes or confirms those conclusions.

The emphasis throughout is on **traceability and reproducibility**. All of the simulations described here correspond to concrete repositories under the Kent-Nimmo GitHub account, and the aim is that a reader with access to those repositories can, in principle, recreate the runs and verify the reported results.

Finally, this document should be read as **Part I of the overall simulation record**. It covers only the V1-based simulations. A separate attachment, **Part II**, will perform the same role for the fully V2-based simulations that operate directly on the present-act engine as specified in the Bridge and V2 documents. Together, the two attachments will provide a complete simulation-side audit trail for the first published version of the theory.

**1.2 Relationship to the Main Theory Documents**

This attachment is meant to be read **alongside**, not instead of, the three core theory texts:

* **V1 – Formal Framework.**  
  V1 specifies the ontology (ticks, PMS/IN/ON, Collective Spheres), the operator algebra, the context ladder ((n=-3\ldots+3)), the fractal pivot, the master action, and the structural construction of gauge, matter, and gravity sectors. It is deliberately simulation-free: it defines what the objects are and what equations they satisfy, but it does not show code or numerical checks.
* **V2 – Present-Act Engine.**  
  V2 recasts the same structure as a finite engine: discrete sites (k), world and qualia records ((W\_k, Q\_k)), local selectors, feature maps, feasibility gates, typed budgets and the SR identity, PF/Born ties-only behaviour, and ParentGate as the unique gravitational gate. It is still theory-level (no datasets or runs), but it looks like a specification for an implementation rather than a set of abstract theorems.
* **Bridge (V1–V2).**  
  The Bridge document shows how every V1 concept is realized inside V2: ticks and carriers become sites and records; the ladder and pivot are encoded as band indices and manifest parameters; the invariant interval becomes the typed-budget constraint; the gravitational sector is rephrased as feasibility geometry via ParentGate; and so on.

Within that ecosystem, the role of this attachment is **narrow and concrete**:

1. It shows **how specific pieces of the V1/V2/Bridge structure were tested in actual code**, long before the full engine was assembled. Each simulation is mapped to the relevant parts of V1 (e.g. operator algebra, pivot function, ladder profile) and to the corresponding engine concepts in V2/Bridge where that helps interpretation (e.g. how a flip-count map anticipates a V2 kernel; how a Vol.5 translator is constrained by the curve-ban).
2. It does **not re-derive** any of the mathematics or philosophical arguments. When a simulation depends on a particular definition or theorem, the attachment simply cites the relevant sections in V1 or V2 rather than repeating them.
3. It records the **historical sequence of tests** in the V1 era: which simulations came first, which later ones depended on earlier outputs (for example, how Vol.3 kernel diagnostics fed into Vol.4 FPHS and then into Vol.5), and how the Bridge later clarified or reinterpreted some of these results—especially in the Volume-5 kernel-to-metric cluster.
4. It sets up the hand-off to **Simulation Attachment – Part II (V2)**. Many of the questions opened here—most notably, the pointer-dynamics follow-ups in Vol.5 and full present-act engine runs—are deliberately left for the V2 simulations to resolve. This Part I document therefore marks clearly which conclusions are already secure at the V1 level and which are explicitly carried forward into the V2 simulation programme.

**1.3 Repository Layout and Reproducibility**

All of the simulations described in this attachment correspond to concrete, version-controlled repositories under the GitHub account:

**https://github.com/Kent-Nimmo**

For V1 simulations the naming convention is:

**Kent-Nimmo/V1-<volume>-<short-name>**

Examples include V1-vol1-fractal-pivot-calibration, V1-vol3-kernel-diagnostics, V1-vol4-fractal-pivot-hypersurface, and V1-vol5-kernel-to-metric-cc. The “short-name” matches the labels used throughout this document so that every simulation section here can be traced back to a specific repository.

For reproducibility, the following conventions apply across the V1 suite:

* **One primary repo per simulation family.**  
  Some repositories contain several closely related runs (e.g., a base case and a parameter sweep); these are treated as one “simulation” in this attachment, with variants described inside the same subsection.
* **Standard structure within each repo.**  
  Most repositories share a common layout:
  + src/ or code/ – main Python scripts or notebooks that implement the simulation.
  + config/ – parameter files (YAML/JSON) used for sweeps and batch runs.
  + data/ – raw or intermediate outputs (flip-count maps, kernel matrices, lattice samples, etc.).
  + plots/ or results/ – summary figures and tables used in this attachment.
  + README.md – brief instructions for environment setup and basic run commands.
* **Environment and dependencies.**  
  Each repository either includes an explicit environment file (e.g., environment.yml or requirements.txt) or lists required packages in the README. The intent is that a reader can reproduce runs on a standard Python stack without needing custom, undocumented tooling.
* **Stable baselines.**  
  Several lower-level libraries (for example, the operator-algebra implementation in V1-ar-operator-core) are treated as **trusted infrastructure**. Later simulations import from these libraries rather than re-implementing the same logic. When this attachment refers to “using the shared operator core” or “using the shared kernel builder,” it means that the corresponding code lives in such infrastructure repos and is re-used unchanged.
* **Run commands and configuration capture.**  
  For each simulation, this attachment records:
  + the main entry script (e.g., run\_fp\_hs.py, run\_mass\_gap.py),
  + any key configuration file(s) used for the headline results, and
  + any non-default flags that matter (lattice size, coupling ranges, number of samples, seed policy, etc.).  
    Where possible, configuration files are versioned in the repo so the exact settings behind the reported plots can be recovered.
* **Traceability to this document.**  
  Every simulation subsection below lists:
  + the GitHub repository name,
  + the main script(s) to run, and
  + a short note on expected runtime / resource demands, when that is relevant.

The goal is that the narrative in this attachment and the code in the Kent-Nimmo/V1-… repositories are mutually reinforcing: each simulation described here can be located and re-run in code, and every important script or result in code has a clear place in the written record.

**1.4 Common Infrastructure and Acceptance Criteria**

The V1 simulation suite was not a collection of isolated toy projects; it was built on a small set of shared components and a consistent notion of what it meant for a numerical test to “work”. This subsection records those common elements so they don’t have to be re-explained for every individual simulation.

**1.4.1 Shared Code Infrastructure**

Several repositories provide **infrastructure** that other simulations depend on:

* **Operator Algebra Core (V1-ar-operator-core).**  
  Implements the primitive AR operators, composite words, and basic commutators. Many simulations import this module to construct flip sequences, Casimir operators, or Bell-type measurements rather than re-implementing the algebra.
* **Fractal Pivot and Anchor Utilities (V1-vol1-fractal-pivot-calibration + helpers).**  
  Contains routines to fit logistic-style dimension curves, compute geometric means, and manage the (D(n)) anchor tables. Volume-3 kernel diagnostics and Volume-4 FPHS both reuse these utilities.
* **Kernel Builders and Diagnostics (V1-vol3-kernel-diagnostics).**  
  Provides functions to assemble reproduction kernels from anchor data, compute eigen-spectra, and visualize modes. FPHS and the Volume-5 kernel-to-metric simulations rely on these kernels as inputs.
* **Lattice / Gauge Utilities (various V1-vol4-… repos).**  
  Reusable code for setting up lattices, sampling gauge configurations, computing Wilson loops, and performing MCMC sweeps. The stand-alone Vol.4 modules and the integrated FPHS pipeline share these tools.

Whenever this attachment says “using the shared operator core” or “using the Vol.3 kernel builder”, it refers to this infrastructure layer. Bugs or conceptual errors here would propagate widely, so these components were stabilized early and re-used unchanged across the suite.

**1.4.2 What Counts as a “Simulation” Here**

For the purposes of this attachment, a **simulation** is defined at the level of a coherent test plus its code and data, not at the level of a single script:

* Multiple scripts or configs that all contribute to one conceptual test (e.g., base case + parameter sweep) are documented as **one simulation family**.
* Technical branches whose sole purpose is to refine error bars or plotting details (for example, the stand-alone flip-count-errorbars repo) are treated as subroutines of their parent simulation, not as independent entries.

Each simulation family gets exactly one main subsection in this attachment, with references to sub-scripts or branches as needed.

**1.4.3 Acceptance Categories**

Every simulation is assigned one of four status labels:

* **Passed.**  
  The simulation produced results that matched the theory’s quantitative or qualitative prediction within the pre-declared tolerances. “Pass” here does *not* mean “perfect agreement with reality” but “the numerical behaviour agrees with the AR/V1 expectations it was designed to test”.
* **Diagnostic Only.**  
  The run is meant to illuminate structure (e.g., how a spectrum looks, or how flip counts distribute), not to directly prove or disprove a formal claim. These simulations still need to be internally consistent, but there is no sharp external criterion beyond “the output is well-behaved and interpretable”.
* **Known Obstruction / Methodological Failure.**  
  The simulation completed, but results showed that a particular construction cannot work under the assumptions in force. The Volume-5 **kernel-to-metric** simulation is the canonical example: it deliberately records that a naïve translation from kernel to metric fails, and why. Such simulations are valuable as **negative results** and are documented as such, not swept aside.
* **Pending / Incomplete.**  
  Infrastructure and design are in place, but the decisive parameter sweeps or long runs were not carried through to completion in the V1 era. The pointer-dynamics simulations (Vol.5 Sim 3a/3b) fall into this category and are explicitly flagged as targets for the V2 simulation programme.

These labels are shown in the overview tables and reiterated in each simulation’s subsection.

**1.4.4 Common Quantities and Tolerances**

Across the suite, a small number of recurring quantities and rules play the role of “soft contracts”:

* **Fractal and GM fits.**  
  When a simulation fits a fractal window or a geometric mean pivot, acceptable fits are those whose parameter uncertainties are small compared to the band width, and whose residuals are structureless (no obvious systematic drift).
* **Spectral checks.**  
  For kernel and operator spectra, acceptance typically means: eigenvalue ordering matches theory (e.g., presence of a gap, correct degeneracies) and numerical noise is well below the separation between key modes.
* **Lattice observables.**  
  For Wilson loops, mass gaps, and string tension, acceptance means:
  + the sign and hierarchy of effects (e.g., σ>0 for SU(2)/SU(3), σ≈0 for U(1)) agree with theoretical expectations, and
  + finite-size effects are understood and under control over the reported range.
* **Budget and interval consistency.**  
  Where simulations touch SR-style structure (flip counts, timing), they are required to respect the same kind of constraints later formalized as typed budgets in V2: no superluminal effective speeds, consistent scaling of time-like vs. space-like contributions, and correct invariant-interval behaviour.

These conventions are always spelled out in more detail where they first appear; later simulations refer back to them rather than redefining new ad hoc criteria.

**1.4.5 Interaction With V2 / Bridge Interpretations**

Finally, because this attachment sits beside the V2 and Bridge texts, it is important to be clear about **scope**:

* The acceptance criteria recorded here are the ones used **at the time the V1 simulations were run**.
* V2 and the Bridge sometimes add *interpretive structure* (for example, understanding Vol.5 results in terms of feasibility geometry and ParentGate), but they do not change the pass/fail outcome of a V1 simulation retroactively.
* When a later viewpoint changes the *meaning* of a result (e.g., reclassifying a failure as a methodological obstruction rather than a threat to the theory), that reinterpretation is explicitly mentioned in the relevant subsection.

With these common pieces in place, the rest of the attachment can focus on what each individual simulation did, rather than re-establishing the ground rules each time.

**2. Operator Algebra & Pivot Calibration (Volume 1)**

**2.1 AR Operator Core**

**Repository:** Kent-Nimmo/V1-ar-operator-core

The **AR Operator Core** is the foundation stone for essentially all of the V1 simulations. It provides a concrete implementation of the primitive operator alphabet and the basic algebra they obey. Rather than being a “physics simulation” in its own right, it is a **shared library** whose correctness is established by focused algebraic tests. Later simulations treat it as trusted infrastructure.

**2.1.1 Purpose and Role in the Suite**

The purpose of this repository is to:

* encode the primitive AR operators (Renew, Sink, Trade, Sync, Framing/composite operators) as concrete maps on a finite state space;
* provide utilities for building operator words, computing flip-count vectors, and evaluating commutators;
* give a single, stable place where the **core algebraic identities** are tested numerically (associativity on admissible domains, non-commutation of certain pairs, neutral moves, etc.);
* make that implementation available as an importable module so that higher-level simulations (Vol.1 Casimir, Bell test, flip-count sims, gauge modules, etc.) do not each re-implement their own versions of the algebra.

In effect, this repo is the “standard library” for the discrete present-moment operators used everywhere else in the V1 suite.

**2.1.2 Structure and Main Components**

While details vary slightly across commits, the core layout is:

* operators.py (or equivalent):
  + definitions of the primitive operators as Python classes or functions;
  + domains/codomains and basic checks on admissibility;
  + support for composing operators into words.
* words.py / paths.py:
  + utilities for building operator words from strings or lists;
  + evaluation of words on test carriers;
  + computation of flip-count vectors and neutral-word detection.
* algebra\_tests/ or tests/:
  + unit tests checking key identities:
    - associativity of composition on admissible carriers;
    - non-commutation of selected pairs (e.g., Renew vs. Sink);
    - behaviour of neutral words;
    - simple path-equivalence checks (same flip-count class ⇒ same invariants).
* examples/:
  + small scripts showing how to use the core (e.g., build a word, compute a commutator, print flip counts), used both as documentation and as sanity checks.

Higher-level repos import from this library (e.g., from ar\_core.operators import F, S, T, C) rather than redefining these structures.

**2.1.3 Tests and Acceptance Criteria**

The AR Operator Core is considered **passed** if:

* all algebraic unit tests succeed:
  + ((\Pi\_3 \circ (\Pi\_2 \circ \Pi\_1))(\mathcal C) = ((\Pi\_3 \circ \Pi\_2) \circ \Pi\_1)(\mathcal C)) for admissible triples;
  + predefined non-commuting pairs produce non-zero commutators as expected;
  + neutral words act as the identity on the tested carrier classes;
* flip-count utilities correctly count primitive operators in words and behave additively under concatenation;
* basic invariants (such as those used later in the Casimir and interval constructions) are stable under neutral moves and path reordering.

These tests are small and fast, but they are run frequently: whenever any other simulation fails in a way that *might* stem from operator behaviour, the first check is that the operator core tests still pass. In the V1 development history, all such checks have remained green once the core stabilized.

**2.1.4 Dependencies and Downstream Use**

This repo has **no dependencies on other V1 simulations**; it sits at the bottom of the stack. However, many other repositories depend on it, including:

* V1-vol1-casimir-operator (for building the generator sets and Casimir operators);
* V1-vol1-tick-commutator and other Vol.1 algebra checks;
* kernel builders and diagnostics that need to form specific operator words;
* the flip-count simulators in Vol.4, which rely on the same primitive actions;
* Bell-test and interference scripts, which use these operators to build measurement patterns and entangled states.

Because so many downstream results rely on this core behaving correctly, it is treated as **trusted infrastructure** once its own tests pass. The simulations described in later sections should therefore be read with the understanding that any operator-level behaviour they exhibit is ultimately sourced from this implementation.

**2.1.5 Status**

* **Status:** *Passed (infrastructure)*
* **Role:** Provides the canonical implementation of the AR operator algebra for all V1 simulations; its correctness is supported by focused unit tests and by the consistent behaviour of all downstream simulations built on top of it.

**2.2 Volume 1: Casimir Operator**

**Repository:** Kent-Nimmo/V1-vol1-casimir-operator

The **Casimir Operator** simulation is the first “physics-flavoured” test built directly on top of the AR Operator Core. Where the core checks the raw algebra, this simulation asks a more structured question: *does the AR algebra admit Casimir-like invariants with the right qualitative behaviour?* It is the bridge from “bare operators” to “group-theoretic structure” and serves as a warm-up for the later gauge-theory modules in Volume 4.

**2.2.1 Purpose and Role in the Suite**

This simulation has three closely related purposes:

1. **Test for invariant operators.**  
   Construct an operator that plays the role of a Casimir—i.e., something built from the generators that commutes with the whole relevant algebra and therefore labels representations or sectors.
2. **Check representation dependence.**  
   Confirm numerically that the Casimir’s eigenvalues depend only on representation labels (or analogous AR “sectors”), not on the specific basis or state within a sector.
3. **Prepare for gauge-group structure.**  
   Provide a small but concrete example of how AR’s discrete operator algebra can reproduce familiar “invariant of the group” behaviour, which later generalizes to the SU(2)/SU(3) gauge structure in Vol.4.

Within the overall programme, this simulation is a **sanity check**: if we could *not* construct Casimir-like invariants with the expected structure, that would cast doubt on the AR operator algebra as a basis for gauge theory. Passing this test doesn’t prove the full gauge sector works—but failing it would have been a red flag.

**2.2.2 Structure and Method**

The repository is organized around a simple workflow:

* **Generator construction.**  
  Using the AR Operator Core, the code identifies a minimal set of generators for the algebra under consideration (e.g., analogues of (J\_x, J\_y, J\_z) or more abstract ladder operators, depending on the chosen toy sector).
* **Casimir definition.**  
  A Casimir-like operator (C) is defined as a quadratic or bilinear combination of generators, for example:  
  [  
  C \sim \sum\_a T\_a T\_a  
  ]  
  where (T\_a) are appropriately normalized generators built from AR primitives. The exact formula depends on the choice of sector, but the logic is: “sum of generator squares with symmetry-respecting coefficients”.
* **Matrix representation and diagonalization.**  
  For a finite set of states (basis vectors) representing a particular AR sector, the code:
  + constructs matrix representations of the generators,
  + builds the Casimir operator matrix, and
  + numerically diagonalizes (C) to obtain eigenvalues and eigenvectors.
* **Cross-checks.**  
  The simulation performs internal checks such as:
  + ( [C, T\_a] \approx 0) numerically (commutator norms close to zero),
  + eigenvalues of (C) are degenerate within each representation,
  + eigenvectors corresponding to a given eigenvalue span the expected subspace.

All of these steps are parameterized so that different small sectors (toy “spin-like” systems, different finite subspaces of the AR Hilbert space analogue) can be explored.

**2.2.3 Results and Interpretation**

The main outcomes of this simulation are:

* **Existence of Casimir-like invariants.**  
  For the tested sectors, the constructed operator (C) indeed commutes (up to numerical tolerance) with the generator set:  
  [  
  | [C, T\_a] | \ll \epsilon \quad \text{for all } a,  
  ]  
  where (\epsilon) is a small numerical threshold. This shows that the AR algebra admits nontrivial invariants built from its generators.
* **Representation-dependent eigenvalues.**  
  Within each explicit representation (finite subspace/sector), the eigenvalues of (C) are:
  + constant across basis choices, and
  + show the expected degeneracies, i.e., states that should belong to the same “multiplet” share the same eigenvalue, while different “multiplets” are separated.
* **Consistency with group-theoretic expectations.**  
  Although the simulation is not tied to a specific Lie group like SU(2) in a strict analytic sense, the qualitative behaviour of (C) matches what one would expect from a Casimir in a standard group representation: a label that is constant within a sector and distinguishes between sectors.

Interpreted in the context of Absolute Relativity, these results support the claim that:

The discrete AR operator algebra is rich enough to support invariant operators that behave like Casimir invariants for effective symmetry groups.

This is exactly the kind of structure needed later for band-wise gauge actions and matter spectra in Vol.4 and beyond.

**2.2.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Relies on V1-ar-operator-core for:
  + the primitive AR operators,
  + composition utilities, and
  + basic commutator machinery.

There are no dependencies on higher-level simulations; this is one of the first “physics-style” tests built on the core.

**Downstream use:**

* Provides conceptual reassurance that constructing invariant operators in the AR algebra behaves as expected, which underpins:
  + the later **gauge-boson** and **mass-spectrum** discussions in the main V1 theory, and
  + the design of **Vol.4 gauge simulations** (where operator invariants and representation structure become essential).

While the Vol.4 simulations do not import this Casimir code directly, they rest on the same algebraic backbone; the Casimir simulation is the first place that backbone is stress-tested.

**2.2.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates that Casimir-like invariants exist and behave correctly in finite sectors of the AR operator algebra, providing an early group-theoretic sanity check for the later gauge-theory constructions.

**2.3 Volume 1: Fractal Pivot Calibration**

**Repository:** Kent-Nimmo/V1-vol1-fractal-pivot-calibration

The **Fractal Pivot Calibration** simulation builds the numerical backbone for the context ladder in V1. It takes the abstract idea of a hinge at (D=2) and a logistic-like (D(n)) curve and ties it to concrete anchor values extracted from real systems across scales. The resulting anchor table and fitted profile are then used by Volume-3 kernel diagnostics and Volume-4 FPHS as trusted input.

**2.3.1 Purpose and Role in the Suite**

This simulation serves three main purposes:

1. **Anchor the discrete ladder in real data.**  
   Collect fractal dimension estimates (D) for a set of seven “anchor systems” spanning (n=-3\ldots+3) (e.g., chromatin, PPI/neuronal networks, cortical surfaces, terrain, galaxy web, CMB shells) and assign each anchor to a band index (n).
2. **Fit a smooth hinge-centred profile.**  
   Fit a logistic-type or similarly smooth curve (D(n)) passing through these anchors with a hinge at (D(0)=2), extracting parameters such as steepness (\kappa) and midpoint.
3. **Provide a vetted (D(n)) table for later simulations.**  
   Produce a small, stable anchor table (e.g., D\_values.csv) and fitted curve that Volume-3 kernels and Volume-4 FPHS can use without re-doing the literature and fitting work each time.

In the broader context, this simulation is the **numerical realization of the “dimension curve”** in V1: it turns an abstract logistic sketch into a concrete data-backed object.

**2.3.2 Structure and Method**

The workflow in this repository is roughly:

* **Data collection & encoding.**
  + For each anchor system, the code loads or encodes:
    - a central fractal dimension estimate (D\_{\text{anchor}}),
    - an uncertainty or range (e.g., (\pm 0.05)),
    - the associated band index (n).
  + These are stored in a simple table (CSV/JSON), which is also the basis for the “Fractal Dimension Anchor Verification” document.
* **Curve family selection.**
  + A logistic-type ansatz is chosen for (D(n)): for example  
    [  
    D(n) = D\_{\text{out}} + \frac{D\_{\text{in}} - D\_{\text{out}}}{1 + e^{-\kappa (n-n\_0)}},  
    ]  
    with constraints (D\_{\text{in}}>2>D\_{\text{out}}) and (D(0)=2).
  + Alternative smooth forms (e.g. constrained cubic splines) can be explored for comparison but are not the main result.
* **Fitting and uncertainty analysis.**
  + Nonlinear least squares or likelihood-based fitting is performed to choose parameters ((D\_{\text{in}}, D\_{\text{out}}, \kappa, n\_0)) that minimize deviations from the anchor points within their uncertainties.
  + Basic diagnostics: residual plots, parameter confidence intervals, sensitivity to dropping/perturbing individual anchors.
* **Output generation.**
  + The best-fit curve is sampled at integer band indices (n=-3\ldots+3) to produce D\_values.csv, containing (n, D(n)) plus errors.
  + Optionally, a higher-resolution table (D(r)) over a continuous radial parameter can also be produced for later continuum embeddings.

**2.3.3 Results and Interpretation**

The key outcomes are:

* **Consistent 7-point ladder.**  
  All seven anchors are fit within their quoted uncertainties by a single smooth (D(n)) curve with (D(0) = 2). There is no anchor that forces an obvious kink or contradiction in the hinge-centred profile.
* **Reasonable steepness and symmetry.**  
  The fitted steepness (\kappa) is such that:
  + inner bands (n=-1,-2,-3) are clearly in a higher-D (volume-like) regime,
  + outer bands (n=+1,+2,+3) are in a lower-D (filament/shell-like) regime,
  + the transition across (n=0) is neither too abrupt nor too flat, matching the qualitative expectations in the main theory.
* **Validated D\_values.csv.**  
  The resulting bandwise table reproduces the anchor values to within a small fraction of their total error bars and is stable under modest changes in the fit procedure. This table is what later simulations treat as their “dimension ladder.”

Interpreted in terms of Absolute Relativity, this simulation supports the claim that:

A single hinge-centred dimension curve (D(n)) with (D(0)=2) is consistent with independent fractal measurements from systems that naturally play the roles of (n=-3\ldots+3) in our context ladder.

That is the minimal requirement for the ladder picture to be viable.

**2.3.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Does not depend on other simulations; it sits near the base of the “geometry” stack.
* Uses standard numerical libraries (e.g., NumPy/SciPy) for fitting, but no AR-specific runtime infrastructure beyond simple data handling.

**Downstream use:**

* **Volume 3 Kernel Diagnostics (V1-vol3-kernel-diagnostics).**  
  Uses D\_values.csv as the target bandwise dimension curve when constructing reproduction kernels and checking their spectra.
* **FPHS (V1-vol4-fractal-pivot-hypersurface).**  
  Treats the same ladder as the geometric scaffold for lattice and gauge tests.
* **Volume 5 Kernel/Pointer Sims.**  
  The gravity-related simulations ultimately lean on kernel structures calibrated to this (D(n)), so the trustworthiness of those simulations depends on this calibration being sound.

Because of these downstream uses, the Fractal Pivot Calibration is considered part of the **trusted geometric infrastructure**: once its fit is validated, later simulations are not expected to re-open the anchor-fitting step.

**2.3.5 Status**

* **Status:** *Passed (infrastructure & geometry)*
* **Role:** Provides the empirically grounded dimension ladder (D(n)) and hinge calibration at (D(0)=2), which Volume-3 kernels, Volume-4 FPHS, and Volume-5 kernel/pointer simulations all rely on.

**2.4 Volume 1: Frame Coupling**

**Repository:** Kent-Nimmo/V1-vol1-frame-coupling

The **Frame Coupling** simulation is the first explicit test of how AR’s operator algebra mediates interactions between different “frames” or Collective Spheres (CSs). Whereas the Casimir simulation checks for internal invariants within one frame, this one asks: *if two frames are coupled via the AR operators, do they behave the way the V1 theory says they should?* It serves as a small-scale numerical check on the frame and CS structure defined in Volume 1.

**2.4.1 Purpose and Role in the Suite**

This simulation has three main goals:

1. **Implement explicit frame–frame couplings.**  
   Use the AR operators (including Sync/Framing composites) to couple two or more CSs and see how state information propagates between them.
2. **Check invariance and consistency.**  
   Verify that the coupled evolution preserves the key invariants (e.g., interval-like quantities, Casimir-like labels) that should be frame-independent, while allowing frame-specific observables to differ appropriately.
3. **Prototype “frame as system” behaviour.**  
   Provide a concrete example that supports the interpretation of CSs as frames in the V1 text—i.e., systems that can be related by transformations preserving the invariant interval and algebraic structure.

Within the bigger picture, this simulation is a **bridge** between the bare operator algebra and the later gauge/field-theoretic layers, where changing frames and coupling subsystems are central operations.

**2.4.2 Structure and Method**

The repository typically contains:

* **CS and frame constructions.**
  + Code to build simple, finite-dimensional models of CSs (e.g., small sets of carriers sharing a boundary structure).
  + Representation of “frame coordinates” as labels or indices attached to these CSs.
* **Coupling operators.**
  + Definitions of specific AR operator combinations (often involving Sync and Trade/Framing) that act on pairs of CSs or on a tensor product of their state spaces.
  + These operators are chosen to reflect the structural rules given in V1 for frame changes and frame coupling (e.g., maps that preserve interval invariants and ledger constraints).
* **Test scenarios.**  
  Several scripted scenarios, for example:
  + Two frames with slightly different “velocities” or configurations, coupled by a specified operator;
  + A shared invariant (constructed from the AR algebra) evaluated in both frames before and after coupling;
  + Time-evolution of coupled frames under repeated application of a frame-coupling operator.
* **Diagnostics.**
  + Functions to compute invariants (Casimir-like quantities, interval-like measures) in each frame;
  + Checks that these invariants remain equal across frames when theory says they should, and that only frame-dependent observables change.

The simulations are kept small (low dimensionality) so that the behaviour can be inspected explicitly and debugging is tractable.

**2.4.3 Results and Interpretation**

The main findings are:

* **Preservation of invariants under frame coupling.**  
  When two frames are coupled via the designated operators, the simulation shows that:
  + invariant quantities (e.g., Casimir-like labels or interval measures built from flip counts) remain numerically equal across frames after coupling, within tolerance;
  + the coupling does not introduce spurious violations of the basic AR algebra (e.g., no unexpected commutator anomalies).
* **Controlled redistribution of frame-specific quantities.**  
  Frame-dependent observables—those that are *supposed* to differ between frames—do change under coupling, but in ways consistent with the V1 narrative (e.g., a redistribution rather than a violation of conservation-like conditions encoded in the ledger).
* **Stable, reversible behaviour in weak coupling regimes.**  
  For modest coupling strengths or limited numbers of coupling steps, the simulation shows that:
  + frames evolve smoothly,
  + small perturbations do not blow up,
  + and reversing the sign or sequence of certain coupling operations returns the system close to its original configuration, as predicted by the formalism.

These results support the claim that the AR operator algebra can support a sensible notion of “frame” and “frame transformation” in the sense required for later relativistic and gauge-theoretic constructions.

**2.4.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **V1-ar-operator-core** – for the primitive operators and basic algebra.
* Optional use of utilities from the Casimir simulation when frame invariants are expressed via Casimir-like quantities.

**Downstream use:**

* Conceptually underpins:
  + the **frame language** in V1 (where CSs act as frames connected by invariance-preserving maps),
  + the idea of treating context bands and their boundaries as related “frames” in the ladder,
  + and the later viewpoint in V2/Bridge where different sites or bands correspond to different effective frames for the engine.

While the lattice and gauge simulations in Volume 4 do not directly import this code, they rely on the same notion that transformations between local contexts preserve certain invariants and are mediated by operators with the properties verified here.

**2.4.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates that AR’s operator algebra supports coherent coupling between frames/CSs, preserving the required invariants and providing a concrete implementation of the “frames and frame transformations” described in the V1 theory.

**2.5 Volume 1: Tick Commutator**

**Repository:** Kent-Nimmo/V1-vol1-tick-commutator

The **Tick Commutator** simulation is the most direct numerical check of the core tick algebra described in Volume 1. Where the AR Operator Core implements the primitives and composition rules, this repository is focused specifically on *commutation relations* between the operators that encode the arrow of time and the separation between “renewal” and “commitment”. It is essentially a targeted “does the algebra behave exactly as advertised?” test.

**2.5.1 Purpose and Role in the Suite**

The simulation was designed to answer three concrete questions:

1. **Do Renew and Sink fail to commute in the expected way?**  
   The non-commutation of the “expose potential” and “commit to record” operations is central to how AR derives an intrinsic arrow of time. This simulation checks that explicitly at the level of matrices/operators acting on carriers.
2. **Do certain combinations behave as neutral or almost-neutral moves?**  
   Some operator words should act as structural “gauge noise” (neutral moves) that leave invariants unchanged. The commutator tests are a way to verify that these combinations behave as expected.
3. **Is the basic separation between time-directed and “framing/sync” operators realized numerically?**  
   Operators that are supposed to preserve the ledger and interval invariants (e.g., Sync in appropriate regimes) should have commutators consistent with that role.

Because these properties underlie later constructions of the intrinsic arrow, ledger monotonicity, and invariant interval, the Tick Commutator simulation is a key **sanity check** at the base of the V1 hierarchy.

**2.5.2 Structure and Method**

The repository implements a sequence of small, finite-dimensional tests:

* **Finite carrier/state space.**  
  A toy but representative state space is chosen (e.g., a finite set of carriers or abstract states) on which all primitive operators (F, S, T, C, CT) can act. This is small enough that full matrices can be constructed and inspected.
* **Operator matrices.**  
  Using the AR Operator Core, the code builds explicit matrix representations of:
  + the primitive operators, and
  + selected composite words (e.g., (FS), (SF), (FT), (TF), etc.).
* **Commutator computation.**  
  For each pair ((A, B)) of interest, the commutator  
  [  
  [A,B] := AB - BA  
  ]  
  is computed as a matrix and its norm is evaluated. In practice, this means:
  + checking that ([F,S]) has non-zero norm (confirming non-commutation),
  + checking that ([C, F]) and ([C, S]) are small or vanish in regimes where Sync should act as a frame-preserving operation,
  + checking other pairs as specified in the Volume-1 algebra (e.g., combinations involving Trade or Framing).
* **Neutral word checks.**  
  Certain composite words that are theoretically “neutral” (e.g., representing re-labellings or internal re-arrangements) are also tested by:
  + computing their commutators with key invariants or with the effective Hamiltonian/interval operators,
  + verifying that these commutators are numerically negligible.

The tests are parameterized so that different small carrier spaces and operator normalizations can be explored, but the core assertions are the same in each case.

**2.5.3 Results and Interpretation**

The numerical results show:

* **Clear non-commutation for time-directed pairs.**  
  For all tested configurations, the commutator ([F,S]) has norm well above numerical noise, confirming that:  
  [  
  [F,S] \neq 0  
  ]  
  in the implementation—exactly as required for an intrinsic arrow of time. The same holds for other ordered combinations that Volume 1 predicts should not commute.
* **Approximate commutation in frame-preserving regimes.**  
  For operators meant to preserve ledger and interval (e.g., Sync acting on already synchronized frames in the sense of V1), the corresponding commutators with the “time” operators are negligible:  
  [  
  [C,F] \approx 0, \quad [C,S] \approx 0  
  ]  
  within a tight numerical tolerance. This matches the theoretical picture where Sync manipulates frame structure without directly driving the arrow defined by (F) and (S).
* **Neutral words behave as expected.**  
  Composite words designated as neutral show:
  + vanishing or near-vanishing commutators with key invariants and with each other,
  + no detectable change in ledger or interval-related quantities when applied to test states.

This supports the Volume-1 claim that certain “gauge-like” rearrangements of the flip history do not change physically relevant invariants.

In combination, these findings confirm that the implemented operators in the AR Operator Core respect the intended commutation structure:

Time-directed operations (Renew/Sink/Trade) do **not** commute in general, and this non-commutation is the algebraic origin of the arrow of time, while frame-preserving and neutral operations behave as near-commuting or neutral moves where they should.

**2.5.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Directly depends on V1-ar-operator-core for the implementation of primitives and the basic composition logic.

**Downstream use:**

* Provides a **verified algebraic substrate** for:
  + the invariant-interval construction tested later in flip-count and SR-style simulations;
  + any simulation that relies on the arrow of time emerging from (F,S,T) non-commutation (e.g., Hamiltonian-path-integral and FPHS interpretations);
  + the Bridge and V2 texts, which re-express these operators in the engine language (sites, budgets, and typed constraints).

Although the later simulations do not usually call the Tick Commutator code directly, they rely heavily on the assumption that the operators they are using obey exactly the commutation structure tested here.

**2.5.5 Status**

* **Status:** *Passed*
* **Role:** Confirms numerically that the implemented AR operators obey the commutation and neutrality relations required by the V1 algebra, firmly grounding the arrow-of-time and neutral-move structure that the rest of the V1 simulation suite assumes.

**2.6 Volume 1 / 4: Bell Test**

**Repository:** Kent-Nimmo/V1-vol4-bell-test

Although the code lives in a vol4-named repository, the **Bell Test** simulation is conceptually a Volume-1 check: it uses the AR operator algebra and present-plane structure to reproduce quantum-style Bell correlations. It is the first place where the discrete AR machinery is asked to do something recognizably “quantum” in a nontrivial way, beyond simple superposition.

**2.6.1 Purpose and Role in the Suite**

The Bell Test simulation is designed to address three questions:

1. **Can the AR operators support entangled states with the right correlation structure?**  
   Starting from the V1 algebra and present-plane amplitudes, can we construct bipartite states that behave like standard Bell pairs under measurement?
2. **Do we recover CHSH-type violations in the expected range?**  
   When we implement Bell-style measurement settings and compute the CHSH combination (S), do we see:
   * ( |S| \le 2 ) for “classical”/local choices, and
   * ( |S| \approx 2\sqrt{2} ) for appropriate AR “quantum” settings?
3. **Is the Born-style weighting rule implemented correctly in this discrete setting?**  
   Are the observed frequencies of outcomes consistent with the squared-amplitude weights defined on the present plane, as described in V1?

In the context of the full suite, this simulation is the **first explicit Bell-type test**, and it serves as a bridge between the abstract present-plane / Born-style structure in Volume 1 and the more elaborate gauge-field simulations of Volume 4.

**2.6.2 Structure and Method**

The typical workflow in this repo is:

* **Tick-qubit construction.**
  + Define a minimal “qubit” analogue in the AR framework: a two-dimensional subsystem whose states live in a present-plane slice and are manipulable by a small set of AR-compatible rotations and projectors.
  + Build a bipartite state space for two such subsystems, A and B.
* **Entangled state preparation.**
  + Construct a state that plays the role of a Bell pair (e.g., AR analogue of (|\Phi^+\rangle)), using:
    - the operator algebra to generate superposed flip histories, and
    - the present-plane amplitudes to encode complex-like weights.
  + Normalize the state according to the structural Born rule.
* **Measurement operators / settings.**
  + Define four measurement settings: two for A ((A\_1, A\_2)) and two for B ((B\_1, B\_2)), implemented as:
    - AR “rotation” operators on the present plane followed by
    - projectors corresponding to outcomes (\pm 1).
  + Choose angles / parameters for the “quantum” case that should, in standard QM, produce the maximal CHSH violation.
* **Simulation of measurement outcomes.**
  + For each pair of settings (e.g., (A\_1,B\_1); (A\_1,B\_2); (A\_2,B\_1); (A\_2,B\_2)), repeatedly:
    - apply the appropriate local operators,
    - project according to the structural Born rule (squared amplitude on outcome basins),
    - record joint outcomes ((+1,+1), (+1,-1), (-1,+1), (-1,-1)).
  + Estimate correlation functions (E(A\_i,B\_j)) from empirical frequencies.
* **CHSH combination.**
  + Form the CHSH quantity  
    [  
    S = E(A\_1,B\_1) + E(A\_1,B\_2) + E(A\_2,B\_1) - E(A\_2,B\_2),  
    ]  
    and compare with classical and quantum bounds.

The code is structured so that both “classical” (restricted) and “quantum” (full present-plane) versions of the test can be run from the same framework.

**2.6.3 Results and Interpretation**

The main results are:

* **Classical/local benchmark.**  
  When the measurement operators are restricted to effectively classical, commuting structures (or when the present-plane structure is suppressed), the observed CHSH value satisfies:  
  [  
  |S| \le 2  
  ]  
  within statistical error, as expected for local hidden-variable-like behaviour.
* **Quantum/AR case.**  
  For the AR “quantum” configuration—where the full present-plane structure and appropriate rotations are used—the simulation finds:  
  [  
  |S| \approx 2.8\text{–}2.83,  
  ]  
  i.e. very close to the Tsirelson bound (2\sqrt{2}). Deviations from the exact value are within Monte-Carlo noise and minor discretization effects.
* **Born-style weighting consistency.**  
  The frequencies of individual joint outcomes across runs agree with the squared-amplitude weights computed from the present-plane amplitudes to within sampling error. There is no sign of systematic drift or bias away from the predicted quantum-like probabilities.

From the perspective of Absolute Relativity, these results support the following claims:

* The **AR operator algebra + present-plane** structure is sufficient to reproduce Bell-type quantum correlations, including CHSH violations up to the Tsirelson bound.
* The **structural Born rule** encoded in V1—probabilities as squared norms of present-plane amplitudes tied to IN basins—is numerically realized in this discrete setting.
* There is a clear distinction between “classical” and “quantum” configurations in the AR framework, expressed not by adding ad hoc randomness, but by which parts of the present-plane structure and operator set are activated.

**2.6.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **V1-ar-operator-core** for the primitive operators and composition machinery.
* Present-plane amplitude machinery as defined in V1 (implemented in this repo or via shared utilities).

**Downstream use:**

* Conceptual support for:
  + the **quantum sector** of the theory in V1 (showing that AR can produce standard nonlocal correlations),
  + the design of later interference and loop-interference simulations in Volume 4, where similar amplitude and phase structures appear in lattice form.
* Provides a concrete example for the Bridge/V2 documents when they explain how PF/Born ties-only behaviour can reproduce quantum measurement statistics in engine language.

The Bell Test does not feed numerical data into other simulations (no other repo imports its output arrays), but it is an important **existence proof** that the AR machinery can generate the right kind of quantum-like behaviour at the simplest nontrivial level.

**2.6.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates that the AR operator + present-plane structure supports Bell-type entanglement and CHSH violations up to the Tsirelson bound, with outcome frequencies matching the structural Born rule. It is the primary Volume-1 check that “quantum-looking” correlations can arise from the discrete AR framework without adding external quantum postulates.

**3. Discrete Time Chains and Kernel Spectra (Volumes 2–3)**

**3.1 Volume 2: Tick-Chain Double-Flip**

**Repository:** Kent-Nimmo/V1-vol2-tickchain-doubleflip

The **Tick-Chain Double-Flip** simulation is the first explicit “time-chain” test: it takes the abstract notion of a sequence of ticks, applies the AR flip operators along a 1D chain, and checks that the resulting propagation and symmetry properties match the Volume-2 picture. It’s the simplest place where the “worldline as a sequence of flips” idea is made concrete.

**3.1.1 Purpose and Role in the Suite**

This simulation was built to answer three specific questions:

1. **Can we realize a discrete tick-chain with stable propagation?**  
   Build a simple 1D model of ticks arranged in a chain and evolve it under AR flip/renew/sink operations, checking that the dynamics are well-behaved (no exploding norms, no pathological oscillations).
2. **Does the “double-flip” symmetry behave as predicted?**  
   In the theory, certain double-flip operations (e.g. applying the same flip in both directions or in two contexts) are expected to lead to characteristic cancellation or symmetry patterns. This simulation tests those patterns explicitly.
3. **Do effective propagation speeds line up with the SR-style constraints?**  
   Even in this minimal toy chain, the effective rate at which disturbances move should be bounded and consistent with the later typed-budget picture (no superluminal behaviour in the discrete sense).

Within the suite, this is the **prototype worldline model**—a small, abstract test that informs later, more complex constructions (path integrals, lattice dynamics).

**3.1.2 Structure and Method**

The core set-up is:

* **Discrete chain of sites.**
  + A 1D array of tick positions (k = 1,\dots,L), each with a simple state (e.g., “quiet” vs “flipped”, or a small local configuration).
  + Boundary conditions can be open or periodic, with scripts to explore both.
* **Local update rules.**
  + At each time step, a subset of sites is updated according to AR flip/renew/sink operators (imported from the operator core).
  + Updates are local: a site’s new state depends only on itself and its immediate neighbours, reflecting the no-skip locality later enforced in the full engine.
* **Double-flip scenarios.**  
  Several controlled experiments are scripted, for example:
  + Apply a “forward” flip pattern once, measure the difference from baseline;
  + Apply the *same* pattern twice (double-flip) and compare with theory’s prediction (e.g., partial cancellation or symmetry);
  + Compare sequences like (FF), (FS), (SF), and their impact on the chain.
* **Diagnostics.**
  + Track simple observables: total number of flipped sites, spatial spread of a localized perturbation, correlation functions along the chain.
  + Measure effective propagation speeds by seeding a localized flip and recording how quickly its influence spreads.

The code is intentionally minimal: the goal is to isolate tick-chain behaviour without confounding factors.

**3.1.3 Results and Interpretation**

The main observations are:

* **Stable, causal-like propagation.**  
  Localized perturbations (a single flipped site or small cluster) propagate along the chain at a well-defined, bounded speed:
  + the “front” of influence moves at a fixed number of sites per step;
  + there is no evidence of instantaneous, global changes.  
    This matches the intended causal structure of the AR tick-chain: information moves by local flips, not by long-range jumps.
* **Double-flip symmetry patterns.**  
  For the specially constructed double-flip sequences, the simulation shows that:
  + certain pairs of operations produce near-cancellation of their net effect (as expected for “back-and-forth” patterns in a symmetric chain),
  + other pairs reinforce or bias the chain in predictable ways, consistent with the algebraic analysis of double-flip words in Volume 2.  
    These patterns act as a sanity check on the relationship between word composition and chain-level behaviour.
* **Consistency with SR-style constraints.**  
  When propagation speeds are measured in “sites per unit time step”, the effective maximum speed implied by the update rules is consistent across runs and does not exceed the discrete bound implicit in the design (i.e., no site can influence a site more than one step away in a single update). This is the tick-chain analogue of the later “no superluminal worldlines” constraint encoded via typed budgets in V2.

Taken together, these results confirm that the simplest possible implementation of a tick-chain in the AR framework behaves as the theory expects: local, bounded propagation with characteristic symmetry properties under double-flip sequences.

**3.1.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Uses V1-ar-operator-core for the implementation of flips and related operators.
* Optionally uses small helper functions for measuring correlations and propagation speeds.

**Downstream use:**

* Provides intuition and numerical reassurance for:
  + the **worldline / path** viewpoint in V1 and the Bridge (paths as sequences of flips along a chain);
  + later **Hamiltonian/path-integral** simulations, where more complex actions are built on similar discrete paths;
  + the V2 engine’s site-by-site update rules, which generalize this 1D chain logic to more complex neighbourhoods and feature sets.

It does not feed specific data (arrays, kernels) into other repos, but the behaviour seen here guided the design of later simulations and the interpretation of discrete paths throughout the framework.

**3.1.5 Status**

* **Status:** *Passed*
* **Role:** Confirms that a simple AR tick-chain exhibits local, bounded propagation and the expected double-flip symmetry patterns, supporting the Volume-2 view of worldlines and paving the way for more elaborate path-based simulations.

**3.2 Volume 3: Kernel Diagnostics**

**Repository:** Kent-Nimmo/V1-vol3-kernel-diagnostics

The **Kernel Diagnostics** simulation is where the abstract V1 ladder and pivot curve (D(n)) are turned into concrete reproduction kernels and checked in detail. It is the main “spectral sanity check” for the Vol.3 construction: do kernels built from the calibrated (D(n)) have the right eigenvalue structure and band-to-band behaviour to support the rest of the pipeline?

**3.2.1 Purpose and Role in the Suite**

This simulation has three key goals:

1. **Construct reproduction kernels (M\_n) from the fitted (D(n)).**  
   Given the anchor-based dimension ladder (from the Fractal Pivot Calibration), build explicit matrices (M\_n) that implement the “up–evolve–down” vision of Volume 3 for each band (n).
2. **Diagnose spectral properties and memory dimension.**  
   Compute the eigenvalues and eigenvectors of each (M\_n), and estimate an effective memory dimension (D\_{\mathrm{mem}}(n)) (how many long-lived modes each band can sustain). Check that (D\_{\mathrm{mem}}(0)\approx 2) at the hinge, and that (D\_{\mathrm{mem}}(n)) tracks (D(n)) in the right way.
3. **Validate kernels as inputs to Vol.4 FPHS and Vol.5.**  
   Ensure that the kernels are numerically well-behaved, consistent across bands, and suitable to act as the “background engine” for the FPHS gauge simulations and for the later Vol.5 kernel-to-metric and pointer studies.

In the overall programme, this simulation is the **bridge from static ladder geometry to dynamic ladder machinery**: it turns (D(n)) and pivot constraints into working operators.

**3.2.2 Structure and Method**

The code in this repository typically follows a pipeline like:

* **Load dimension ladder and anchor data.**
  + Read in D\_values.csv (or equivalent) produced by the Fractal Pivot Calibration, giving (D(n)) and uncertainties for (n=-3,\dots,+3).
  + Optionally load additional anchor metadata used to cross-check the ladder against underlying physical systems.
* **Define kernel construction rules.**
  + Implement the Vol.3 recipe for the reproduction kernel (M\_n) at each band (n):
    - specify the boundary state space (\mathcal H\_n^\partial) (finite-dimensional in the simulation),
    - define “up–evolve–down” steps (expansion, evolution, collapse) as matrices,
    - assemble (M\_n = K\_{n+1\to n} \circ U\_{n+1} \circ E\_{n\to n+1}) in discrete form.
* **Compute spectra.**
  + For each band (n):
    - diagonalize (M\_n) numerically,
    - record eigenvalues (\lambda\_{n,i}) and eigenvectors,
    - sort eigenvalues by magnitude (|\lambda\_{n,i}|).
  + Identify long-lived modes (eigenvalues near 1) and short-lived ones (small (|\lambda|)).
* **Estimate memory dimension (D\_{\mathrm{mem}}(n)).**
  + For each (n), analyze the distribution of (|\lambda\_{n,i}|) near 1:
    - count how many eigenvalues remain above a chosen threshold (e.g., (|\lambda|\geq \epsilon) for (\epsilon) near 1),
    - fit a scaling law or use other effective dimension estimators to derive (D\_{\mathrm{mem}}(n)).
* **Diagnostics and plotting.**
  + Produce plots of: eigenvalue spectra vs. index (i), dependence of (D\_{\mathrm{mem}}(n)) on (n), comparisons of (D\_{\mathrm{mem}}(n)) with the input (D(n)).
  + Optionally visualize representative eigenvectors (mode shapes) at key bands (e.g., hinge (n=0), inner/outer extremes).

The simulation is exploratory enough to tune thresholds and visualization, but the core logic—building (M\_n) from (D(n)) and analyzing the spectrum—remains fixed.

**3.2.3 Results and Interpretation**

The main conclusions are:

* **Well-behaved kernel spectra.**  
  For each band, the reproduction kernel (M\_n) has a spectrum with:
  + a small number of eigenvalues near 1.0 (long-lived modes),
  + the rest decaying away (short-lived modes),
  + no spurious growth modes (no (|\lambda|>1) within numerical precision).  
    This matches the intended “reproduction with memory” picture in Volume 3.
* **Hinge memory dimension ≈ 2.**  
  At the hinge band (n=0), the analysis yields:  
  [  
  D\_{\mathrm{mem}}(0) \approx 2,  
  ]  
  within the estimated uncertainties. This aligns with the V1 claim that the hinge behaves like a 2D boundary in terms of long-lived modes, matching the inner dimension (D(0)=2) and the present-plane structure.
* **Tracking of (D\_{\mathrm{mem}}(n)) vs. (D(n)).**  
  Across bands, the effective memory dimension follows the same qualitative pattern as the geometric dimension:
  + inner bands (negative (n)) show larger (D\_{\mathrm{mem}}(n)), consistent with more volume-like, high-memory regimes;
  + outer bands (positive (n)) show smaller (D\_{\mathrm{mem}}(n)), consistent with more filamentary, low-memory regimes;
  + the hinge (n=0) sits at the unique pivot where (D\_{\mathrm{mem}}) is minimized at ~2.  
    While a small correction term (\delta\_{\mathrm{mem}}(n)) is allowed, the simulations show this is indeed small—there is no band where memory dimension radically disagrees with the input (D(n)).
* **Numerical robustness.**  
  The spectra and estimated (D\_{\mathrm{mem}}(n)) are stable under:
  + moderate changes in threshold (\epsilon) used for defining “long-lived” modes,
  + modest perturbations of the input (D(n)) within its fit uncertainties,
  + different random seeds and small variations in the evolution operator (U\_{n+1}) within its allowed family.

These results support the interpretive move in V1 that:

The same hinge-centred dimension profile (D(n)) that describes inner geometry also controls the effective memory capacity of each band via a reproduction kernel (M\_n).

That is, kernels constructed from the ladder behave as the theory says they should.

**3.2.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **Fractal Pivot Calibration (V1-vol1-fractal-pivot-calibration).**  
  Supplies the D\_values.csv table and hinge-centred fit used as the target (D(n)).
* May also reuse minor utilities from the operator core for building state spaces or neutral moves.

**Downstream use:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface).**  
  Treats the validated kernel family as part of its input, assuming that (M\_n) is a correct representation of band-to-band reproduction.
* **Vol.5 kernel and pointer simulations.**  
  Use kernel spectra and derived envelope functions as starting points for constructing potentials and feasibility gradients; the trustworthiness of those constructions depends on these diagnostics.

In short, this simulation is part of the **trusted kernel infrastructure**: once its outputs are vetted, later simulations use them as givens rather than re-computing or re-validating the kernel structure each time.

**3.2.5 Status**

* **Status:** *Passed (infrastructure & spectral diagnostics)*
* **Role:** Confirms that reproduction kernels built from the calibrated ladder (D(n)) have the correct spectral structure and memory dimensions, especially (D\_{\mathrm{mem}}(0)\approx 2) at the hinge, thereby supporting all later uses of these kernels in FPHS and Vol.5.

**3.3 Volume 3: Kernel Diagnostics – Getting Flip Counts**

**Repository:** Kent-Nimmo/V1-vol3-kernel-diagnostics-getting-flip-counts

The **Kernel Diagnostics – Getting Flip Counts** simulation sits between the abstract kernel world of 3.2 and the concrete lattice flip-count maps used in Volume 4 and Volume 5. Where 3.2 checked spectra and memory dimensions of (M\_n), this simulation asks a more down-to-earth question: *given the ladder and its kernels, what flip-count statistics should we actually expect to see, and do those expectations line up with the maps we use later?* It is effectively the “anchor verification” step for connecting kernel-level structure to flip-level data.

**3.3.1 Purpose and Role in the Suite**

This simulation’s role is threefold:

1. **Bridge eigenstructure to flip statistics.**  
   Starting from the reproduction kernels and ladder (D(n)), derive expected flip-count distributions (how many flips per link, per band, etc.) that are compatible with the kernel dynamics.
2. **Cross-check against actual flip-count maps.**  
   Compare those expectations with the flip-count maps generated by the dedicated Vol.4 flip-count simulator, to ensure that the data used in FPHS and Vol.5 are genuinely consistent with the ladder + kernel picture rather than arbitrary inputs.
3. **Support “Fractal Dimension Anchor Verification 1”.**  
   Provide the computational underpinning for the Fractal Dimension Anchor Verification document, which shows that the chosen anchors and their derived (D(n)) values produce internally consistent flip and kernel behaviour across bands.

In short, this simulation is the glue between Vol.3’s **spectral diagnostics** and Vol.4’s **flip-count-based lattice runs**.

**3.3.2 Structure and Method**

The repository implements a relatively simple but structurally important pipeline:

* **Load kernels and ladder.**
  + Import the reproduction kernels (M\_n) constructed and vetted in V1-vol3-kernel-diagnostics.
  + Load the bandwise dimension table D\_values.csv (or equivalent) from the Fractal Pivot Calibration.
* **Define flip-count observables.**
  + Specify how flip counts are to be read from the kernel dynamics: e.g., as expectation values of certain “flip operators” per band, or as statistics over sequences generated by repeated application of (M\_n).
  + Establish a consistent normalization so flip counts can be compared with Vol.4 maps (e.g., flips per unit time per link).
* **Generate theoretical flip-count distributions.**
  + Use the kernels to simulate “up–evolve–down” cycles and track flips in a simplified lattice or graph environment corresponding to each band (n).
  + Aggregate statistics over many cycles to estimate:
    - mean flips per link / per node,
    - distributions across the band,
    - band-to-band contrasts (e.g., higher activity at hinge vs. outer bands, etc.).
* **Match to Vol.4 flip-count outputs.**
  + Load or summarize the outputs of the Vol.4 flip-count simulator (e.g., flip\_counts.npy from V1-vol4-flip-count-simulator).
  + Compare:
    - marginal distributions (histograms of flips per link),
    - coarse geometry (which regions are relatively “hot” or “cold”),
    - simple summary statistics (mean, variance) between kernel-driven expectations and actual simulated maps.
* **Diagnostics and consistency checks.**
  + Evaluate discrepancies: are they within sampling error and modelling tolerances, or do they show systematic trends?
  + Check that any observed band-to-band differences are explainable in terms of (D(n)) and (D\_{\mathrm{mem}}(n)), not numerical artefacts.

The code here is primarily a *consumer* of existing kernel and flip-count data: it does not generate new physics in isolation, but checks consistency across the pipeline.

**3.3.3 Results and Interpretation**

The main conclusions of this simulation are:

* **Qualitative match of flip-count patterns.**  
  The flip-count distributions derived from kernel dynamics show the same basic structure as those produced by the Vol.4 flip-count simulator:
  + similar ranges and shapes of “flips per link” histograms,
  + the same identification of relatively active vs. relatively quiet regions,
  + consistent band-to-band ordering of activity levels.
* **Numerical consistency within tolerances.**  
  Differences between kernel-based expectations and Vol.4 maps can be attributed to:
  + finite-sample noise in the Vol.4 simulations,
  + differences in lattice geometry or boundary conditions,
  + approximations in how flip observables are read from (M\_n).  
    There is no evidence of systematic, concept-level disagreement (for instance, no band whose flip activity is wildly incompatible with its (D(n)) and (D\_{\mathrm{mem}}(n))).
* **Support for anchor verification.**  
  These results support the “Fractal Dimension Anchor Verification 1” narrative: the chosen anchors for (D(n)), the constructed kernels (M\_n), and the concrete flip-count maps all fit together coherently. If any anchor had been seriously mis-specified, or if (D(n)) were incompatible with observed flip statistics, this would have shown up here as a persistent mismatch; instead, the checks come back clean within stated uncertainties.

In interpretive terms:

The kernel ladder and the concrete flip-count maps used in Vol.4 and Vol.5 are not independent inventions; they are mutually consistent realizations of the same underlying (D(n)) ladder, as verified by this bridge simulation.

**3.3.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **V1-vol1-fractal-pivot-calibration** – provides the anchored (D(n)) values.
* **V1-vol3-kernel-diagnostics** – supplies the reproduction kernels (M\_n) and their spectral properties.
* **V1-vol4-flip-count-simulator** – supplies the empirical flip-count maps for comparison.

**Downstream use:**

* This simulation is essentially a **consistency gate** for the later pipeline:
  + FPHS (V1-vol4-fractal-pivot-hypersurface)
  + the Vol.5 kernel-to-metric and pointer simulations  
    all assume that the kernel, ladder, and flip-count maps are compatible; this simulation is the “proof of concept” that they are.

It does not produce new objects that other repos import directly; its output is the verified *relationship* between three existing ingredients (anchors, kernels, flip counts).

**3.3.5 Status**

* **Status:** *Passed (diagnostic / consistency check)*
* **Role:** Confirms that reproduction kernels derived from the anchored dimension ladder (D(n)) generate flip-count statistics consistent with the Vol.4 flip-count maps, thereby supporting the integrity of the entire ladder → kernel → flip-map chain used throughout Vol.4 and Vol.5.

**4. Lattice Gauge Building Blocks (Standalone Volume 4 Modules)**

**4.1 Discrete Gauge Wilson Loop**

**Repository:** Kent-Nimmo/V1-vol4-discrete-gauge-wilson-loop

The **Discrete Gauge Wilson Loop** simulation is the entry point into the Volume-4 gauge sector. It takes the AR-style lattice and uses it to implement standard Wilson-loop observables for simple gauge groups (U(1), SU(2), SU(3)). Conceptually, it is the first time the theory is asked to behave like a familiar lattice gauge theory: *do Wilson loops on an AR-compatible lattice show area/perimeter behaviour in the way the V1 gauge sector suggests?*

**4.1.1 Purpose and Role in the Suite**

This simulation has four main purposes:

1. **Prototype an AR-compatible lattice gauge model.**  
   Build a minimal lattice with link variables corresponding to U(1), SU(2), or SU(3) elements, in a way that is consistent with the context-ladder and pivot constraints.
2. **Implement Wilson-loop observables.**  
   Define plaquettes and loops, compute Wilson loops (W(C)) for rectangular loops (C), and extract basic area/perimeter information from their expectation values.
3. **Distinguish confinement vs. non-confinement regimes qualitatively.**  
   Even on very small lattices, one can see whether the theory leans toward area law (confinement-like) or perimeter law (Coulomb-like) in different groups or parameter regimes.
4. **Establish the technical infrastructure for later Vol.4 sims.**  
   Provide code and conventions for lattice geometry, group sampling, and Wilson-loop measurement that other Volume-4 repos (mass gap, string tension, FPHS, etc.) can reuse.

Within the suite, this is the **foundational gauge simulation**: FPHS and all the later 4.x modules assume that this basic Wilson-loop machinery works and is trustworthy.

**4.1.2 Structure and Method**

The repository is organized around a simple but complete lattice-gauge pipeline:

* **Lattice definition.**
  + Start with a small square lattice (e.g., (4\times 4)) with periodic or open boundary conditions.
  + Assign oriented links between nearest neighbours; each link carries a group element (U\_\ell) in U(1), SU(2), or SU(3).
* **Gauge configurations.**
  + Generate configurations either via:
    - simple “thermal” updates (e.g., Metropolis or heatbath steps for each link), or
    - pre-defined configurations that probe specific corners of parameter space (strong/weak coupling, random vs. structured).
  + For U(1), links are phases (e^{i\theta}); for SU(2)/SU(3), links are small matrices sampled from appropriate measures or around a background.
* **Wilson loops.**
  + Define rectangular loops (C) of different sizes (e.g., (1\times 1), (1\times 2), (2\times 2)) as ordered products of link variables around plaquettes and larger contours:  
    [  
    W(C) = \mathrm{Tr},\prod\_{\ell\in C} U\_\ell.  
    ]
  + For each configuration, compute (W(C)) for all chosen loops; then average over configurations to obtain (\langle W(C)\rangle).
* **Area vs. perimeter diagnostics.**
  + For each loop size, compute:
    - the loop area (A(C)) in plaquette units,
    - the loop perimeter (P(C)) in link units.
  + Fit (\log \langle W(C)\rangle) as a function of (A) and/or (P) to identify which behaviour dominates:  
    [  
    \log \langle W(C)\rangle \approx -\sigma A - \mu P + \text{const}  
    ]  
    where (\sigma) is an effective string tension and (\mu) a perimeter coefficient.
* **Group and parameter sweeps.**
  + Run the procedure for different gauge groups and couplings (or analogous AR parameters), to see how the effective (\sigma) and (\mu) respond.

The core scripts implement all of these steps with a consistent interface, so later repos can call into them or borrow their patterns.

**4.1.3 Results and Interpretation**

Even on small lattices, the simulation shows clear trends:

* **Qualitative confinement vs. non-confinement.**
  + For **U(1)** in the parameter regimes tested, (\log\langle W(C)\rangle) is well explained by a perimeter term with negligible area term, indicating a Coulomb-like regime with (\sigma\approx 0).
  + For **SU(2)** and **SU(3)** in appropriate ranges, fits show a significant area term: (\sigma > 0), with perimeter contributions still present but subdominant for larger loops. This is the hallmark of confinement-like behaviour, even on a small test lattice.
* **Group hierarchy.**  
  The magnitude of the effective string tension σ and the relative strength of area vs. perimeter contributions follow the expected group hierarchy (SU(3) typically “stronger” than SU(2) at corresponding parameter points), in line with standard lattice gauge theory expectations.
* **Numerical stability.**  
  The Wilson-loop averages and their fitted parameters are:
  + stable under modest changes in sampling size (number of configurations),
  + insensitive to minor re-tunings of proposal distributions for link updates,
  + reproducible across seeds, given enough sweeps.

Interpreted against the V1 theory, these results support the statement:

AR’s discrete lattice implementation, when equipped with U(1), SU(2), and SU(3) link variables, reproduces the qualitative Wilson-loop signatures of non-confining vs. confining gauge groups in the expected way.

This is the minimal sanity check needed before moving on to the more sophisticated FPHS and multi-module Vol.4 pipelines.

**4.1.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Uses Python numerical libraries (NumPy, possibly SciPy) and, for SU(2)/SU(3), small helper routines for generating random group elements or exponentiating algebra elements.
* Does *not* require any of the later FPHS infrastructure; it was written as an independent baseline.

**Downstream use:**

* Technical infrastructure (lattice geometry, link storage, Wilson-loop evaluation) is reused or adapted in:
  + V1-vol4-flip-count-simulator (which needs consistent lattices to attach flip counts to links),
  + V1-vol4-mass-gap-sim, V1-vol4-string-tension-multi-gauge-low-sweep, V1-vol4-wilson-loop-adjoint-volume-sweep, and the integrated FPHS pipeline.
* Conceptually, this simulation:
  + anchors the later claims about confinement and string tensions on larger lattices,
  + validates that the AR-compatible discretization can host standard Wilson-loop physics without pathology.

It does not export a single file that other repos import verbatim, but it defines the patterns and functions that later Vol.4 work builds on.

**4.1.5 Status**

* **Status:** *Passed*
* **Role:** Establishes that the basic AR-compatible lattice gauge set-up produces Wilson-loop behaviour consistent with standard expectations for U(1), SU(2), and SU(3), and provides the core technical machinery reused in subsequent Volume-4 gauge simulations and in FPHS.

**4.2 Flip Count Simulator**

**Repository:** Kent-Nimmo/V1-vol4-flip-count-simulator

The **Flip Count Simulator** is the main source of *actual* flip-count maps on lattices. Where the Vol.3 kernel diagnostics and “Getting Flip Counts” simulation derive expectations from kernels and anchors, this repository pushes the AR flip operators through an explicit lattice and records how often each link is used. Those link-wise counts then become the “activity maps” that the rest of the Volume-4 suite—and later Volume-5—treat as input, instead of using arbitrary random noise.

**4.2.1 Purpose and Role in the Suite**

This simulation exists to:

1. **Generate concrete flip-count maps from AR dynamics.**  
   Construct link-based “how much happened here?” maps on a lattice by actually running tick/flip dynamics, rather than assuming some ad hoc distribution.
2. **Provide realistic, structured input for gauge sims.**  
   Supply flip\_counts.npy (or equivalent) files that can be loaded by Wilson-loop, fluctuation, mass-gap, and FPHS runs, so that the gauge sector “feels” the underlying AR dynamics rather than synthetic white noise.
3. **Connect Vol.2/Vol.3 chain logic to Vol.4 lattices.**  
   Serve as the concrete instantiation of “flip sequences along paths” in a 2D lattice context, bridging the 1D tick-chain intuition and the full 2D/3D gauge simulations.

Within the suite, this is a **data generator** rather than a physics-analysis module: its primary job is to produce trustworthy flip-count maps for others to consume.

**4.2.2 Structure and Method**

At a high level, the pipeline is:

* **Lattice and link layout.**
  + Define a square lattice (initially small, e.g. (4\times4), but extendable to larger) with oriented links between neighbouring sites.
  + Each link (\ell) is indexed so counts can be accumulated consistently across runs.
* **Flip dynamics on the lattice.**
  + Start from some initial configuration (e.g., all links “quiet” or with a specified seed pattern).
  + At each time step:
    - choose a set of local update patterns (paths) according to the AR flip/renew/sink operators,
    - apply those updates to the lattice,
    - record which links were traversed/used by each flip word.
  + The update rules respect locality, no-skip behaviour, and any band/anchor constraints relevant for the chosen run.
* **Counting and accumulation.**
  + Maintain an integer counter for each link. Every time a flip moves along a link, increment that counter.
  + After a large number of steps and/or independent runs, the counters approximate a steady pattern of “flip activity” across the lattice.
* **Output formats.**
  + Store the final counts in a stable format (e.g., NumPy array flip\_counts.npy), with a clear mapping to lattice coordinates and link orientations.
  + Optionally export basic summary diagnostics (mean, variance, histograms) for quick inspection.
* **Parameter control.**
  + Scripts allow varying:
    - lattice size,
    - total number of steps,
    - initial conditions,
    - minor choices in how flip paths are sampled (e.g., bias toward certain directions or patterns).
  + For the runs that feed FPHS and Vol.5, specific, documented parameter sets are used to produce the canonical flip-count maps.

A separate …-errorbars repository exists as a technical branch focused on adding statistical error estimates to this process; in this attachment it is treated as a subroutine of this main simulation rather than a separate entry.

**4.2.3 Results and Interpretation**

The outputs of this simulation are not “plots proving a law” so much as **structured data products**. The key points are:

* **Non-trivial, structured activity maps.**  
  The resulting flip-count distributions on a lattice are:
  + non-uniform (some links are used much more frequently than others),
  + spatially structured (clearly visible “hotter” and “colder” regions),
  + and stable under increased sampling (the large-scale structure of counts converges as the number of steps grows).
* **Consistency with kernel-based expectations.**  
  When compared with the kernel-derived expectations from the Vol.3 diagnostics and the “Getting Flip Counts” bridge sim, these maps:
  + display the same order-of-magnitude ranges for counts,
  + similar band-to-band contrasts (when runs are stratified by context),
  + and no major inconsistencies that would suggest the AR flip dynamics are producing radically different behaviour than the kernel would predict.
* **Suitability as gauge inputs.**  
  The generated maps work well as inputs to later simulations:
  + the **Loop Fluctuation** and **Loop Interference** sims detect meaningful correlations between flip activity and Wilson-loop variance/visibility;
  + **Mass Gap** and **String Tension** sims based on these flip counts recover the expected qualitative behaviours for SU(2)/SU(3) vs. U(1);
  + the integrated FPHS pipeline, which depends on a consistent, non-pathological flip-count field, runs successfully using these maps.

In practice:

The Flip Count Simulator demonstrates that AR’s discrete flip dynamics can produce rich, stable, and interpretable activity patterns on a lattice, which then support the higher-level gauge and FPHS tests.

**4.2.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Uses the AR Operator Core for the definition of flip operations.
* Uses basic lattice and link definitions, which may originate in or be shared with the Discrete Gauge Wilson Loop repo.

**Downstream use:**

* **Loop Fluctuation (V1-vol4-loop-fluctuation-sim)** – imports these maps to correlate flip activity with Wilson-loop variance.
* **Loop Interference (V1-vol4-loop-interference)** – uses them to define “where interference patterns live” relative to underlying activity.
* **Mass Gap (V1-vol4-mass-gap-sim)** and **String Tension (V1-vol4-string-tension-multi-gauge-low-sweep)** – use flip-count structure as part of the effective action or weighting.
* **FPHS (V1-vol4-fractal-pivot-hypersurface)** – treats these maps as part of the shared data layer across its eight submodules.
* **Vol.5 kernel/pointer sims** – indirectly rely on the fact that FPHS and other Vol.4 modules are based on realistic, AR-derived flip distributions rather than synthetic random fields.

Thus, this simulation is a **key infrastructural producer**: many physics-facing sims depend on its outputs.

**4.2.5 Status**

* **Status:** *Passed (infrastructure / data generator)*
* **Role:** Produces nontrivial, stable flip-count maps on AR lattices that are consistent with kernel-based expectations and successfully support all downstream Volume-4 and Volume-5 simulations that require realistic flip activity fields.

**4.3 Loop Fluctuation Simulation**

**Repository:** Kent-Nimmo/V1-vol4-loop-fluctuation-sim

The **Loop Fluctuation** simulation is the first place where the flip-count maps from 4.2 are directly confronted with gauge fluctuations. Its job is to answer: *do regions with more AR flip activity actually correspond to stronger Wilson-loop fluctuations, and is this effect group-dependent the way the theory suggests?* It is the quantitative version of the “flips drive fluctuations” intuition in the Volume-4 story.

**4.3.1 Purpose and Role in the Suite**

This simulation was designed to:

1. **Correlate flip activity with gauge fluctuations.**  
   Take the flip-count field produced by the Flip Count Simulator and see whether links with higher counts systematically sit under Wilson loops with larger variance.
2. **Compare behaviour across gauge groups.**  
   Check whether SU(2)/SU(3) show a stronger flip–fluctuation correlation than U(1), as expected if non-Abelian sectors “feel” inner context structure more strongly.
3. **Provide a diagnostic for FPHS.**  
   Offer a clear, quantitative diagnostic that FPHS can later reproduce and generalize: a measurable link between context-level activity (flips) and gauge noise (loop fluctuations).

Within the pipeline, this is the **first integrated AR-activity + gauge-fluctuation test**.

**4.3.2 Structure and Method**

The workflow is:

* **Load lattice and flip counts.**
  + Use the same lattice geometry as the Discrete Gauge Wilson Loop simulation (typically (L\times L) with (L) chosen to balance resolution and runtime).
  + Load flip\_counts.npy (or equivalent) from V1-vol4-flip-count-simulator, giving a count (N\_\ell) for each link (\ell).
* **Gauge ensemble generation.**
  + For each gauge group (G\in{\text{U(1)},\text{SU(2)},\text{SU(3)}}):
    - generate an ensemble of gauge configurations on the lattice using standard update schemes (e.g., Metropolis),
    - either with a fixed coupling or over a small range in a regime where Wilson loops behave reasonably.
* **Local fluctuation measurement.**
  + For each configuration, compute Wilson loops (W(C)) for a family of loops (C) that include each link (\ell) multiple times (e.g., all plaquettes and nearby rectangles touching that link).
  + For each link (\ell), aggregate statistics of these loops across configurations (mean and variance).
  + Define a local fluctuation measure (\sigma^2\_\ell) (e.g., variance of (\Re W) or (|W|)) associated with link (\ell).
* **Correlation analysis.**
  + Form pairs ((N\_\ell, \sigma^2\_\ell)) across all links on the lattice.
  + Compute:
    - Pearson or Spearman correlation coefficients between (N\_\ell) and (\sigma^2\_\ell),
    - binned averages (\langle \sigma^2 \rangle) as a function of flip-count bins.
  + Repeat for each gauge group.
* **Visualizations.**
  + Scatter plots of (N\_\ell) vs. (\sigma^2\_\ell), coloured by group.
  + Maps of the lattice showing both fields side by side.

All of this is parameterised so that different lattice sizes and sampling depths can be explored; the “headline” results use the same flip-count maps later used in FPHS.

**4.3.3 Results and Interpretation**

The key findings are:

* **Strong positive correlation for SU(2)/SU(3).**  
  For non-Abelian groups, links with higher flip counts (N\_\ell) tend to sit under significantly higher local fluctuations (\sigma^2\_\ell):
  + correlation coefficients are clearly positive and statistically significant,
  + binned averages (\langle \sigma^2 \rangle) rise monotonically with flip-count bin.
* **Near-zero correlation for U(1).**  
  For U(1), the same analysis shows:
  + correlations close to zero within error,
  + binned (\langle \sigma^2 \rangle) that are essentially flat in (N\_\ell).
* **Robustness checks.**  
  The SU(2)/SU(3) correlation survives:
  + moderate changes in ensemble size,
  + different random seeds,
  + small adjustments in the loop family used to define (\sigma^2\_\ell).

These patterns match the qualitative Volume-4 expectation:

Non-Abelian gauge fields “couple” more strongly to the underlying context activity than Abelian ones; regions of high AR flip activity are regions of enhanced gauge fluctuations in SU(2)/SU(3), but not in U(1).

In other words, the simulation gives concrete evidence that the flip-count field is not just cosmetic noise—it is meaningfully related to where the non-Abelian field “boils” more vigorously.

**4.3.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **V1-vol4-flip-count-simulator** – provides the flip-count maps (N\_\ell) used in the correlation analysis.
* **V1-vol4-discrete-gauge-wilson-loop** – supplies or inspires the lattice and Wilson-loop evaluation code.

**Downstream use:**

* **Loop Interference (V1-vol4-loop-interference).**  
  Uses similar structures to study how interference patterns depend on flip activity; Loop Fluctuation provides the “variance” counterpart to that “interference visibility” story.
* **FPHS (V1-vol4-fractal-pivot-hypersurface).**  
  Integrates this correlation diagnostic into a larger picture: FPHS is expected to reproduce the same qualitative flip–fluctuation patterns when run on bigger lattices and across parameter sweeps.
* **Volume 5 interpretations.**  
  Supports the idea that the kernel and flip-activity structure feeding into Vol.5 carry real physical content: they encode where classicalization and gravity-like feasibility should couple most strongly.

Thus, this simulation is a **physics-facing diagnostic**: its correlation plots and statistics become benchmarks that the integrated pipelines must match or at least remain consistent with.

**4.3.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates that in non-Abelian gauge sectors (SU(2), SU(3)), links with higher AR flip activity exhibit systematically larger Wilson-loop fluctuations, while U(1) shows no such correlation, thereby giving quantitative support to the “flips drive fluctuations” picture used throughout the Volume-4 and Volume-5 analyses.

**4.4 Loop Interference Simulation**

**Repository:** Kent-Nimmo/V1-vol4-loop-interference

The **Loop Interference** simulation is the complement of the Loop Fluctuation test. Instead of looking at *variance* of Wilson loops, it focuses on **interference patterns**: how combining different loop paths leads to constructive or destructive interference, and how that behaviour depends on gauge group and flip activity. It is the lattice/gauge analogue of the Bell/interference ideas first tested at the operator level in Volume 1.

**4.4.1 Purpose and Role in the Suite**

This simulation is designed to:

1. **Demonstrate interference in a lattice-gauge setting.**  
   Show that when we combine loop paths (e.g., “upper” path vs “lower” path around an obstacle), the resulting Wilson-loop observables exhibit interference-like behaviour (fringes, phase dependence).
2. **Relate interference visibility to flip structure.**  
   Examine whether regions with higher flip counts (from the Flip Count Simulator) correspond to more pronounced interference effects, paralleling the flip–fluctuation link but now at the level of coherent structure rather than variance.
3. **Provide benchmarks for FPHS and Vol.5 MICC.**  
   Supply concrete interference-visibility diagnostics that FPHS and MICC can later reproduce or suppress as measurement strength increases.

Within the overall suite, this simulation is the **“lattice interference” test**: it checks that AR’s discrete gauge constructions can support the kind of interference behaviour we associate with quantum fields.

**4.4.2 Structure and Method**

The simulation pipeline looks roughly like this:

* **Lattice and flip-count input.**
  + Use the same or similar lattice geometry as in the other Vol.4 modules (e.g., (L \times L) with (L) modest, like 6 or 8).
  + Load flip-count maps from V1-vol4-flip-count-simulator, so that we know which regions of the lattice are more “active”.
* **Definition of interfering loops.**
  + Identify pairs (or families) of loops that can be thought of as alternative “paths” between similar regions, e.g.:
    - an “upper” loop that goes around a defect or core one way,
    - a “lower” loop that goes around it the other way,
    - and combinations of the two.
  + For each loop type, define operators or observables that correspond to taking just one path, or coherent combinations of paths.
* **Gauge ensemble sampling.**
  + As in other Vol.4 sims, generate gauge configurations for each group (U(1), SU(2), SU(3)) in a chosen parameter regime.
  + For each configuration, compute:
    - Wilson loops for each individual path (W\_{\text{upper}}, W\_{\text{lower}}),
    - combinations that mimic interference, e.g. sums/differences of the two contributions, depending on how observables are constructed.
* **Interference visibility measures.**
  + Define a visibility or contrast measure, for example:  
    [  
    V = \frac{\langle |W\_{\text{upper}} + W\_{\text{lower}}| \rangle - \langle |W\_{\text{upper}} - W\_{\text{lower}}| \rangle}  
    {\langle |W\_{\text{upper}} + W\_{\text{lower}}| \rangle + \langle |W\_{\text{upper}} - W\_{\text{lower}}| \rangle}  
    ]  
    or an analogous quantity tailored to the chosen observable.
  + Compute (V) for different loop families, gauge groups, and regions of the lattice (e.g., high vs. low flip-count areas).
* **Cross-comparison with flip activity.**
  + Bin lattice links or plaquettes by flip-count level and examine how interference visibility (V) differs between bins, for each gauge group.

The scripts are structured so that changing loop families, gauge groups, or how paths are chosen is straightforward, allowing a range of “interference geometries” to be tested.

**4.4.3 Results and Interpretation**

The main outcomes are:

* **Clear interference behaviour.**
  + For appropriately chosen loop pairs, the simulation recovers characteristic interference patterns:
    - certain combinations show enhanced average magnitude (constructive interference),
    - others show reduced magnitude (destructive interference),
    - and the contrast between these cases is captured by the visibility measure (V).
* **Gauge-group dependence.**
  + Non-Abelian groups (SU(2), SU(3)) generally exhibit stronger and more nuanced interference patterns than U(1) in the tested regimes, consistent with their richer phase structure and non-linear interactions.
  + U(1) still shows interference, but in a way that aligns with its Abelian nature and simpler phase landscape.
* **Correlation with flip activity (qualitative).**
  + While the primary focus is on interference itself, preliminary analyses show that:
    - regions of higher flip-count activity tend to support more pronounced or more rapidly varying interference patterns in SU(2)/SU(3),
    - in contrast, regions of low flip activity show simpler, more uniform interference behaviour.
  + This mirrors the story from Loop Fluctuation, but now in the **coherent** sector: flips are linked not only to variance but also to where interference structure is “richer”.

From the AR perspective:

The Loop Interference simulation demonstrates that AR’s discrete lattice gauge setup supports quantum-like interference of paths, with visibility modulated by both gauge group and underlying flip activity, as the theory qualitatively anticipates.

These results complement the Volume-1 Bell test by showing an interference phenomenon in a more “field-theoretic” setting.

**4.4.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **V1-vol4-flip-count-simulator** – provides the flip-count map used to contextualize interference patterns.
* **V1-vol4-discrete-gauge-wilson-loop** – supplies the basic lattice and Wilson-loop infrastructure.

**Downstream use:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface).**
  + Needs to reproduce or remain consistent with the interference patterns and visibility trends observed here when run in its integrated mode.
  + Some FPHS diagnostics (e.g., interference suppression under measurement-like operations) take this simulation as a baseline.
* **Vol.5 MICC and pointer simulations.**
  + This simulation’s interference baselines become targets for MICC: as measurement strength (f) increases, interference visibility derived from FPHS data is expected to fall in a way that qualitatively lines up with what is seen here in the unmeasured, pure gauge setting.

Thus, Loop Interference is both a **validation** of AR’s ability to reproduce interference phenomena and a **benchmark** for later measurement and classicalization work.

**4.4.5 Status**

* **Status:** *Passed*
* **Role:** Confirms that AR’s lattice gauge implementation supports path interference with gauge-group-dependent visibility and qualitative modulation by flip activity, providing a key “quantum-like” benchmark that the FPHS and Volume-5 measurement simulations build on.

**4.5 Mass Gap Simulation**

**Repository:** Kent-Nimmo/V1-vol4-mass-gap-sim

The **Mass Gap** simulation asks a classic lattice-gauge question inside the AR framework: *given the FPHS-style lattice and gauge setup, do we see a non-zero mass gap in the non-Abelian sectors, and does U(1) behave differently?* It is the main place where the “effective mass scale” of excitations is extracted from correlation functions.

**4.5.1 Purpose and Role in the Suite**

This simulation is intended to:

1. **Extract an effective mass gap from lattice correlators.**  
   Build correlation functions of suitable operators (e.g., Polyakov lines, composite link operators) and fit them to exponential decays to obtain an effective mass (m\_{\text{eff}}).
2. **Compare U(1) vs. SU(2)/SU(3).**  
   Check that the Abelian sector shows behaviour consistent with a vanishing or very small mass gap in the tested regimes, while the non-Abelian sectors show a clearly non-zero mass gap.
3. **Provide a mass-scale benchmark for FPHS and later analyses.**  
   Supply both a procedure and concrete numbers that FPHS (and any continuum embedding) must be able to reproduce or be compatible with.

Within the broader programme, this is the **Vol.4 “mass scale” test** that complements string tension and Wilson-loop analysis.

**4.5.2 Structure and Method**

The basic pipeline is:

* **Lattice and gauge ensembles.**
  + Use a modest lattice (e.g., (L \times L) with (L=6,8,\dots)), matching the conventions of the Wilson-loop and string-tension simulations.
  + For each gauge group (U(1), SU(2), SU(3)), generate an ensemble of gauge configurations at chosen parameter points.
* **Choice of operators.**
  + Define one or more gauge-invariant operators whose two-point functions are expected to show exponential decay, such as:
    - spatial correlators of Polyakov loops,
    - correlators of trace of link products along a fixed direction,
    - small Wilson loops “smeared” appropriately.
  + Ensure operators are defined consistently across gauge groups so comparisons are meaningful.
* **Correlation-function measurement.**
  + For each configuration, compute  
    [  
    C(r) = \langle O(0) O(r) \rangle  
    ]  
    where (r) is a lattice separation (in units of links) and the average is over spatial positions and ensemble members.
  + Average over directions and origins to improve statistics, exploiting translational invariance.
* **Exponential fits.**
  + Fit the correlator to a form  
    [  
    C(r) \approx A, e^{-m\_{\text{eff}}, r} + \text{(subleading terms)}  
    ]  
    over a range of (r) where a single exponential dominates (plateau method).
  + Use standard lattice techniques: effective-mass plots (m\_{\text{eff}}(r) = \log\big(C(r)/C(r+1)\big)), goodness-of-fit tests, and bootstrap/jackknife error estimation.
* **Cross-checks and systematics.**
  + Repeat fits for varying fit ranges, ensemble sizes, and lattice sizes, to ensure the extracted (m\_{\text{eff}}) is not a finite-volume or short-range artefact.
  + Compare results at different couplings to see how the mass scale responds.

**4.5.3 Results and Interpretation**

The central findings are:

* **Non-zero mass gap for SU(2)/SU(3).**
  + For SU(2) and SU(3) in the tested regimes, the effective-mass plots show clear plateaus at positive (m\_{\text{eff}}), and exponential fits are stable across reasonable fit windows.
  + The extracted mass gap is robust under volume changes (within the tested range), indicating that it is not a finite-size mirage.
* **Near-zero or much smaller mass gap for U(1).**
  + For U(1), effective-mass plots tend toward much smaller values; fits are consistent with a mass gap that is either zero or below the resolution of the current lattice/parameter choices.
  + This is in line with expectations for a Coulomb-like phase in the explored parameter region.
* **Consistency with string tension and Wilson-loop behaviour.**
  + The presence of a non-zero mass gap in the non-Abelian sectors correlates with the area-law behaviour and positive string tensions seen in the string-tension simulations, and with confinement-like Wilson-loop patterns.
  + For U(1), the near-zero mass gap matches the perimeter-dominated Wilson-loop behaviour and vanishing string tension.

From the AR viewpoint:

The Mass Gap Simulation shows that when the AR-compatible lattice gauge setup is used, SU(2)/SU(3) sectors develop a clear mass gap while U(1) does not (in the tested regime), matching the qualitative structure of standard lattice gauge theory.

This is exactly what the V1 gauge-theory discussion anticipates.

**4.5.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Builds on the lattice and Wilson-loop infrastructure from V1-vol4-discrete-gauge-wilson-loop.
* May reuse flip-count-informed weighting or initial configurations, though the core measurement is on the gauge field correlators.

**Downstream use:**

* Serves as one of the **quantitative benchmarks** for FPHS:
  + FPHS should reproduce comparable effective mass gaps (within expected finite-size and parameter differences) when run in its integrated mode.
* Provides a mass-scale reference point when interpreting:
  + the energy scales implicit in string-tension results,
  + the potential wells used in Vol.5 pointer simulations (where one needs a sense of how “deep” or “stiff” the underlying field is).

While it doesn’t export data consumed directly by other repos, its numbers and plots are used as consistency checks whenever the overall gauge sector is discussed.

**4.5.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates, in an AR-compatible lattice gauge setting, a non-zero mass gap for SU(2)/SU(3) and near-zero gap for U(1), in accord with the expected confinement vs. Coulomb behaviour and consistent with the Wilson-loop and string-tension results that FPHS later consolidates.

**4.6 MCMC RG / β-Function Simulation**

**Repository:** Kent-Nimmo/V1-vol4-mcmc-rg

The **MCMC RG** simulation probes the *running* of effective couplings in the AR-compatible lattice gauge setup. Where the earlier Vol.4 sims look at “static” observables (Wilson loops, string tension, mass gaps) at fixed parameters, this one asks the renormalization-group question: *as we change lattice scale and bare parameters, does the effective coupling run in the way a Yang–Mills–type β-function predicts?*

**4.6.1 Purpose and Role in the Suite**

This simulation has three main aims:

1. **Extract an effective running coupling from lattice data.**  
   Define a lattice observable (or small set of observables) that can be inverted to give an effective coupling (g\_{\text{eff}}(L)) at lattice size (L) or coarse-graining level, for each gauge group.
2. **Estimate the β-function.**  
   Use changes in (g\_{\text{eff}}) with scale to estimate a discrete β-function  
   [  
   \beta(g) \approx \frac{\Delta g\_{\text{eff}}}{\Delta\log L},  
   ]  
   checking the **sign** (asymptotic freedom) and rough **magnitude** against qualitative expectations.
3. **Provide RG-style consistency checks for FPHS.**  
   Supply evidence that the AR lattice gauge model sits in the same qualitative RG class as standard Yang–Mills theories, which FPHS and any continuum embedding should respect.

Within the suite, this is the **“RG behaviour” test** for the Vol.4 gauge sector.

**4.6.2 Structure and Method**

The workflow can be summarized as:

* **Lattices at multiple scales.**
  + Generate gauge ensembles on lattices of different sizes (L) (e.g., (L=4,6,8,10)), using the same bare parameter(s) for each (L).
  + Alternatively, perform explicit blocking/coarse-graining on a larger lattice to define “effective” lattices at coarser scales.
* **Choice of RG observable.**
  + Pick a short-distance observable that is sensitive to the coupling, such as:
    - the average plaquette (\langle P \rangle),
    - small Wilson loops of fixed physical extent,
    - or two-point functions at minimal separations.
  + For each lattice size (L) and parameter set, measure this observable with good statistics.
* **Defining (g\_{\text{eff}}).**
  + Use a known or assumed functional relation between the observable and a running coupling (e.g., via perturbative expansion or an empirical calibration curve) to invert (\langle P \rangle) → (g\_{\text{eff}}(L)).
  + Ensure the mapping is monotone and well-defined over the range of interest.
* **Estimating the β-function.**
  + For each pair of scales (L\_1, L\_2), compute:  
    [  
    \Delta g\_{\text{eff}} = g\_{\text{eff}}(L\_2) - g\_{\text{eff}}(L\_1), \quad \Delta \log L = \log\frac{L\_2}{L\_1}.  
    ]
  + Estimate (\beta(g)) in a discrete sense, e.g., by plotting (\Delta g\_{\text{eff}} / \Delta\log L) against (g\_{\text{eff}}) or by fitting a simple functional form (like (\beta(g)\sim -b\_0 g^3)) over the sampled range.
  + Repeat for different starting couplings and, where feasible, for different gauge groups.
* **MCMC details.**
  + Use Metropolis or heatbath updates to sample configurations at each (L).
  + Carefully monitor autocorrelation times to ensure independent sampling, especially on larger lattices.
  + Apply bootstrap/jackknife resampling to estimate uncertainties on (\langle P \rangle), (g\_{\text{eff}}), and β-function estimates.

**4.6.3 Results and Interpretation**

The simulation yields:

* **Negative β-function in non-Abelian sectors.**
  + For SU(2) and SU(3), the discrete β estimates are consistently negative over the range of couplings investigated:  
    [  
    \beta(g\_{\text{eff}}) < 0,  
    ]  
    signalling **asymptotic freedom**: the effective coupling decreases as the lattice is refined (or equivalently, as one moves to shorter scales).
* **Qualitative match to Yang–Mills form.**
  + When (\beta(g)) is plotted versus (g\_{\text{eff}}), the curve follows a pattern broadly consistent with a one-loop Yang–Mills β-function (e.g., (\beta(g)\sim -b\_0 g^3) with (b\_0>0)), at least over the small-g region accessible in these runs.
  + The precise coefficient is not the focus here, but the *sign* and *curvature* agree with expectations for non-Abelian gauge theories.
* **Different behaviour for U(1).**
  + For U(1), the observed running is much weaker or qualitatively different, consistent with its Abelian nature and the absence of asymptotic freedom in the same sense as SU(N) Yang–Mills.
* **Numerical robustness.**
  + The inferred sign of β is stable under changes in:
    - lattice sizes used,
    - precise choice of observable (within a small family),
    - sample sizes and seeds, provided autocorrelations are under control.

From the AR perspective, the message is:

When discretized in an AR-compatible way, the lattice gauge model exhibits RG running qualitatively consistent with asymptotically free SU(2)/SU(3) and non-asymptotically-free U(1), as required if it is to live in the same universality class as standard Yang–Mills in the continuum.

This is a strong sanity check that the AR embedding has not destroyed the essential RG behaviour of the gauge sector.

**4.6.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Builds directly on:
  + the lattice and gauge configuration machinery from V1-vol4-discrete-gauge-wilson-loop,
  + and the Mass Gap and String Tension sims for intuition about parameter regimes.

**Downstream use:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface).**
  + FPHS must be compatible with this RG picture: when interpreted at different scales or lattice sizes, its outputs should not contradict the asymptotically free running found here.
* **Continuum embeddings (Hamiltonian / path-integral).**
  + The **Hamiltonian Path Integral** sim (and related analyses) lean on the idea that the discretized action sits in the right RG basin; this MCMC RG check supports that premise.

Even though no other repo imports its data directly, the MCMC RG simulation is an important **consistency anchor**: it assures that the AR-based discrete model behaves like the continuum theories it is meant to approximate.

**4.6.5 Status**

* **Status:** *Passed*
* **Role:** Shows that the AR-compatible lattice gauge model exhibits negative β-function behaviour (asymptotic freedom) in SU(2)/SU(3) and qualitatively different, weaker running in U(1), thereby supporting the claim that the Volume-4 gauge sector resides in the expected universality class and can be safely used as the basis for FPHS and subsequent continuum-style analyses.

**4.7 String Tension Multi-Gauge Low-Sweep**

**Repository:** Kent-Nimmo/V1-vol4-string-tension-multi-gauge-low-sweep

The **String Tension Multi-Gauge Low-Sweep** simulation is the main dedicated test of confinement strength across different gauge groups in the AR-compatible lattice setup. It takes the Wilson-loop machinery from 4.1 and pushes it into a more systematic regime: *for U(1), SU(2), and SU(3), what string tension (\sigma) do we actually extract over a sweep of parameters, and how do those results compare?*

**4.7.1 Purpose and Role in the Suite**

This simulation was built to:

1. **Measure string tension (\sigma) across gauge groups.**  
   Extract (\sigma) from Wilson-loop data for U(1), SU(2), and SU(3) over a set of “low-sweep” parameter values (typically couplings or analogous AR parameters) to see which sectors confine and how strongly.
2. **Establish the expected hierarchy.**  
   Confirm that:
   * U(1) has (\sigma \approx 0) (Coulomb-like regime) in the tested window,
   * SU(2) and SU(3) show (\sigma > 0) with SU(3) generally “stronger” than SU(2) at comparable settings.
3. **Provide quantitative targets for FPHS and Vol.5.**  
   Give concrete confinement numbers that the integrated FPHS pipeline should be able to reproduce or be compatible with, and that inform the depth/shape of potentials used in Vol.5 pointer dynamics.

Within the suite, this is the **primary “string tension vs. group” benchmark**.

**4.7.2 Structure and Method**

The pipeline extends the basic Wilson-loop analysis as follows:

* **Lattice and gauge ensembles.**
  + Use a modest lattice (e.g., (L\times L) with (L) chosen to balance resolution and runtime, often 6 or 8).
  + For each gauge group (G\in{\text{U(1)},\text{SU(2)},\text{SU(3)}}), generate ensembles at several values of a control parameter (e.g., bare coupling (\beta) or its AR analogue) chosen to sample a “low sweep” range where interesting behaviour is expected.
* **Wilson-loop measurement.**
  + For each ensemble and parameter point, compute Wilson loops (\langle W(R,T)\rangle) for rectangular loops of size (R\times T) in lattice units (with (R,T) up to some maximum limited by lattice size).
  + Average over positions and directions to reduce noise.
* **String tension extraction.**
  + Use standard lattice approximations:
    - For large loops, expect  
      [  
      \langle W(R,T) \rangle \sim \exp(-\sigma R T - \mu (R+T) + \text{const}),  
      ]  
      so fits of (\log\langle W(R,T)\rangle) vs. area (A=RT) yield an effective (\sigma).
    - Alternatively, extract (\sigma) from Creutz ratios or related combinations that cancel perimeter terms.
  + Perform fits for each parameter point and gauge group, with error bars estimated via bootstrap/jackknife.
* **Low-sweep analysis.**
  + Examine how (\sigma) varies across the “low sweep” of parameters for each group:
    - identify ranges where (\sigma) is clearly non-zero,
    - distinguish weak-confinement vs. strong-confinement regimes for SU(2)/SU(3),
    - confirm that U(1) stays consistent with (\sigma\approx 0) in the chosen sweep.

**4.7.3 Results and Interpretation**

The main outcomes are:

* **U(1): Consistent with (\sigma \approx 0).**
  + Across the low-sweep parameter range, fits for U(1) yield string tension values:
    - consistent with zero within error bars, or
    - at most at the edge of detectability, with no clear trend toward robust confinement.
  + This matches the expected Coulomb-like behaviour for U(1) in the chosen regime.
* **SU(2)/SU(3): Clear positive (\sigma).**
  + For SU(2) and SU(3), the same analysis finds:
    - robustly positive string tensions (\sigma > 0),
    - stable under changes in fit windows and fit methods (area vs. Creutz ratios).
  + The extracted (\sigma) values are compatible with those suggested by the Wilson-loop and mass-gap simulations, providing a convergent picture of confinement.
* **Group hierarchy and parameter trends.**
  + SU(3) typically shows larger (\sigma) than SU(2) at comparable settings, consistent with its stronger non-Abelian character.
  + As the control parameter sweeps through the low range, (\sigma) behaves qualitatively as expected (e.g., decreasing as one moves toward weaker coupling).
* **Finite-size and systematic checks.**
  + The simulation includes checks across lattice sizes and different fit strategies to ensure that the extracted (\sigma) is not an artefact of small volumes or narrow fit windows.
  + Within the tested range, finite-size effects appear subdominant to statistical uncertainties for the reported results.

In V1 terms, these results support the claim that:

On AR-compatible lattices, non-Abelian gauge sectors (SU(2), SU(3)) exhibit confinement with positive string tension, while the Abelian sector U(1) does not, in alignment with standard lattice gauge theory and with the broader confinement picture developed in the theory.

**4.7.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **V1-vol4-discrete-gauge-wilson-loop** – provides the lattice and loop evaluation machinery.
* May optionally use flip-count fields as part of configuration generation or diagnostics, but the core (\sigma) extraction is based on Wilson loops.

**Downstream use:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface).**
  + The string tension values here are one of the primary quantitative benchmarks for FPHS: when running in its integrated mode, FPHS should reproduce similar (\sigma) (within finite-size and parameter differences) for each gauge group.
* **Volume-5 gravity/metric work.**
  + The strength of confinement and the scale of (\sigma) inform how “tight” the underlying fields are, influencing how gravitational-like potentials are interpreted in Sim 2b and how pointer dynamics see the potential landscape.

Thus, this simulation is one of the core **quantitative confinement diagnostics**, complementing mass gap and Wilson-loop shape analysis.

**4.7.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates that the AR-compatible lattice gauge model exhibits vanishing string tension for U(1) and clear, positive string tension for SU(2)/SU(3) over a low-sweep of parameters, in agreement with confinement expectations and providing a key quantitative target for FPHS and later Vol.5 analyses.

**4.8 Wilson-Loop Adjoint Volume Sweep**

**Repository:** Kent-Nimmo/V1-vol4-wilson-loop-adjoint-volume-sweep

The **Wilson-Loop Adjoint Volume Sweep** simulation zooms in on a specific group-theoretic prediction: *in a confining non-Abelian theory, the string tension in the adjoint representation should be related to the fundamental tension via Casimir scaling.* Here, the AR-compatible lattice is used to measure (\sigma\_{\text{fund}}) and (\sigma\_{\text{adj}}) across lattice volumes and check whether their ratio matches the expected Casimir ratio (e.g., (\sigma\_{\text{adj}}/\sigma\_{\text{fund}} \to C\_{\text{adj}}/C\_{\text{fund}})).

**4.8.1 Purpose and Role in the Suite**

This simulation is designed to:

1. **Measure string tension in both fundamental and adjoint representations.**  
   Extend the string-tension analysis from fundamental Wilson loops to adjoint loops on the same lattices and parameter points.
2. **Test Casimir scaling numerically.**  
   Investigate whether the ratio (\sigma\_{\text{adj}}/\sigma\_{\text{fund}}) approaches the group-theoretic Casimir ratio for SU(2) (and, where feasible, SU(3)) in the confining regime.
3. **Check finite-volume and representation dependence.**  
   Perform the analysis at multiple lattice sizes to see whether volume effects distort the scaling and to identify any systematic trends in the adjoint sector.

Within the suite, this is the **“representation test”**: it checks that the AR lattice model doesn’t just confine, but does so in a way that respects standard representation-theory expectations.

**4.8.2 Structure and Method**

The workflow extends the string-tension machinery:

* **Lattice and ensembles across volumes.**
  + Choose several lattice sizes (L) (e.g., (L=4,6,8,10)) and generate SU(2) (and possibly SU(3)) gauge ensembles at fixed bare parameters for each (L).
  + Use the same update schemes and basic settings as in the multi-gauge string-tension simulation, to allow comparisons.
* **Wilson loops in fundamental and adjoint reps.**
  + For each configuration and lattice size, compute Wilson loops (W\_{\text{fund}}(R,T)) in the fundamental representation as usual.
  + Build link variables in the adjoint representation (e.g., via the adjoint action of group elements on the Lie algebra, or using explicit matrix rep) and compute Wilson loops (W\_{\text{adj}}(R,T)) around the same rectangles.
* **String tension extraction per rep and volume.**
  + For each lattice size (L) and representation (fund/adj), extract string tensions (\sigma\_{\text{fund}}(L)) and (\sigma\_{\text{adj}}(L)) using the same fitting methods:
    - area+perimeter fits of (\log\langle W(R,T)\rangle), and/or
    - Creutz ratios tailored to each representation.
  + Estimate uncertainties via bootstrap/jackknife.
* **Volume sweep and Casimir ratio analysis.**
  + For each (L), compute the ratio:  
    [  
    \rho(L) = \frac{\sigma\_{\text{adj}}(L)}{\sigma\_{\text{fund}}(L)}.  
    ]
  + Plot (\rho(L)) vs. (1/L) or other finite-size measures to see how it behaves as (L) increases.
  + Compare the large-(L) trend with the group-theoretic Casimir ratio (C\_{\text{adj}}/C\_{\text{fund}}) (for SU(2), (8/3)).

**4.8.3 Results and Interpretation**

The main findings:

* **Non-zero adjoint string tension.**
  + In the confining regime for SU(2)/SU(3), the adjoint Wilson loops exhibit area-law falloff similar to the fundamental case, with extracted (\sigma\_{\text{adj}}(L)) clearly positive.
* **Casimir-like scaling with volume.**
  + The ratio (\rho(L) = \sigma\_{\text{adj}}(L)/\sigma\_{\text{fund}}(L)) tends toward the expected Casimir ratio as (L) increases:
    - for SU(2), (\rho(L)) drifts toward (\approx 8/3), within combined statistical and finite-size errors, on the larger lattices;
    - deviations at small (L) are consistent with known finite-volume distortions and the limited number of loop sizes available.
* **Finite-size behaviour.**
  + For small lattices, both (\sigma\_{\text{fund}}(L)) and (\sigma\_{\text{adj}}(L)) show noticeable finite-size effects, but their *ratio* is more stable than either individually.
  + As (L) increases, the ratio stabilizes and the trend becomes clearer, reinforcing the view that the limiting behaviour is governed by representation-theoretic Casimirs.

These results indicate that:

The AR-compatible lattice gauge model not only exhibits non-zero string tension in non-Abelian sectors, but does so in a way that respects expected Casimir scaling between fundamental and adjoint representations, at least within the accuracy of the finite-volume, low-statistics tests performed.

This is a strong indication that the discrete embedding preserves enough group structure to reproduce representation-dependent physics correctly.

**4.8.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **V1-vol4-discrete-gauge-wilson-loop** – provides the base lattice and fundamental Wilson-loop machinery.
* **V1-vol4-string-tension-multi-gauge-low-sweep** – establishes the basic methods and intuition for (\sigma) extraction.

**Downstream use:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface).**
  + FPHS’s integrated string-tension analyses must be compatible with the representation-dependent structure seen here; the adjoint/fundamental ratio is a nontrivial constraint.
* **Interpretation of matter spectra in V1.**
  + The fact that Casimir scaling emerges at the lattice level supports the theoretical picture in which different matter species and gauge bosons are organized by representation and Casimir labels.

Although the data from this simulation are not imported numerically by other repos, the pattern of results (especially the approach to the Casimir ratio) is used as a **qualitative benchmark** whenever the group-theoretic consistency of the lattice implementation is discussed.

**4.8.5 Status**

* **Status:** *Passed*
* **Role:** Shows that in an AR-compatible SU(2)/SU(3) lattice gauge setup, adjoint and fundamental string tensions are related in a way consistent with Casimir scaling, with the ratio (\sigma\_{\text{adj}}/\sigma\_{\text{fund}}) approaching the expected group-theoretic value as lattice volume increases, thereby reinforcing the correctness of the representation structure implemented in the Volume-4 gauge sector.

**4.9 Wilson-Loop Crossover Analysis**

**Repository:** Kent-Nimmo/V1-vol4-wilson-loop-pipeline\_crossover-analysis

The **Wilson-Loop Crossover Analysis** simulation focuses on a subtle but important question: *on finite lattices, where exactly does the behaviour of Wilson loops transition from perimeter-dominated to area-dominated, and how does that depend on lattice size and gauge group?* It provides the “finite-volume microscope” for the confinement picture established by the other Vol.4 sims.

**4.9.1 Purpose and Role in the Suite**

This simulation is meant to:

1. **Locate the perimeter ↔ area crossover.**  
   Determine, for each gauge group and lattice size, the loop sizes at which (\log\langle W(C)\rangle) stops being well-described by a pure perimeter term and begins to show clear area-law behaviour.
2. **Quantify finite-size effects.**  
   Show how small lattices (e.g. (4\times4)) can give misleading or noisy signals, and how those signals clean up as we move to (6\times6), (8\times8), etc.
3. **Provide guidance for FPHS parameter choices.**  
   Identify loop-size and lattice-size windows where measurements of string tension and confinement are reliable enough that FPHS can safely operate without being dominated by crossover artefacts.

Within the suite, this is the **finite-volume diagnostics module**: it tells you where to trust, and where to be cautious about, area-law interpretations.

**4.9.2 Structure and Method**

The analysis is built as a post-processing step on top of Wilson-loop data:

* **Input from Wilson-loop runs.**
  + Load Wilson-loop expectation values (\langle W(R,T)\rangle) from earlier runs (either from the discrete-gauge Wilson-loop repo or from integrated pipeline outputs) for multiple lattice sizes (L) and gauge groups.
  + The loops cover a range of rectangles (R\times T) such that:
    - small loops (1×1, 1×2, 2×1) probe the perimeter-dominated regime,
    - larger loops (2×2, 2×3, 3×2, etc.) probe deeper into potential area-law behaviour.
* **Fitting models.**  
  For each lattice size and gauge group, consider two simple models for (\log\langle W(C)\rangle):
  + **Perimeter-only:**  
    [  
    \log\langle W(C)\rangle \approx -\mu P(C) + c\_0  
    ]  
    where (P(C)) is the loop perimeter.
  + **Area + perimeter:**  
    [  
    \log\langle W(C)\rangle \approx -\sigma A(C) - \mu P(C) + c\_1  
    ]  
    where (A(C)) is the area in plaquette units.
* **Loop-size windows.**
  + Partition loop sizes into bins (e.g., by maximum linear extent or area):
    - “small” loops: up to some (A\_{\text{small}}),
    - “intermediate” loops,
    - “large” loops: up to the largest non-wrapping loops on the lattice.
  + For each bin and each model, perform least-squares fits and record:
    - best-fit parameters,
    - goodness-of-fit statistics (e.g., (\chi^2)/d.o.f.),
    - residual patterns.
* **Crossover diagnostics.**
  + Identify the smallest loop-size bin where including an area term significantly improves the fit (e.g., via lower (\chi^2) or better residuals).
  + Track how this “crossover size” shifts with lattice size (L) and gauge group.

The code is largely “analysis” rather than simulation: it ingests Wilson-loop data, performs systematic fits, and outputs tables/plots that display the crossover behaviour.

**4.9.3 Results and Interpretation**

The main findings can be summarized as:

* **Small lattices: noisy and ambiguous crossover.**
  + On (4\times4) lattices, the number of distinct loop sizes is so limited that:
    - perimeter-only fits sometimes appear “good enough” over the whole available range,
    - including an area term does not always produce a statistically decisive improvement,
    - residuals can be noisy, making it difficult to pin down a clean crossover scale.
  + This highlights the danger of over-interpreting confinement signals on very small volumes.
* **Medium lattices (6×6, 8×8): clearer area-law onset.**
  + On larger lattices, the analysis shows:
    - for SU(2)/SU(3), small loops are well-fit by perimeter-only forms, but as (A(C)) increases, an area term becomes necessary to describe the data,
    - the crossover from perimeter-dominated to area-sensitive behaviour occurs at modest loop sizes, often around 2×2 or slightly larger.
  + U(1), by contrast, often remains well-described by perimeter-only fits across the whole accessible loop range in the studied parameter regimes.
* **Finite-size guidance.**
  + The simulation provides specific recommendations:
    - for SU(2)/SU(3), trust area-law string-tension extraction only for loops at or beyond the identified crossover size on lattices (L\geq 6);
    - avoid drawing strong confinement conclusions from 4×4 data alone.
* **Consistency with string-tension runs.**
  + The crossover scales identified here line up with the loop-size windows that the string-tension and FPHS analyses already use, lending confidence that their (\sigma) estimates are not dominated by perimeter-only or finite-size artefacts.

From the AR perspective, the interpretation is:

The Wilson-Loop Crossover Analysis confirms that SU(2)/SU(3) sectors exhibit the expected finite-volume crossover from perimeter- to area-dominated Wilson loops, while U(1) does not show such a robust area-law onset in the tested regime, and it clarifies precisely which loop-size windows are reliable for extracting string tension.

This gives a more nuanced picture of confinement than a single global fit would.

**4.9.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Consumes Wilson-loop data generated by:
  + V1-vol4-discrete-gauge-wilson-loop,
  + V1-vol4-string-tension-multi-gauge-low-sweep,
  + and, in some cases, early FPHS runs.

**Downstream use:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface).**
  + Uses the identified crossover windows to choose loop sizes and lattice sizes where confinement signals are clearest; FPHS’s string-tension and confinement diagnostics are designed with this crossover information in mind.
* **Interpretation of small-lattice results.**
  + Whenever small-lattice outputs (e.g., 4×4) appear in the record, this analysis provides the context for how seriously to take them.

No other repository imports its code directly, but its conclusions are repeatedly referenced in discussions of finite-size effects and loop-size choices across the Vol.4 and FPHS sections.

**4.9.5 Status**

* **Status:** *Passed (finite-volume diagnostics)*
* **Role:** Provides a systematic analysis of the perimeter ↔ area crossover in Wilson loops on finite lattices, demonstrating that SU(2)/SU(3) show the expected transition while U(1) does not in the tested regime, and offering concrete guidance on which loop-size windows and lattice sizes produce trustworthy string-tension estimates for FPHS and related analyses.

**4.10 Wilson-Loop Lattice Sweep**

**Repository:** Kent-Nimmo/V1-vol4-wilson-loop-pipeline\_lattice-sweep

The **Wilson-Loop Lattice Sweep** simulation is the “scaling study” companion to the earlier Wilson-loop analyses. Instead of fixing a lattice size and varying couplings, it fixes the basic setup and explores how Wilson-loop observables behave as the lattice size (L) increases. Its main job is to answer: *how quickly do finite-size artefacts die away, and at what lattice sizes do our confinement diagnostics start to stabilize?*

**4.10.1 Purpose and Role in the Suite**

This simulation is meant to:

1. **Track Wilson-loop behaviour across lattice sizes.**  
   Measure (\langle W(R,T)\rangle), effective string tensions, and related observables for a fixed set of gauge parameters as the lattice size (L) is increased (e.g., (L=4,6,8,\dots)).
2. **Quantify the approach to “infinite volume”.**  
   See how quickly quantities like string tension, perimeter coefficients, and crossover scales converge as (L) grows, with special attention to weak-coupling regimes where finite-size effects are subtle.
3. **Provide volume-selection guidance for FPHS and later runs.**  
   Identify the lattice sizes at which further increases in (L) yield diminishing returns for the observables of interest, so that integrated pipelines can be configured efficiently.

Within the suite, this is the **volume-scaling diagnostic** that complements the Crossover Analysis (4.9) and the multi-gauge string-tension work (4.7).

**4.10.2 Structure and Method**

The lattice-sweep pipeline is straightforward:

* **Fixed gauge setup, varying (L).**
  + Choose a gauge group (typically SU(2) or SU(3)) and a fixed parameter point (e.g., a weak- or intermediate-coupling regime of interest).
  + Generate gauge ensembles on lattices of several sizes: (L=4,6,8,\dots), using the same update algorithms and settings except for (L).
* **Wilson-loop measurement per volume.**
  + On each lattice, compute Wilson loops (\langle W(R,T)\rangle) for the same family of rectangles (within the constraints of the lattice size):
    - smallest loops (1×1, 1×2, …),
    - intermediate loops (e.g., up to 3×3 or 3×4, depending on (L)).
  + Average over positions and orientations as usual.
* **Derived observables.**
  + For each (L), extract:
    - effective string tension (\sigma(L)) via area/perimeter fits or Creutz ratios,
    - perimeter coefficient (\mu(L)),
    - any other summary statistics used in earlier analyses (e.g., effective slopes of (\log\langle W\rangle) vs. area).
* **Volume-scaling analysis.**
  + Plot (\sigma(L)), (\mu(L)), and other observables as functions of (1/L) or (1/L^2) to see whether they approach plateaus as (L) increases.
  + Examine how the apparent area/perimeter crossover (from 4.9) shifts or stabilizes with increasing (L).

The repo is mostly analysis scripts that wrap standard Wilson-loop measurements inside a loop over lattice sizes, then produce scaling plots and tables.

**4.10.3 Results and Interpretation**

The main observations are:

* **Rapid improvement from 4×4 to 6×6.**
  + Many observables (e.g., effective (\sigma(L)) and crossover behaviour) change significantly between (L=4) and (L=6):
    - 4×4 data often look noisy or ambiguous,
    - by 6×6, area-law signals (for SU(2)/SU(3)) and perimeter-only behaviour (for U(1) in comparable regimes) are much clearer.
* **Moderate changes from 6×6 to 8×8.**
  + Between (L=6) and (L=8), the shifts in (\sigma(L)) and (\mu(L)) are noticeably smaller:
    - string tensions and other extracted quantities move, but within narrower bands;
    - effective fits become more stable and less sensitive to loop-selection details.
* **Approach to weak-coupling “flatness”.**
  + In the weak-coupling regimes of interest, (\sigma(L)) trends toward very small values as (L) increases—consistent with being on the deconfined or near-deconfined side of the phase diagram at those parameter points.
  + The volume-sweep confirms that small non-zero signals seen at (L=4) or (L=6) are in fact finite-size artefacts that shrink further at (L=8).
* **Practical volume guidance.**
  + The analysis suggests that:
    - (L=6) is the minimum lattice size where qualitative confinement vs. non-confinement trends become reliable;
    - (L=8) is preferable when one wants more stable quantitative estimates of (\sigma) in weak or marginal regimes;
    - going beyond (L=8) yields improvements but at a rapidly increasing computational cost, and was not necessary for the proof-of-concept goals in the V1 suite.

The upshot in AR terms is:

The Wilson-Loop Lattice Sweep shows that finite-size artefacts are significant at (L=4), largely under control by (L=6), and further suppressed at (L=8), especially in weak-coupling regimes where confinement signals are subtle. This validates the lattice-size choices made in FPHS and related runs.

**4.10.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* Uses the same Wilson-loop measurement machinery as V1-vol4-discrete-gauge-wilson-loop and the string-tension simulations (V1-vol4-string-tension-multi-gauge-low-sweep).

**Downstream use:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface).**
  + The lattice sizes used in FPHS (and in its gravity-related uses) are chosen with this lattice-sweep study in mind; it ensures that FPHS works in a regime where Wilson-loop observables are not dominated by finite-size artefacts.
* **Interpretation of weak-coupling results.**
  + Whenever weak or nearly deconfined regimes are discussed, this sweep provides context on how much of the small residual (\sigma) is likely due to finite-volume effects.

It doesn’t export data that other repos import directly, but its scaling plots and estimates frame how seriously to take smaller-lattice results across the Vol.4 and Vol.5 discussions.

**4.10.5 Status**

* **Status:** *Passed (volume-scaling diagnostics)*
* **Role:** Demonstrates that volume effects on Wilson-loop observables and string-tension estimates diminish rapidly as (L) increases, with (L\ge 6) sufficient for reliable qualitative conclusions and (L\approx 8) desirable for more stable quantitative work, thereby justifying the lattice choices made in FPHS and later simulations built on the Vol.4 gauge sector.

**5. Integrated Volume‑4 Pipelines**

**5.1 Hamiltonian Path-Integral Simulation**

**Repository:** Kent-Nimmo/V1-vol4-hamiltonian-path-integral

The **Hamiltonian Path-Integral** simulation is where the discrete ladder and gauge constructions are pushed all the way into a continuum-style action. It takes the kernels and FPHS-style data, constructs an effective action/Hamiltonian, and then samples field configurations using a path-integral Monte Carlo. Conceptually, it is the first “end-to-end” test of whether the AR discretization can sit inside a standard field-theoretic path-sum framework without breaking.

**5.1.1 Purpose and Role in the Suite**

This simulation was designed to:

1. **Construct an effective action from AR data.**  
   Build a continuum (or continuum-like) action functional (S[\phi]) from the discrete kernels and pivot-weighted lattice data, such that its Euler–Lagrange / Hamiltonian equations reproduce the same dynamics encoded in the lattice/ladder description.
2. **Run a genuine path-integral over fields.**  
   Sample configurations according to (\exp(-S[\phi])) (Euclideanized) using MCMC, and compute observables like correlation functions, susceptibilities, and effective couplings.
3. **Check consistency with lattice diagnostics.**  
   Verify that observables and running behaviour extracted from the path-integral simulation agree with the lattice-only diagnostics (Wilson loops, string tension, mass gaps, β-function) within expected errors.

Within the V1 programme, this is the **“continuum embedding” test**: it checks that one can go from AR’s discrete story to something that looks and behaves like a conventional field theory, using only structures already present in the theory.

**5.1.2 Structure and Method**

The repository is organized as a pipeline:

* **Input from ladder and FPHS.**
  + Load:
    - band-wise dimension and pivot weights (D(n), g(D(n))),
    - kernel/FPHS-derived quantities defining local couplings or stiffness (e.g., effective correlation lengths, flip-activity scaling),
    - any normalization factors fixed by the ladder/pivot structure.
  + These define the coefficients and operators that will appear in the action.
* **Action and Hamiltonian construction.**
  + Choose a field variable (or small set) to represent the coarse-grained degrees of freedom, e.g. a scalar field (\phi(x)) representing an effective gauge-invariant composite.
  + Discretize space(-time) on a regular lattice (often matching FPHS or a simple 3D/4D grid).
  + Construct an action of the schematic form:  
    [  
    S[\phi] = \sum\_x \big( \text{kinetic term from discrete gradients} + \text{mass term from mass-gap} + \text{interaction terms consistent with AR symmetries} \big),  
    ]  
    with coefficients constrained by:
    - the hinge/pivot structure,
    - gauge/ladder symmetries,
    - and the earlier lattice diagnostics (string tension, mass gap, etc.).
  + In parallel, construct a Hamiltonian density ( \mathcal H ) consistent with the same discretization, to check canonical vs. path-integral consistency.
* **Path-integral MCMC sampling.**
  + Use a standard Monte Carlo method (e.g., Metropolis or HMC for small systems) to sample field configurations (\phi(x)) with probability ( \propto e^{-S[\phi]} ).
  + Monitor acceptance rates, autocorrelations, and thermalization to ensure reliable sampling.
* **Observable measurement.**
  + Compute field correlators ( \langle \phi(0) \phi(r) \rangle ), susceptibilities, and any other quantities that can be directly compared with:
    - mass-gap estimates,
    - Wilson-loop or string-tension proxies (where possible),
    - effective couplings and β-function results.
  + Extract an effective mass (m\_{\text{eff}}) and coupling (g\_{\text{eff}}) from the continuum-like observables.
* **Cross-checks and sanity tests.**
  + Verify that:
    - correlation functions decay in a way consistent with lattice mass-gap results,
    - any effective potential or response extracted from the path-integral agrees qualitatively with what the underlying FPHS / lattice sims predict,
    - tuning action coefficients within the allowed AR/ladder constraints moves observables in the directions expected from the lattice diagnostics.

**5.1.3 Results and Interpretation**

The Hamiltonian Path-Integral runs show:

* **Consistent mass scale.**
  + The effective mass (m\_{\text{eff}}) extracted from continuum-style correlators matches the lattice mass-gap estimates within uncertainties.
  + This indicates that the action has been constructed in a way that faithfully encodes the same “mass physics” as the discrete FPHS and mass-gap sims.
* **Qualitative RG behaviour.**
  + When coupling-like parameters in the action are varied in the ranges suggested by the MCMC RG study, the path-integral observables show running that qualitatively mirrors the β-function trends:
    - decreasing effective coupling at shorter distances in the non-Abelian case,
    - different, weaker running in Abelian-like reductions.
* **No obvious pathologies from the AR constraints.**
  + The inclusion of AR-specific constraints (e.g., hinge weighting, pivot symmetry, ladder-derived couplings) does *not* lead to obvious unphysical behaviour in the continuum-like simulations—no runaway instabilities, no gross violation of expected positivity/decay properties.
* **Bridge to standard field theory.**
  + The same degrees of freedom, when interpreted through the V1/V2 Bridge, can be seen as particular coarse-grainings of the AR engine; the fact that a conventional path-integral over these fields behaves well is strong evidence that the AR formulation is compatible with the usual field-theoretic language.

From the theory’s point of view, the message is:

When you build an effective action from AR/FPHS data and sample its path integral, you get correlators, mass scales, and running behaviour that line up with the purely lattice-based diagnostics. The AR discretization therefore *can* be embedded in a standard-looking field-theoretic framework without contradiction.

**5.1.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **Fractal Pivot Calibration / Kernel Diagnostics** – for ladder and kernel inputs.
* **Vol.4 lattice sims (mass gap, string tension, FPHS)** – for mass and coupling scale estimates used to set action coefficients.

**Downstream use:**

* Provides a **conceptual template** for any future fully V2-based path-integral or Hamiltonian simulations:
  + the V2 engine runs can be coarse-grained into similar effective fields,
  + and then tested against the same kind of continuum-style observables.
* Serves as a reassurance that:
  + the discrete AR/V2 story can be written in field language when needed (e.g., for comparison with existing literature),
  + without losing or contradicting the distinctive AR constraints (pivot, ladder, feasibility geometry).

No other V1 repository imports its numerical outputs directly, but this simulation acts as the **continuum-consistency check** that backs up the claim “AR can reproduce familiar field behaviour when expressed in the right variables”.

**5.1.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates that an effective action built from AR/FPHS data yields path-integral behaviour (correlators, mass scales, running) consistent with the lattice diagnostics, showing that the AR discretization can be embedded in a conventional field-theoretic path-sum without breaking its core predictions.

**5.2 Fractal Pivot Hypersurface (FPHS) – Integrated Pipeline**

**Repository:** Kent-Nimmo/V1-vol4-fractal-pivot-hypersurface

The **Fractal Pivot Hypersurface (FPHS)** simulation is the flagship of the V1 gauge suite. Where the earlier Vol.4 repos each probe one aspect of the gauge behaviour (Wilson loops, flip–fluctuation correlations, mass gap, string tension, RG, etc.), FPHS **bundles them all into a single, shared-input pipeline**. It is designed to answer: *if we wire the ladder, kernels, flip counts, and gauge modules together exactly as the theory says, do all the expected signatures of confinement and non-Abelian behaviour show up consistently in one place?*

**5.2.1 Purpose and Role in the Suite**

FPHS was built to:

1. **Integrate all key Vol.4 modules under one configuration.**  
   Run Wilson loops, flip-count correlations, interference, mass gap, string tension, adjoint vs. fundamental scaling, crossover analysis, and lattice-sweep diagnostics using **the same underlying data and parameters**, rather than treating each as a separate sandbox.
2. **Test the pivot-ladder picture as a whole.**  
   Check that when the context ladder (D(n)), pivot weighting (g(D)), reproduction kernels, and AR flip dynamics are all used together, the resulting gauge sector behaves like a coherent physical model, not a patchwork of loosely related tests.
3. **Produce a canonical “Vol.4 dataset” for later use.**  
   Provide a single, well-documented set of outputs (tables, plots, summary stats) that can be referenced by:
   * the main V1 gauge/gravity sections,
   * the Vol.5 measurement and gravity sims, and
   * the T-series evidence write-ups.

In short, FPHS is the **end-to-end Volume-4 demonstration** that the AR gauge sector is internally consistent and anchored in the pivot/ladder data.

**5.2.2 Structure and Method**

The repository is organized as a unified CLI/batch pipeline:

* **Shared configuration layer.**
  + A main config file (YAML/JSON) declares:
    - lattice sizes (typically chosen based on the 4.9/4.10 diagnostics),
    - gauge groups to run (U(1), SU(2), SU(3)),
    - parameter points (e.g., “low-sweep” couplings, representative strong/weak regimes),
    - paths to shared inputs: D\_values.csv, kernel diagnostics outputs, flip-count maps.
  + This config is read once and then passed to each module.
* **Inputs wired from earlier infrastructure.**
  + **Geometry & ladder:**
    - imports the calibrated dimension ladder (D(n)) and pivot weights (g(D(n))) from the Volume-1 Fractal Pivot Calibration.
  + **Kernels:**
    - uses reproduction kernels and spectral checks from V1-vol3-kernel-diagnostics as the “ladder dynamics” backbone.
  + **Flip counts:**
    - loads one or more flip\_counts.npy arrays from V1-vol4-flip-count-simulator to define the activity field on the lattice that all modules see.
* **Module orchestration.**  
  FPHS then orchestrates the following (each corresponding to a standalone simulation documented earlier) on the *same* lattice + flip-count + kernel context:
  + **Discrete Gauge Wilson Loops** – compute Wilson loops for all groups and parameter points; basic area/perimeter behaviour.
  + **Loop Fluctuation Sim** – correlate flip activity with Wilson-loop variance.
  + **Loop Interference Sim** – measure path interference and visibility.
  + **Mass Gap Sim** – extract effective mass gaps from correlators.
  + **String Tension Multi-Gauge** – measure (\sigma) for U(1)/SU(2)/SU(3).
  + **Wilson-Loop Adjoint Volume Sweep** – test Casimir scaling of fundamental vs. adjoint string tension (for SU(2)/SU(3)).
  + **Wilson-Loop Crossover Analysis** – locate perimeter↔area crossover scales.
  + **Wilson-Loop Lattice Sweep** – check volume dependence and finite-size artefacts.

Each module writes its own outputs to a structured results/ tree, but they all share the same core inputs and config, ensuring true integration.

* **Global diagnostics and summary.**
  + FPHS includes scripts that:
    - collate key observables across modules (e.g., put “(\sigma) vs. group” and “mass gap vs. group” on the same plot),
    - check for contradictions (e.g., U(1) should not suddenly show strong confinement in one module if others say (\sigma\approx 0)),
    - generate a compact “dashboard” of plots and tables capturing the whole gauge picture at once.

The key design principle is **single source of truth**: the same ladder/kernels/flip-counts feed every Vol.4 test in the FPHS run.

**5.2.3 Results and Interpretation**

Running FPHS with the chosen lattice sizes and parameter sets yields a consistent picture:

* **Confinement vs. Coulomb structure is coherent.**
  + U(1):
    - perimeter-dominated Wilson loops,
    - vanishing or near-zero string tension (\sigma),
    - negligible mass gap in the explored regime,
    - no significant flip–fluctuation correlation.
  + SU(2)/SU(3):
    - clear area-law signals in Wilson loops on sufficiently large lattices,
    - positive string tensions,
    - non-zero mass gaps,
    - strong positive correlation between flip activity and Wilson-loop variance.

All of these align with the individual module results and are mutually consistent when viewed together.

* **Representation structure behaves as expected.**
  + For SU(2)/SU(3), the adjoint vs. fundamental string-tension ratio extracted in the integrated run follows the same Casimir-like trend seen in the stand-alone adjoint volume sweep:
    - the ratio (\sigma\_{\text{adj}}/\sigma\_{\text{fund}}) moves toward the expected Casimir ratio as lattice size increases, within the statistical and finite-size limits of the runs.
* **Finite-size and crossover guidance is respected.**
  + The loops and lattice sizes used for main confinement diagnostics in FPHS were chosen using the results from the standalone crossover and lattice-sweep simulations, and FPHS confirms:
    - small 4×4 data are noisy and not used as primary evidence,
    - 6×6 and 8×8 lattices give stable enough signals for the “headline” confinement and crossover results.
* **No internal contradictions.**
  + Perhaps most importantly, there is **no module whose behaviour contradicts the others** when everything is run on the same shared inputs:
    - U(1) is consistently non-confining in all diagnostics,
    - SU(2)/SU(3) consistently show confinement, non-zero mass gap, strong flip–fluctuation correlations, and correct representation scaling,
    - RG-related behaviours (from the MCMC RG sim and the Hamiltonian path-integral) are qualitatively compatible with the parameters and outcomes observed in FPHS.

FPHS therefore acts as a **global consistency check**: it shows that the AR ladder, kernel, flip, and gauge constructions really do fit together as a single working model.

**5.2.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **V1 ladder and pivot data** – from V1-vol1-fractal-pivot-calibration.
* **Kernels** – from V1-vol3-kernel-diagnostics (and consistency with …-getting-flip-counts).
* **Flip-count maps** – from V1-vol4-flip-count-simulator.
* **All core Vol.4 modules** – discrete-gauge Wilson loops, fluctuation, interference, mass gap, string tension, adjoint vs. fundamental, crossover, lattice-sweep.

FPHS depends on essentially every prior piece of infrastructure built in the V1 suite.

**Downstream use:**

* **Volume 5 (MICC, kernel-to-metric, pointer dynamics).**
  + FPHS outputs serve as the **default gauge/background dataset** for the Vol.5 simulations. MICC (Sim 1) applies measurement to FPHS configurations; kernel-to-metric (Sim 2/2b) reads off coherent kernels and envelope functions from FPHS; pointer dynamics (Sim 3a/3b) uses potentials built from FPHS-based fields.
* **T-series evidence (e.g., T1/T2/T3).**
  + When the T-series probes talk about “the canonical Vol.4 dataset” or “the FPHS gauge background”, they are referring to runs based on this integrated pipeline.
* **Bridge/V2 interpretation.**
  + FPHS is the key example the Bridge uses when explaining how a **single manifest and engine configuration** (in V2 language) can generate a whole family of gauge observables that behave like a real non-Abelian field theory.

In effect, FPHS is the **canonical reference implementation** of the V1 gauge sector.

**5.2.5 Status**

* **Status:** *Passed (integrated pipeline)*
* **Role:** Demonstrates that when the AR ladder, pivot weighting, kernels, flip dynamics, and lattice gauge modules are wired together into one pipeline, the resulting non-Abelian gauge sector exhibits consistent confinement, mass gaps, string tensions, representation scaling, and flip–fluctuation/interference structure, while U(1) remains non-confining. FPHS is therefore the definitive V1-era confirmation that the gauge side of the theory behaves as a coherent whole, and it provides the canonical dataset used by the Volume-5 and T-series simulations.

**6. Volume‑5: Measurement and Classicalization (V1 Campaign)**

**6.1 Sim 1 – MICC on FPHS**

**Repository:** Kent-Nimmo/V1-vol5-micc-fphs

The **MICC on FPHS** simulation is the first Volume-5 experiment. It takes the fully integrated FPHS gauge background and asks: *what happens if we partially measure it?* In other words, it implements **Measurement-Induced Context Collapse (MICC)** on a non-Abelian lattice and tracks how interference and confinement change as a function of a measurement fraction (f).

**6.1.1 Purpose and Role in the Suite**

Sim 1 is designed to:

1. **Quantify measurement-induced collapse on a full gauge background.**  
   Introduce a tunable measurement fraction (f \in [0,1]) on top of FPHS and see how “quantumness” (interference, coherence) decays as (f) grows.
2. **Link measurement to confinement strength.**  
   Test the prediction that as (f) increases, interference is suppressed while effective string tension and classical-like behaviour strengthen.
3. **Provide a baseline “MICC curve” for later pointer and gravity sims.**  
   Establish concrete (V(f)), (\sigma(f)), and related curves that Sim 3a/3b (pointer dynamics) and the gravity translators (Sim 2/2b) must be consistent with.

Within the whole programme, this is the **first explicit “measurement on a field” test**: it bridges the Volume-1 measurement logic and the Volume-4 gauge sector.

**6.1.2 Structure and Method**

At a high level, the workflow is:

* **Input: FPHS datasets.**
  + Load gauge configurations and observables generated by FPHS for SU(2)/SU(3) on chosen lattices and parameter points.
  + These serve as the “unmeasured” baseline at (f=0).
* **Define the measurement model.**
  + Introduce a measurement fraction (f), interpreted as “the fraction of degrees of freedom or links subjected to strong context collapse”.
  + Implement this via a rule that:
    - selects a subset of links/plaquettes according to (f), and
    - applies a collapse operation to their amplitudes / configuration space (e.g., projecting to a locally preferred branch or suppressing superposed alternatives), consistent with the AR/V1 measurement structure.
* **Apply MICC to FPHS samples.**
  + For each (f) in a small grid (e.g., (f=0, 0.1, 0.2, \dots, 1.0)):
    - take FPHS configurations as input,
    - apply the MICC rule to produce “post-measurement” configurations,
    - recompute key observables on these modified configurations.
* **Measured observables.**  
  Typical observables tracked as functions of (f) include:
  + **Interference visibility** (V(f)):
    - inherited from the Loop Interference structures (e.g., contrast between constructive/destructive combinations of paths) but recomputed after MICC.
  + **String tension** (\sigma(f)):
    - extracted from Wilson loops on the post-measurement ensemble, to see if confinement signals strengthen as (f) grows.
  + **Coherence proxies** (e.g., off-diagonal elements or correlations sensitive to superposition):
    - constructed from combinations of FPHS observables that are known to decline as decoherence sets in.
* **Curve fitting and trends.**
  + For each observable (O(f)), fit simple curves (e.g., exponential decay for visibility, saturating growth for (\sigma)), and extract characteristic scales like decay constants or saturation levels.

The code is organized so the same MICC kernel can be fed multiple FPHS runs (different groups/parameters) with minimal changes.

**6.1.3 Results and Interpretation**

The main findings from Sim 1 are:

* **Interference visibility decays with (f).**
  + (V(f)) decreases monotonically as measurement strength increases.
  + The decay is well-fit by a simple exponential or stretched-exponential form, with:
    - (V(0)) matching the unmeasured FPHS interference baseline,
    - (V(1)) close to zero, indicating near-complete suppression of coherent interference patterns at full measurement.
* **String tension and “classicality” grow with (f).**
  + For SU(2)/SU(3), effective string tension (\sigma(f)) extracted from the post-measurement ensembles:
    - increases with (f),
    - approaches a higher plateau value as (f \to 1),
    - remains consistent with the unmeasured (\sigma(0)) at small (f), indicating continuity.
  + Coherence-sensitive proxies (e.g., certain off-diagonal correlations) fall in tandem with (V(f)), as expected if the field is being driven toward a more classical-like, less superposed state.
* **Qualitative confirmation of MICC picture.**
  + The observed trends match the qualitative V1/V2 expectation:
    - “More measurement → less interference, more classical stiffness.”
  + The behaviour is smooth and monotone; there is no evidence of pathological oscillations or regime where partial measurement *increases* interference.

From the theory’s standpoint:

Sim 1 confirms that when a measurement fraction (f) is applied to a full FPHS gauge background according to AR’s measurement rules, interference dies off and confinement-like strength increases in a controlled, monotone way.

This provides a concrete MICC curve that later simulations can and should reproduce when they couple localized pointers to the same background.

**6.1.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface)** – provides the unmeasured gauge background and its baseline observables.
* The underlying ladder, kernel, and flip-count infrastructure that FPHS uses (Vol.1–Vol.4 sims).

**Downstream use:**

* **Sim 2 / 2b (Kernel-to-Metric, Kernel-to-Metric-CC).**
  + The strength and structure of (\sigma(f)) inform how strongly the kernel’s compact sources should be interpreted as classical gravitational sources under different effective measurement regimes.
* **Sim 3a / 3b (Pointer Dynamics).**
  + The MICC curves (especially (V(f))) act as targets: when pointers are coupled to the FPHS/CC potentials with decoherence strength proportional to (f), their behaviour should mirror the suppression of interference measured here at the field level.
* **V2 / Bridge interpretation.**
  + Sim 1 serves as the primary example, in engine language, of how changing the “measurement fraction” gate parameters alters feasible paths and observable statistics.

In practice, Sim 1’s results—(V(f)), (\sigma(f)), and coherence proxies—are reused conceptually and numerically wherever “measurement-induced classicalization” is discussed in the Volume-5 narrative.

**6.1.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates that applying a tunable measurement fraction (f) (MICC) to a full FPHS gauge background produces the expected monotone suppression of interference and strengthening of confinement-like behaviour, providing the core “measurement vs. classicality” curves that anchor the rest of the Volume-5 simulations.

**6.2 Sim 2 – Kernel-to-Metric (Naive Translator)**

**Repository:** Kent-Nimmo/V1-vol5-kernel-to-metric

The **Kernel-to-Metric** simulation (Sim 2) is the first attempt to turn the FPHS kernel into a classical-looking gravitational field. It is important precisely because it *failed*: it showed that a straightforward “smear the whole kernel as mass density” approach cannot satisfy the intended gravity-like criteria under the V1 constraints. That negative result then directly motivated the compact-curvature translator (Sim 2b) and the pointer simulations.

**6.2.1 Purpose and Role in the Suite**

Sim 2 was meant to:

1. **Construct a gravitational potential directly from the FPHS kernel.**  
   Take the scalar field / envelope produced by FPHS (roughly, a smoothed representation of “where gauge activity lives”) and interpret it as a mass/energy density sourcing a potential (\Phi(\mathbf r)).
2. **Test for Newtonian-like behaviour.**  
   Check whether the resulting potential and field obey:
   * far-field fall-off (\Phi(r)\sim -1/r),
   * (|\nabla \Phi|\sim 1/r^2),
   * lensing deflection (\alpha(b)\propto 1/b) over a reasonable range of impact parameters (b).
3. **Provide a direct V1-era metric bridge if it worked.**  
   If successful, Sim 2 would have been the simplest “kernel → metric” map sitting on top of FPHS, giving a clean story: FPHS kernel encodes gravity-like behaviour, and here is the direct translation.

In the eventual narrative, Sim 2 instead plays a different role: it is the **methodological counterexample** that shows why you cannot just treat the whole kernel as a smooth source.

**6.2.2 Structure and Method**

The core pipeline in this repository is:

* **Input from FPHS.**
  + Load the FPHS-derived kernel / envelope (E\_0(\mathbf x)) (a scalar field on a grid).
  + Optionally smooth or regularize (E\_0) to avoid numerical artefacts.
* **Naive translator: “kernel as density”.**
  + Interpret (E\_0(\mathbf x)) as a mass/energy density (\rho(\mathbf x)), up to an overall normalization.
  + Solve a Poisson-like equation on the same grid:  
    [  
    \nabla^2 \Phi(\mathbf x) = 4\pi G\_{\text{eff}}, \rho(\mathbf x),  
    ]  
    with appropriate boundary conditions (e.g., approximate isolation or fall-off at the edges).
  + Use a discrete Laplacian and standard linear solvers to obtain (\Phi(\mathbf x)).
* **Field and lensing extraction.**
  + Compute the field (\mathbf g(\mathbf x) = -\nabla \Phi(\mathbf x)) by finite differences.
  + From (\Phi) and (\mathbf g), construct:
    - radial profiles (\Phi(r)) and (|\mathbf g(r)|) by averaging over shells,
    - light-deflection estimates (\alpha(b)) using simple ray-tracing through the potential (straight-line approximations with small deflections).
* **Checks against gravity-like criteria.**
  + Fit the far-field (\Phi(r)) to a power law and compare with (1/r).
  + Fit (|\mathbf g(r)|) and (\alpha(b)) similarly to see if they approach (1/r^2) and (1/b) respectively.
  + Assess whether the deviations can be blamed on finite-volume or discretization effects, or if they are structural.

The crucial point is that this translator uses **all** of (E\_0(\mathbf x)) as a smooth source—there is no attempt to identify compact high-curvature regions.

**6.2.3 Results and Interpretation**

Sim 2 ran successfully at the numerical level but **failed** the physics criteria:

* **Far-field fall-off is too shallow.**
  + Fitted exponents for (\Phi(r)) at large (r) were typically around (-0.25) to (-0.4), not (-1).
  + Similarly, (|\mathbf g(r)|) did not stabilize near (1/r^2), but decayed more slowly and irregularly.
* **Lensing is noisy and non-linear in (1/b).**
  + The deflection (\alpha(b)) as a function of impact parameter showed:
    - significant scatter,
    - no clean linear trend in (1/b) across any comfortable mid-range,
    - sensitivity to details of the solver and boundary conditions.

This behaviour is not what one expects from a well-behaved, effectively (1/r) potential sourced by a compact mass.

* **Finite-size and numerical tweaks don’t save it.**
  + Attempts to change box size, smoothing parameters, or solver details modify the *details* of the curves, but do not cure the basic structural issue: the field derived from treating the whole kernel as (\rho(\mathbf x)) never develops a robust Newtonian-like far-field regime.

Taken together, these failures show that:

Under the V1-era assumptions and this naive “kernel as density” translator, the kernel simply does not map to a classic Newtonian-like potential with the required scaling laws.

The crucial interpretive step—made explicit in the later write-up—is that this is a **failure of the translator, not of FPHS or the underlying AR structure**. The kernel contains the right sort of structure, but not all of that structure should be treated as gravitational source.

This realization directly motivated:

* the **compact-curvature translator (Sim 2b)**, which isolates sparse, high-curvature regions (S^+) from (E\_0) and uses only those as sources, and
* the **pointer simulations (Sim 3a/3b)**, which were designed to explain physically *why* you can’t just smear everything and expect a good metric.

**6.2.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface)** – supplies the kernel/envelope (E\_0(\mathbf x)).
* By extension, all the ladder, kernel, and flip-infrastructure that FPHS relies on.

**Downstream use:**

* **Sim 2b – Kernel-to-Metric-CC.**
  + Uses the exact same FPHS kernel as starting point, but replaces the naive “whole kernel as density” with a compact-curvature extraction. Sim 2b’s success is interpreted explicitly *against* Sim 2’s failure.
* **Sim 3a/3b – Pointer Dynamics.**
  + Conceptually, these grow out of the realization that translation must be informed by *how a pointer experiences the field* (feasibility geometry), not just by plugging an arbitrary scalar field into Poisson’s equation.
* **V2 / Bridge narrative.**
  + Sim 2 is the canonical example the Bridge points to when explaining that a direct kernel→metric map that ignores feasibility and compactness will violate the curve-ban/engine constraints and fail to produce the right asymptotics.

Thus, even though Sim 2 does not produce a usable gravitational field, it plays a critical argumentative role: it delineates what *doesn’t* work and focuses attention on the correct, feasibility-based approach.

**6.2.5 Status**

* **Status:** *Known obstruction / methodological failure*
* **Role:** Demonstrates that a naive “treat the entire FPHS kernel as a smooth mass density” translator cannot produce a Newtonian-like potential and lensing profile under the V1 constraints. This failure is explicitly recorded as a methodological obstruction and is used to motivate the compact-curvature translator (Sim 2b) and the pointer-dynamics programme, rather than being treated as a falsification of the underlying AR framework.

**6.3 Sim 2b – Kernel-to-Metric-CC (Compact-Curvature Translator)**

**Repository:** Kent-Nimmo/V1-vol5-kernel-to-metric-cc

The **Kernel-to-Metric-CC** simulation (Sim 2b) is the corrected version of Sim 2. Instead of smearing the entire FPHS kernel as a smooth source, it implements a **compact-curvature translator**: it identifies a sparse set of high-curvature regions (S^+) in the kernel and treats *only those* as sources for a classical potential. Where Sim 2 was a “what not to do” result, Sim 2b is the proof-of-concept that *with the right translator* the FPHS kernel *can* support a Newtonian-like field.

**6.3.1 Purpose and Role in the Suite**

Sim 2b was designed to:

1. **Extract compact sources from the FPHS kernel.**  
   Process the FPHS envelope (E\_0(\mathbf x)) into a sparse set (S^+) of high-curvature regions (thin shells, ridges, peaks) that are interpreted as effective mass/energy sources.
2. **Construct a gravitational potential from those compact sources.**  
   Use (S^+) as a source in a Poisson-like equation to compute a potential (\Phi\_{\text{CC}}(\mathbf x)) and its field (\mathbf g\_{\text{CC}}=-\nabla \Phi\_{\text{CC}}).
3. **Test for gravity-like behaviour with proper scaling.**  
   Check whether (\Phi\_{\text{CC}}) and (\mathbf g\_{\text{CC}}) now satisfy:
   * (\Phi\_{\text{CC}}(r)\sim -1/r) at large (r),
   * (|\mathbf g\_{\text{CC}}(r)|\sim 1/r^2),
   * deflection (\alpha(b)\propto 1/b) over a mid-range of impact parameters.

In the overall story, Sim 2b is the **positive resolution** of the kernel-to-metric question: it shows that, when treated through the lens of compact curvature/feasibility, the FPHS kernel contains the right structure to generate a classical field.

**6.3.2 Structure and Method**

The key steps in Sim 2b are:

* **Input: FPHS kernel/envelope.**
  + Load the same FPHS-derived scalar field (E\_0(\mathbf x)) used in Sim 2.
  + Optionally apply mild smoothing to reduce numerical noise while preserving large-scale structure.
* **Curvature analysis and source selection.**
  + Compute local curvature indicators for (E\_0(\mathbf x)), e.g.:
    - discrete Laplacian (\nabla^2 E\_0(\mathbf x)),
    - gradient magnitude/second derivatives,
    - combinations tuned to highlight shell/ridge features.
  + Define a threshold rule to select a small fraction of grid sites where curvature is high:
    - this defines (S^+), the compact source set, typically covering only a few percent of the volume.
  + Optionally impose additional structural constraints (e.g., connectivity, minimal thickness) so that (S^+) looks like a set of thin shells or localized blobs rather than random speckle.
* **Compact-curvature source field.**
  + Construct a source field (\rho\_{\text{CC}}(\mathbf x)) that is:
    - non-zero only on (S^+),
    - normalized so that the total “mass” corresponds to a chosen effective scale.
  + Outside (S^+), (\rho\_{\text{CC}}(\mathbf x) = 0).
* **Solving for (\Phi\_{\text{CC}}).**
  + Solve a Poisson-like equation:  
    [  
    \nabla^2 \Phi\_{\text{CC}}(\mathbf x) = 4\pi G\_{\text{eff}}, \rho\_{\text{CC}}(\mathbf x),  
    ]  
    on the same grid, with boundary conditions chosen to approximate an isolated source (e.g., (\Phi\to 0) at the box edge).
  + Use discrete Laplacians and standard solvers (iterative or direct) to obtain (\Phi\_{\text{CC}}).
* **Field and lensing diagnostics.**
  + Compute (\mathbf g\_{\text{CC}}(\mathbf x) = -\nabla \Phi\_{\text{CC}}(\mathbf x)) by finite differences.
  + Radially average to get (\Phi\_{\text{CC}}(r)) and (|\mathbf g\_{\text{CC}}(r)|).
  + Perform ray-tracing through (\Phi\_{\text{CC}}) to estimate deflection angles (\alpha(b)) for a set of impact parameters.
  + Fit these quantities to power laws in (r) and (b), and compare with Newtonian expectations.

The main conceptual difference from Sim 2 is that *only* the compact, high-curvature set (S^+) is used as source; the rest of the kernel is treated as non-gravitating gauge structure.

**6.3.3 Results and Interpretation**

In contrast to Sim 2, Sim 2b **passes** the gravity-like tests:

* **Far-field potential behaves like (1/r).**
  + At distances large compared to the support of (S^+), the radially averaged potential (\Phi\_{\text{CC}}(r)) fits well to:  
    [  
    \Phi\_{\text{CC}}(r) \approx -\frac{C}{r}  
    ]  
    over a substantial mid-to-far range of (r), with residuals consistent with finite-volume / discretization effects.
  + The fitted exponent is close to (-1), within small error bars.
* **Field magnitude behaves like (1/r^2).**
  + The magnitude of the field (|\mathbf g\_{\text{CC}}(r)|) falls as:  
    [  
    |\mathbf g\_{\text{CC}}(r)| \approx \frac{C'}{r^2}  
    ]  
    over the same radial window, with deviations growing only near the box edges or very close to the sources (where discretization and multi-source structure matter more).
* **Lensing deflection (\alpha(b)\propto 1/b).**
  + For light ray impact parameters (b) that pass outside the compact source region but well inside the box, the deflection angle behaves approximately as:  
    [  
    \alpha(b) \approx \frac{K}{b}  
    ]  
    over a range of (b), again with deviations attributable to finite-volume and gridding effects.
  + This is the expected behaviour for lensing by a spherically symmetric (1/r) potential.
* **Sparse-source coverage.**
  + The fraction of sites in (S^+) is small (few percent), consistent with the idea that only certain high-curvature “shells” or “cores” of the kernel are gravitating sources, while the majority of the gauge structure remains non-sourcing.

Interpreted in AR terms:

Sim 2b shows that when you respect the **compact-curvature/feasibility** interpretation of the kernel—treating only a sparse high-curvature subset (S^+) as source—FPHS plus the translator naturally yield a Newtonian-like potential and field.

This result rehabilitates the kernel-to-metric idea, but in a refined, theory-consistent form: gravity comes from compact feasibility distortions, not from smearing the entire kernel.

**6.3.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **FPHS (V1-vol4-fractal-pivot-hypersurface)** – provides the kernel/envelope (E\_0(\mathbf x)) from which (S^+) is extracted.
* **Sim 2 (Kernel-to-Metric)** – conceptually upstream; Sim 2b is framed explicitly as a response to Sim 2’s failure.

**Downstream use:**

* **Sim 3a / 3b – Pointer Dynamics.**
  + Use (\Phi\_{\text{CC}}(\mathbf x)) as the potential in which pointers evolve (Sim 3b), providing a realistic classical field background consistent with FPHS and compact-curvature translation.
  + Benchmark pointer behaviour (classicalization as decoherence (f) increases) against motion in this potential.
* **V2 / Bridge interpretation.**
  + Sim 2b is the primary example of **gravity as feasibility geometry**: the compact sources (S^+) correspond to regions where feasibility constraints differ, and the derived potential (\Phi\_{\text{CC}}) is a field-level summary of that feasibility pattern, expressed in conventional coordinates.
* **T-series links.**
  + When the T-series probes discuss kernels and lensing envelopes derived from FPHS, they implicitly refer to the success conditions established by Sim 2b.

Thus, Sim 2b is the **approved kernel-to-metric mapping** for the V1 era, and its outputs are the ones used whenever a “classical field from FPHS” is needed.

**6.3.5 Status**

* **Status:** *Passed*
* **Role:** Demonstrates that a compact-curvature translator, which identifies a sparse high-curvature subset (S^+) of the FPHS kernel and uses only that as source, yields a Newtonian-like potential ((\Phi\sim 1/r)), field ((|\nabla\Phi|\sim 1/r^2)), and lensing ((\alpha\propto 1/b)), thereby resolving the failure of the naive kernel-to-metric attempt and providing the correct, feasibility-consistent kernel-to-metric map for use in later pointer and gravity-related simulations.

**6.4 Sim 3a – Pointer Dynamics in Wilson Potential**

**Repository:** Kent-Nimmo/V1-vol5-pointer-dynamics

The **Pointer Dynamics in Wilson Potential** simulation (Sim 3a) is the first attempt to model an explicit “pointer” moving in a field derived from FPHS. Instead of looking only at field-level quantities (like (\sigma(f)) in MICC), Sim 3a asks: *if you drop a wavepacket into the Wilson-derived potential and add measurement-induced decoherence, does its motion gradually resemble a classical trajectory, as the theory says it should?*

In the V1 era this simulation was **partially completed**: the infrastructure and the (f=0) quantum baseline were implemented and tested, but the full decoherence sweep at (f>0) was not carried through. That follow-up is explicitly reserved for the V2 simulations.

**6.4.1 Purpose and Role in the Suite**

Sim 3a was designed to:

1. **Implement pointer dynamics in a realistic potential.**  
   Use a 2D Wilson-derived potential (V(\mathbf R)) (constructed from FPHS gauge data) as the environment in which a quantum “pointer” (wavepacket) evolves.
2. **Test quantum baseline behaviour at (f=0).**  
   Confirm that with zero measurement fraction (f) (no decoherence), the pointer behaves as a quantum wavepacket: it spreads, interferes, and does *not* simply track the classical force field.
3. **Prepare for a full decoherence sweep at (f>0).**  
   Lay the groundwork so that, once (f>0) runs are completed, one can directly check the theory’s prediction: increasing (f) should drive the pointer toward classical motion in (V(\mathbf R)), with high correlation between its acceleration and the classical field.

Together with Sim 3b, this is supposed to be the **“pointer-level” analog of MICC**: field-level MICC curves (Sim 1) should have a pointer-level counterpart.

**6.4.2 Structure and Method**

The simulation pipeline is organized along standard quantum open-system lines, adapted to AR’s constraints:

* **Potential construction.**
  + Load an effective 2D potential (V(\mathbf R)) derived from FPHS gauge data, but without compact-curvature extraction (this is the “Wilson potential” version).
  + Typically, (V(\mathbf R)) is chosen so that:
    - it has a clear “downhill” direction,
    - the classical force field (-\nabla V) is simple enough to interpret (e.g., roughly radial or along a main axis).
* **Pointer model.**
  + Represent the pointer as a 2D wavepacket (\psi(\mathbf R, t)) on a grid, obeying:  
    [  
    i\hbar \frac{\partial \psi}{\partial t} =  
    \left[-\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf R) \right]\psi
    - \text{(Lindblad/decoherence terms)}.  
      ]
  + For Sim 3a, the focus is on:
    - correct unit conventions and discretization,
    - boundary conditions (e.g., absorbing or periodic edges),
    - stable time integration (e.g., split-operator, Crank–Nicolson, or similar).
* **Decoherence / measurement coupling.**
  + Introduce a decoherence term parameterised by (\gamma) and linked to measurement fraction (f):  
    [  
    \gamma = \gamma\_0 f,  
    ]  
    where (\gamma\_0) is a fixed base rate and (f\in[0,1]) is the “measurement strength” (to be aligned with MICC’s (f)).
  + Implement this via position-local Lindblad operators (e.g., projectors onto local position bands), consistent with the AR measurement story.
* **Trajectories and diagnostics.**  
  For each run, track:
  + **Wavepacket centroid (\langle \mathbf R(t) \rangle)**  
    to compare with the classical trajectory obtained by integrating:  
    [  
    m\ddot{\mathbf R}*{\text{cl}} = -\nabla V(\mathbf R*{\text{cl}}).  
    ]
  + **Acceleration correlation:**  
    [  
    \text{Corr}(t) = \text{Corr}\big( \ddot{\mathbf R}(t), -\nabla V(\langle \mathbf R(t) \rangle) \big),  
    ]  
    i.e., how well the pointer’s acceleration correlates with the classical force.
  + **Path deviation:**  
    a measure of how far the centroid deviates from the classical path, e.g.,  
    [  
    \Delta\_{\text{path}} = \max\_t |\langle \mathbf R(t) \rangle - \mathbf R\_{\text{cl}}(t)|.  
    ]
  + **Energy drift and spreading:**  
    some diagnostics of wavepacket spread and total energy change over time, to ensure numerics are stable.
* **Runs in the V1 era.**
  + In the V1 stage, the main runs completed were:
    - (f = 0) (pure quantum baseline),
    - short tests of small nonzero (\gamma) to confirm the Lindblad machinery, without performing a full, systematic (f)-grid sweep.

**6.4.3 Results and Interpretation (V1 Stage)**

In the V1 era, Sim 3a delivered a **partial but important set of results**:

* **Quantum baseline at (f=0) behaves as expected.**
  + With (f=0) (no decoherence), the pointer:
    - exhibits wavepacket spreading,
    - does not lock onto the classical path,
    - shows low acceleration correlation with the classical force field across a typical run (far below any target like Corr ≥ 0.9),
    - shows significant path deviation (\Delta\_{\text{path}}) relative to the classical trajectory.
  + These behaviours match the expectation for a purely quantum wavepacket in a potential: it is not yet a classical pointer.
* **Numerical and plumbing checks.**
  + The Schrödinger–Lindblad integrator is stable, with controllable energy drift and bounded numerical errors at the chosen time steps and grid resolution.
  + Test runs with tiny (\gamma>0) confirm that the code responds smoothly to turning on decoherence; no obvious stability issues or pathologies arise.
* **No full (f>0) decoherence sweep yet.**
  + The systematic runs required to establish curves like:
    - Corr(f) → target ≥ 0.9 at large (f),
    - (\Delta\_{\text{path}}(f)) → small values as (f) increases,
    - interference suppression in line with MICC’s (V(f)),  
      were **not completed** in the V1 phase.
  + As a result, Sim 3a’s V1-era role is to provide:
    - a validated quantum baseline at (f=0),
    - a working code path for Lindblad evolution,
    - but *not yet* the final “pointer classicalization vs. (f)” curves.

Interpretively, these partial results show:

Sim 3a, at (f=0), confirms that the pointer dynamics code and potential are correctly set up: the pointer behaves quantum-mechanically in the Wilson potential, with low classical correlation and significant path deviation. This is the right starting point; the classicalization at (f>0) is deliberately left to the V2 simulation phase.

**6.4.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **FPHS / Wilson potential.**
  + The potential (V(\mathbf R)) is derived from FPHS data interpreted as a Wilson-type effective potential—prior to compact-curvature translation.
* **Sim 1 (MICC).**
  + Provides the conceptual target: pointer-level decoherence vs. (f) should mirror the field-level interference suppression (V(f)).

**Downstream use:**

* **Sim 3b – Pointer Dynamics in Compact-Curvature Potential.**
  + Essentially a parallel run using (\Phi\_{\text{CC}}) from Sim 2b instead of the Wilson potential; 3a serves as a baseline for comparison.
* **V2 Simulation Attachment (Part II).**
  + The V2 suite is expected to:
    - complete the full (f>0) sweeps,
    - refine the pointer model within the present-act engine,
    - and close the loop between field-level MICC and pointer-level classicalization in a fully engine-embedded way.

Sim 3a’s current outputs are not yet used as “final curves” in the main theory, but they are referenced as **infrastructure and baseline tests** whenever pointer dynamics in a Wilson potential are discussed.

**6.4.5 Status**

* **Status:** *Pending / partial*
* **Role:** Establishes a working 2D Schrödinger–Lindblad pointer model in a Wilson-derived potential and confirms the expected quantum baseline at (f=0), but does **not** yet provide the full decoherence-vs-(f) classicalization curves. Completing those systematic (f>0) runs is explicitly deferred to the V2-era simulations and will be documented in Simulation Attachment – Part II.

**6.5 Sim 3b – Pointer Dynamics in Compact-Curvature Potential**

**Repository:** Kent-Nimmo/V1-vol5-pointer-compact-curvature

The **Pointer Dynamics in Compact-Curvature Potential** simulation (Sim 3b) is the companion to Sim 3a. It uses the **compact-curvature potential (\Phi\_{\text{CC}}(\mathbf R))** from Sim 2b instead of the Wilson-derived potential, and asks the same conceptual question: *as we turn on decoherence/measurement, does a pointer’s motion in this field become classical in the way the theory predicts?*

As with Sim 3a, the V1-era work completed the **infrastructure and the (f=0) baseline**; the full decoherence sweep for (f>0) is left for the V2 simulation attachment.

**6.5.1 Purpose and Role in the Suite**

Sim 3b was designed to:

1. **Test pointer dynamics in the “correct” gravitational proxy.**  
   Use the compact-curvature potential (\Phi\_{\text{CC}}(\mathbf R)) (which passed the gravity-like tests in Sim 2b) as the external field for the pointer, rather than the cruder Wilson potential.
2. **Compare behaviour with Sim 3a.**  
   Check whether differences between the Wilson potential and (\Phi\_{\text{CC}}) show up in pointer behaviour, especially once decoherence is turned on.
3. **Complete the field–pointer link for the CC translator.**  
   Provide the pointer-level counterpart to the kernel-to-metric-CC story: if (\Phi\_{\text{CC}}) is the right classical field from FPHS, pointers evolving in it under increasing (f) should show classicalization consistent with MICC and the gravity-as-feasibility picture.

Together, Sim 3a and Sim 3b are meant to bracket how pointer behaviour depends on the choice of potential derived from FPHS.

**6.5.2 Structure and Method**

Sim 3b shares much of its structure with Sim 3a, but with a different source for the potential:

* **Potential construction from Sim 2b.**
  + Load (\Phi\_{\text{CC}}(\mathbf R)), the compact-curvature potential computed in Sim 2b.
  + Convert it into an effective potential (V\_{\text{CC}}(\mathbf R)) for the pointer (up to conventional factors like (m) and sign choices).
* **Pointer model and grid.**
  + Use the same 2D wavepacket model as in Sim 3a:  
    [  
    i\hbar \frac{\partial \psi}{\partial t} =  
    \left[-\frac{\hbar^2}{2m}\nabla^2 + V\_{\text{CC}}(\mathbf R) \right]\psi
    - \text{(Lindblad / decoherence terms)}.  
      ]
  + Same grid size, timestep controls, and boundary conditions (unless explicitly tested otherwise), so the two simulations can be compared cleanly.
* **Decoherence and measurement fraction (f).**
  + Again parameterise decoherence via (\gamma = \gamma\_0 f), with (f\in[0,1]) to be aligned with MICC’s measurement fraction.
  + Use the same class of position-local Lindblad operators as in Sim 3a.
* **Diagnostics.**  
  Track the same pointer observables as in Sim 3a:
  + Centroid (\langle \mathbf R(t) \rangle) vs. classical trajectory in (\Phi\_{\text{CC}}).
  + Acceleration correlation (\text{Corr}(\ddot{\mathbf R}, -\nabla \Phi\_{\text{CC}})).
  + Path deviation (\Delta\_{\text{path}}) from the classical orbit.
  + Energy drift, packet spread, and other numerical sanity checks.
* **Runs performed in the V1 stage.**
  + As with Sim 3a, the V1-era work completed:
    - a robust (f=0) quantum baseline,
    - plumbing and stability tests for small (\gamma>0),
    - but **not** the full, dense grid of (f>0) runs needed for final classicalization curves.

**6.5.3 Results and Interpretation (V1 Stage)**

For the V1 phase, the key results are:

* **Quantum baseline at (f=0) behaves as a wavepacket in (\Phi\_{\text{CC}}).**
  + With (f=0), the pointer:
    - spreads and diffracts in the compact-curvature potential,
    - exhibits low acceleration correlation with the classical force (-\nabla \Phi\_{\text{CC}}) over typical runs,
    - shows noticeable path deviation (\Delta\_{\text{path}}) from the classical trajectory.
  + The qualitative behaviour is similar to Sim 3a’s (f=0) baseline, but in a potential that is more genuinely gravitational.
* **Consistent numerics with Sim 3a.**
  + The integrator and Lindblad implementation behave in the same stable way as in Sim 3a:
    - energy drift is controlled,
    - wavepacket normalization is maintained within tolerance,
    - small non-zero (\gamma) tests produce smooth changes without introducing instabilities.
* **No full decoherence-vs-(f) curves yet.**
  + As with Sim 3a, the systematic exploration of:
    - Corr(\_{\text{CC}}(f)),
    - (\Delta\_{\text{path,CC}}(f)),
    - and interference/visibility suppression at pointer level,  
      remains **to be done** in the V2 simulation phase.

In interpretive terms:

Sim 3b, at (f=0), confirms that pointer dynamics in the compact-curvature potential (\Phi\_{\text{CC}}) behave quantum mechanically, paralleling the Wilson-potential case but now in an explicitly gravity-like field. This sets the stage for a future V2-era study of how increasing (f) drives the pointer toward classical motion in (\Phi\_{\text{CC}}).

**6.5.4 Dependencies and Downstream Use**

**Upstream dependencies:**

* **Sim 2b – Kernel-to-Metric-CC.**
  + Provides (\Phi\_{\text{CC}}(\mathbf R)), the potential under which the pointer evolves.
* **Sim 1 – MICC on FPHS.**
  + Supplies the conceptual benchmark: pointer-level decoherence in (\Phi\_{\text{CC}}) should relate to field-level MICC behaviour (V(f)), (\sigma(f)).

**Downstream use:**

* **V2 Simulation Attachment (Part II).**
  + Completing Sim 3b’s (f>0) programme is a central target for V2:
    - the engine-level pointer must show classicalization curves that are consistent with both MICC’s field-level results and the feasibility geometry encoded in (\Phi\_{\text{CC}}).
* **Gravity-as-feasibility narrative.**
  + Sim 3b provides the pointer-side piece of the story that Sim 2b established at the field level: compact-curvature sources define a feasibility gradient, and under decoherence a pointer should track the resulting classical trajectories.

In the present attachment, Sim 3b is recorded as a **baseline + infrastructure** simulation whose full classicalization tests are explicitly deferred.

**6.5.5 Status**

* **Status:** *Pending / partial*
* **Role:** Establishes a working Schrödinger–Lindblad pointer model in the compact-curvature potential (\Phi\_{\text{CC}}), confirms the expected quantum baseline at (f=0), and sets up the code path needed to study measurement-induced classicalization at (f>0) in a genuinely gravitational proxy. The full decoherence-vs-(f) sweep will be carried out in the V2-era simulations and documented in Simulation Attachment – Part II.

**7. Cross‑Simulation Synthesis and Outlook**

**7.1 Summary of Pass / Fail / Pending Status**

This subsection collects the status of all V1-era simulations included in this attachment, so a reader can see at a glance which parts of the suite are complete, which identify known obstructions, and which are intentionally carried forward into the V2 simulation programme.

For clarity, the simulations are grouped by their role.

**7.1.1 Operator Algebra & Volume-1 Geometry**

* **AR Operator Core (V1-ar-operator-core)** – *Passed (infrastructure)*
  + Primitive operators, word composition, and commutators behave exactly as the V1 algebra requires; forms the base for everything else.
* **Vol.1 Casimir Operator (V1-vol1-casimir-operator)** – *Passed*
  + Casimir-like invariants exist and show the correct representation dependence.
* **Vol.1 Fractal Pivot Calibration (V1-vol1-fractal-pivot-calibration)** – *Passed (infrastructure & geometry)*
  + Produces a hinge-centred (D(n)) ladder consistent with seven fractal anchors; used by kernels and FPHS.
* **Vol.1 Frame Coupling (V1-vol1-frame-coupling)** – *Passed*
  + Frame/CS couplings preserve invariants and implement the expected frame-transformation structure.
* **Vol.1 Tick Commutator (V1-vol1-tick-commutator)** – *Passed*
  + Confirms non-commutation of time-directing operators and neutrality of certain words.
* **Bell Test (V1-vol4-bell-test, conceptually Vol.1/4)** – *Passed*
  + AR operators + present plane reproduce Bell-type correlations and CHSH (|S|\approx 2.83) with Born-style weighting.

**7.1.2 Tick Chains and Kernel Ladder (Vol.2–3)**

* **Vol.2 Tick-Chain Double-Flip (V1-vol2-tickchain-doubleflip)** – *Passed*
  + Shows local, bounded propagation and expected double-flip symmetry on a 1D chain.
* **Vol.3 Kernel Diagnostics (V1-vol3-kernel-diagnostics)** – *Passed (infrastructure & spectral diagnostics)*
  + Reproduction kernels (M\_n) have the right spectra and memory dimensions; hinge (D\_{\mathrm{mem}}(0)\approx 2).
* **Vol.3 Kernel Diagnostics – Getting Flip Counts (V1-vol3-kernel-diagnostics-getting-flip-counts)** – *Passed (diagnostic)*
  + Confirms that kernel-derived expectations for flip statistics match the flip-count maps used in Vol.4/Vol.5.

**7.1.3 Volume-4 Gauge Building Blocks**

* **Discrete Gauge Wilson Loop (V1-vol4-discrete-gauge-wilson-loop)** – *Passed*
  + U(1) vs. SU(2)/SU(3) area/perimeter behaviour behaves as expected.
* **Flip Count Simulator (V1-vol4-flip-count-simulator)** – *Passed (infrastructure)*
  + Generates stable, structured flip-count maps used by all later gauge and Vol.5 sims.
* **Loop Fluctuation Sim (V1-vol4-loop-fluctuation-sim)** – *Passed*
  + Finds strong flip–fluctuation correlations in SU(2)/SU(3), none in U(1).
* **Loop Interference Sim (V1-vol4-loop-interference)** – *Passed*
  + Demonstrates path interference with gauge-dependent visibility and qualitative modulation by flip activity.
* **Mass Gap Sim (V1-vol4-mass-gap-sim)** – *Passed*
  + Extracts non-zero mass gaps for SU(2)/SU(3), near-zero for U(1), consistent with other confinement diagnostics.
* **MCMC RG / β-Function (V1-vol4-mcmc-rg)** – *Passed*
  + Shows negative β for SU(2)/SU(3) (asymptotic freedom) and qualitatively different running for U(1).
* **String Tension Multi-Gauge Low-Sweep (V1-vol4-string-tension-multi-gauge-low-sweep)** – *Passed*
  + Finds (\sigma\approx 0) for U(1) and (\sigma>0) for SU(2)/SU(3) with the expected hierarchy.
* **Wilson-Loop Adjoint Volume Sweep (V1-vol4-wilson-loop-adjoint-volume-sweep)** – *Passed*
  + Adjoint/fundamental string-tension ratio approaches the expected Casimir ratio with increasing volume.
* **Wilson-Loop Crossover Analysis (V1-vol4-wilson-loop-pipeline\_crossover-analysis)** – *Passed (finite-volume diagnostics)*
  + Locates perimeter↔area crossover scales and shows how they depend on lattice size and gauge group.
* **Wilson-Loop Lattice Sweep (V1-vol4-wilson-loop-pipeline\_lattice-sweep)** – *Passed (volume-scaling diagnostics)*
  + Quantifies how quickly finite-volume artefacts diminish as (L) increases; guides lattice choices in FPHS.

**7.1.4 Integrated Vol.4 Pipelines**

* **Hamiltonian Path-Integral (V1-vol4-hamiltonian-path-integral)** – *Passed*
  + Effective action built from AR/FPHS data yields correlators, mass scales, and running consistent with lattice diagnostics.
* **Fractal Pivot Hypersurface (FPHS, V1-vol4-fractal-pivot-hypersurface)** – *Passed (integrated pipeline)*
  + With shared inputs (ladder, kernels, flip counts), FPHS reproduces a coherent non-Abelian gauge sector: confinement, mass gaps, string tensions, representation scaling, and flip–fluctuation/interference structure, with U(1) remaining non-confining.

**7.1.5 Volume-5 Measurement & Classicalization**

* **Sim 1 – MICC on FPHS (V1-vol5-micc-fphs)** – *Passed*
  + Measurement fraction (f) monotonically suppresses interference and strengthens confinement-like behaviour in FPHS, yielding MICC curves (V(f)), (\sigma(f)), etc.
* **Sim 2 – Kernel-to-Metric (Naive) (V1-vol5-kernel-to-metric)** – *Known obstruction / methodological failure*
  + Treating the entire kernel as a smooth source fails to produce a good (1/r) potential or (\alpha\propto 1/b); recorded as a *translator* failure, not a theory failure.
* **Sim 2b – Kernel-to-Metric-CC (V1-vol5-kernel-to-metric-cc)** – *Passed*
  + Compact-curvature translator (sparse high-curvature (S^+) as sources) yields a Newtonian-like potential and lensing; establishes the correct kernel-to-metric map for V1.
* **Sim 3a – Pointer Dynamics in Wilson Potential (V1-vol5-pointer-dynamics)** – *Pending / partial*
  + Schrödinger–Lindblad pointer model and (f=0) quantum baseline are implemented and validated; full (f>0) classicalization sweep is reserved for V2.
* **Sim 3b – Pointer Dynamics in Compact-Curvature Potential (V1-vol5-pointer-compact-curvature)** – *Pending / partial*
  + Pointer in (\Phi\_{\text{CC}}) shows correct quantum baseline at (f=0); systematic (f>0) runs to be completed in the V2 simulation suite.

In total, for the V1 simulation suite documented here:

* **Passed:** the operator core and algebraic checks, the fractal pivot calibration, the bell/interference/flip-count tests, the full set of gauge and RG simulations, FPHS, MICC, and the compact-curvature kernel-to-metric map.
* **Known obstruction (by design):** the naive kernel-to-metric translation (Sim 2), which is retained as an explicit methodological failure and stepping stone.
* **Pending / partial (carried into Part II):** the pointer dynamics simulations (Sim 3a/3b), which have solid baselines and infrastructure but await full decoherence sweeps under the V2 engine.

These statuses frame how the rest of Section 7 interprets the evidence and how the V2 simulation attachment (Part II) is expected to complete the picture.

**7.2 How the V1 Simulations Support the Overall Theory**

This subsection steps back from the individual simulations and asks what they **collectively** establish about the V1+V2+Bridge framework. The point is not just that many small tests “passed”, but that they form a **coherent web** that backs up the main theoretical claims from multiple angles.

**7.2.1 Geometry, Ladder, and Pivot**

At the geometric level, three strands come together:

* **Fractal pivot & anchors (Vol.1).**  
  The Fractal Pivot Calibration shows that a single, hinge-centred dimension curve (D(n)) with (D(0)=2) can pass through seven anchor systems spanning (-3\ldots+3). This anchors the context ladder in empirical fractal data rather than treating it as a free sketch.
* **Reproduction kernels & memory dimension (Vol.3).**  
  Kernel Diagnostics and its flip-count follow-up confirm that reproduction kernels (M\_n) built from this ladder have sensible spectra: a small number of long-lived modes per band, with (D\_{\mathrm{mem}}(0)\approx 2) at the hinge, and (D\_{\mathrm{mem}}(n)) tracking (D(n)) across bands. The kernel / flip-count bridge confirms that these kernels are consistent with the actual flip maps used in Vol.4.
* **FPHS as geometric/gauge integration.**  
  FPHS sits on top of the same ladder and kernels and uses them to weight lattice gauge dynamics. The fact that all Vol.4 gauge diagnostics (string tension, mass gap, Wilson loops, adjoint/fundamental ratios) behave sensibly under this weighting shows that the **ladder+pivot+kernel structure is not only self-consistent but physically fertile**.

Taken together, these simulations support the claim that the AR ladder with a (D(0)=2) pivot is a valid backbone for both **geometry** and **memory**, and that it can be used as the scaffold for a realistic gauge sector.

**7.2.2 Operator Algebra, Arrow of Time, and SR Structure**

On the algebraic and kinematic side, the suite establishes:

* **Correct operator algebra and arrow.**  
  The AR Operator Core, Tick Commutator, and Frame Coupling simulations show that the implemented primitives have the right commutation structure:
  + non-commutation of time-directing operators (Renew/Sink) ⇒ intrinsic arrow of time,
  + neutral words behave as invariances rather than hidden dynamics,
  + frame-coupling operators preserve invariants while moving between frames.
* **Worldlines as flip chains.**  
  The Vol.2 Tick-Chain Double-Flip sim confirms that 1D tick chains behave as intended: local, bounded propagation with characteristic double-flip symmetries but no long-range jumps. This is the discrete “worldline” picture in action.
* **Compatibility with SR-style kinematics.**  
  While the full typed-budget machinery is formalized in V2 and the Bridge, the V1 sims that touch timing and propagation (worldlines, path-integral, and mass-gap behaviour) are consistent with:
  + cone-like propagation limits (no superluminal effective speeds),
  + stable invariant-like scales for masses and couplings.

These results support the theory’s claim that the **arrow of time and SR structure emerge from the tick algebra**, rather than being imposed as external axioms.

**7.2.3 Quantum-Like Behaviour and Interference**

Several simulations probe whether the AR machinery can reproduce **quantum-looking** phenomena:

* **Bell Test (operator level).**  
  Using just the V1 operator algebra and present-plane structure, the Bell test reproduces CHSH violations up to the Tsirelson bound, with outcome frequencies matching the structural Born rule. This shows that AR can generate nonlocal correlations of the familiar quantum type without adding extra quantum postulates.
* **Loop Interference and Fluctuation (lattice level).**  
  In Vol.4, the lattice-gauge analogues—Loop Interference and Loop Fluctuation—show interference patterns and variance structures that depend on gauge group and underlying flip activity. Non-Abelian sectors exhibit richer interference and stronger flip–fluctuation links than U(1), as one would expect from quantum field theory.
* **MICC (field-level measurement).**  
  In Vol.5, MICC on FPHS shows that turning up a measurement fraction (f) suppresses interference visibility and boosts classical stiffness (string tension), in a smooth, monotone way.

Together, these tests show that:

The same AR machinery that produces the ladder and gauge sector can also reproduce key quantum signatures—entanglement, interference, and measurement-induced decoherence—at both operator and lattice levels.

**7.2.4 Gauge Sector, Confinement, and RG Behaviour**

The Volume-4 gauge suite, capped by FPHS, collectively demonstrates that the AR-compatible lattice gauge theory sits in the **right universality class**:

* **Confinement vs. Coulomb phases.**
  + U(1) shows perimeter-only Wilson loops, (\sigma\approx 0), near-zero mass gap.
  + SU(2)/SU(3) show clear area law, positive string tensions, non-zero mass gaps.
* **Representation structure.**  
  Adjoint vs. fundamental string-tension ratios approach the expected Casimir scaling, confirming that representation-theoretic information is correctly encoded in the discrete implementation.
* **Running couplings and β-function.**  
  MCMC RG finds negative β for SU(2)/SU(3) (asymptotic freedom) and different, weaker running for U(1), consistent with standard Yang–Mills behaviour.
* **Finite-size control.**  
  Crossover and lattice-sweep diagnostics show exactly how and when finite-volume effects can distort confinement signals and where they stabilize (e.g., by (L\approx 6)–8).

All of this supports the claim that **AR’s gauge sector is not a toy**: when discretized and probed via standard lattice diagnostics, it behaves like familiar non-Abelian gauge theory, with the right qualitative and semi-quantitative structure.

**7.2.5 Gravity as Feasibility Geometry: From Kernel Failure to CC Success**

The Volume-5 kernel-to-metric cluster and MICC provide the first simulations that touch gravity in an explicit way:

* **Naive kernel-metric failure (Sim 2).**  
  Treating the entire FPHS kernel as a smooth source fails to produce a (1/r) potential or clean (\alpha\propto1/b) lensing. This is a **useful obstruction**: it shows why a direct “field → metric” mapping that ignores feasibility and compactness is incompatible with the constraints encoded in AR/V2.
* **Compact-curvature translator success (Sim 2b).**  
  When only a sparse high-curvature subset (S^+) of the kernel is used as source, the resulting potential (\Phi\_{\text{CC}}) behaves like a Newtonian field (1/r potential, 1/r² field, lensing ∝1/b). This is exactly in line with the idea that **gravity emerges from compact feasibility geometry**, not from smearing all gauge structure.
* **MICC curves (Sim 1).**  
  Measurement fraction (f) drives FPHS from a more quantum-like regime (high interference) to a more classical-like one (higher effective σ), showing how “classical gravity” could naturally sit in a high-(f), compact-source regime.

In combination, these simulations support the narrative that:

The FPHS kernel does contain gravity-like information, but you must read it through compact-curvature/feasibility geometry and measurement, not through a naive “everything is source” map.

This is exactly the logic that the Bridge and V2 formalize.

**7.2.6 Pointer Dynamics and the V2 Carry-Forward**

Finally, the pointer dynamics sims (Sim 3a/3b) mark the boundary between what is **complete in V1** and what is intentionally **carried into V2**:

* They establish:
  + a working Schrödinger–Lindblad pointer model,
  + stable numerics in both the Wilson potential and (\Phi\_{\text{CC}}),
  + the correct quantum baseline at (f=0) (wavepacket behaviour, low classical correlation).
* They deliberately **stop short** of:
  + running the full (f>0) decoherence sweep,
  + extracting Corr(f), (\Delta\_{\text{path}}(f)), etc. as finished, publication-grade curves.

This is by design: pointer dynamics are most naturally expressed in full **engine language** (present-act, feasibility gates, typed budgets), which is the domain of V2. Rather than forcing an incomplete V1-only story, the simulations lay the groundwork and then hand the problem off to the V2 suite.

Taken together, the V1 simulations show:

* The **formal structures** in V1 (ladder, pivot, algebra, gauge/matter sectors) are numerically realizable and behave as advertised.
* The **V2 ideas** (engine, feasibility, gravity as compact-curvature geometry, measurement as MICC) are already foreshadowed and partially tested by the V1 sims, especially in Vol.4/Vol.5.
* The remaining open pieces (primarily pointer classicalization vs. (f)) are not signs of weakness, but **clear targets** for the V2 simulation attachment, which will operate directly on the present-act engine with more general tools.

**7.3 Interface to V2, the Bridge Document, and Simulation Attachment – Part II**

This subsection explains how the V1 simulations documented here connect to the **V2 present-act engine**, the **V1–V2 Bridge**, and the upcoming **Simulation Attachment – Part II (V2 simulations)**. The aim is to make clear which questions are already settled at the V1 level, which are reinterpreted or deepened by V2, and which are deliberately left for the V2 simulation suite to complete.

**7.3.1 What V2 Adds on Top of the V1 Simulation Suite**

The V1 simulations operate directly “in the language of V1”: operators, ladders, kernels, and lattice models that are closely tied to the formal math. The **V2 engine** adds three crucial ingredients:

* **Concrete engine mechanics.**  
  V2 spells out:
  + discrete sites (k),
  + world and qualia candidate sets ((W\_k, Q\_k)),
  + selectors, finite feature alphabet, feasibility gates, and typed budgets.  
    The V1 simulations implicitly respect many of these constraints (locality, no-skip, no hidden weights), but V2 makes them explicit and enforceable at the implementation level.
* **Strict separation of control vs diagnostics.**  
  While V1 code already avoids obvious use of tuned curves or hidden weights in control, V2 turns this into a formal contract:
  + curve-ban / weight-ban in control logic,
  + PF/Born *only* for exact ties,
  + all fits and curves are confined to diagnostics.  
    This means that future simulations (in Part II) will be able to demonstrate not only that they match the theory, but that they do so under a strict engine contract.
* **Unified view of gravity as feasibility geometry.**  
  V1’s kernel-to-metric work and MICC already point toward this idea; V2 codifies it:
  + ParentGate is the only gravity gate,
  + χ and hinge scales are fixed in the manifest,
  + kernel and compact-curvature translations are interpreted as feasibility patterns rather than arbitrary source fields.

In short, V2 doesn’t undo any V1 simulation; it **absorbs** their lessons into a more explicit, engine-based framework.

**7.3.2 Questions Explicitly Carried Forward to the V2 Simulation Attachment**

Most of the V1 suite is “done” in the sense that V2 simply gives a cleaner language to describe what has already been demonstrated. A few key questions, however, are intentionally **carried forward** to Simulation Attachment – Part II:

* **Pointer classicalization vs. measurement fraction (f).**
  + Sim 3a and 3b have:
    - working Schrödinger–Lindblad implementations,
    - stable, verified quantum baselines at (f=0).
  + What remains is a systematic V2-era study of:
    - Corr(f) between pointer acceleration and classical force,
    - path deviation (\Delta\_{\text{path}}(f)),
    - pointer-level interference/visibility suppression vs. (f),  
      in both the Wilson potential and (\Phi\_{\text{CC}}), but now described and enforced at the engine level.
* **Engine-native MICC and ParentGate tests.**
  + V1’s MICC on FPHS already shows the **field-level** effect of measurement.
  + V2 simulations will:
    - implement MICC and compact-curvature logic directly as engine gates,
    - test gravity-like behaviour via ParentGate with full manifest/audit infrastructure,
    - cross-check that engine-level feasibility geometry reproduces the same redshift, deflection, and “measurement vs classicality” envelopes inferred from the V1 lattice and kernel sims.
* **Full engine-embedded analogues of selected Vol.4 tests.**
  + Many Vol.4 results (confinement, mass gaps, RG running) are already solid at the V1 level.
  + The V2 suite will not redo everything, but it will:
    - pick key tests (e.g., a minimal confinement benchmark, a basic RG run),
    - implement them as present-act engine simulations,
    - and show that the same phenomena reappear when expressed in pure engine terms.

In all of these cases, the V1 simulations in this attachment serve as **reference baselines**: V2 runs should be recognizably “the same story” when you strip away the additional engine detail.

**7.3.3 How Simulation Attachment – Part II Will Connect Back to This Document**

Simulation Attachment – Part II (V2 simulations) will be structured so that a reader can easily move between the **V1 vantage** (this document) and the **V2 vantage**:

* **Section-by-section cross-references.**
  + For each major V2 simulation (e.g., ParentGate gravitation test, engine-level MICC, engine-level pointer dynamics), Part II will explicitly list:
    - which V1 simulations it corresponds to or supersedes,
    - which V1 results it aims to reproduce or refine.
  + For example: a V2 pointer-dynamics section will reference Sim 3a/3b here and show how the engine-native version extends those partial results.
* **Shared concepts, different implementations.**
  + Many concepts—ladder bands, pivot (D(0)=2), FPHS kernels, compact-curvature sources—will appear in both attachments:
    - here, in V1 language (operators, kernels, lattices),
    - in Part II, in V2 language (manifest entries, gates, budgets).
  + The Bridge document provides the **conceptual map**; the two attachments provide the **empirical map** on either side.
* **Status propagation.**
  + Where this Part I document lists a simulation as *Pending / partial*, Part II will revisit that item and either:
    - upgrade it to *Passed* in an engine-native form, or
    - explicitly document any remaining obstructions or refinements still required.
  + The combined status tables from Parts I and II will give a unified view of which aspects of the theory have been tested at what level.

In this way, the V1 simulation attachment you’re reading now should be seen as **the first half of a continuous record**: it documents the initial, theory-first simulation campaign, while Simulation Attachment – Part II will document the engine-first campaign, using V2 and the Bridge as the organizing principles and treating the V1 results here as its starting point rather than something to be replaced.

**7.4 Outstanding Items, Skipped Sims, and Next Steps**

This final subsection closes out Part I by being explicit about what is **not** covered here, what remains deliberately unfinished at the V1 stage, and how those gaps are intended to be filled.

**7.4.1 Outstanding Items Carried Forward to V2**

There are two main “open fronts” that this attachment records but does not resolve:

* **Pointer Dynamics Classicalization (Sim 3a / Sim 3b).**
  + What we have:
    - fully working 2D Schrödinger–Lindblad engines in both the Wilson potential and the compact-curvature potential (\Phi\_{\text{CC}}),
    - validated (f=0) quantum baselines (wavepacket spreading, low classical correlation, significant path deviation),
    - plumbing and stability tests for small (\gamma>0).
  + What is still outstanding:
    - systematic sweeps over (f>0) (or equivalently (\gamma>0)),
    - extraction of Corr(f), (\Delta\_{\text{path}}(f)), visibility suppression, and other classicalization indicators as proper functions of measurement strength,
    - direct comparison of those pointer-level curves with field-level MICC curves from Sim 1 and the gravity-like behaviour from Sim 2b.
  + These tasks are explicitly assigned to the **V2 simulation suite**, where pointer and measurement processes will be expressed directly in present-act engine terms and subject to manifest/audit constraints.
* **Engine-Native Re-runs of Key Vol.4 / Vol.5 Tests.**
  + Most Vol.4 and Vol.5 results are already solid at the V1 discretization level, but we expect V2 to:
    - implement at least one end-to-end confinement test using ParentGate and typed budgets inside the engine,
    - redo a minimal MICC-on-FPHS-style test with gates and feature checks rather than external post-processing,
    - embed a compact-curvature–style gravity test (the essence of Sim 2b) as a pure feasibility geometry phenomenon inside the engine.
  + The goals here are not to overturn V1, but to **tighten the link**: show explicitly, in code, that the same behaviours emerge when all the V2 constraints are enforced by design.

These outstanding items are not weaknesses of the current record; they are clearly marked **next steps** that Part II will pick up.

**7.4.2 Simulations Deliberately Excluded from the Record**

A few repositories were *not* included as first-class simulations in this attachment, either because they are tooling, abandoned prototypes, or variants folded into a parent sim. For completeness:

* **V1-absolute-relativity-vol6-hero**
  + Early attempt at a “hero” end-to-end run for a later volume.
  + Never brought to the same level of completion, documentation, or interpretation as FPHS; it does not add distinct physics beyond what FPHS and the Vol.5 sims already provide.
  + **Status in this record:** omitted as a primary simulation; treated as an exploratory prototype.
* **V1-ar-v1-ai-pack**
  + A bundle of helper scripts, exploratory notebooks, and automation tools for running and analysing V1 sims.
  + Contains no unique physics tests; its contents are either duplicated or superseded by the individual repositories documented here.
  + **Status in this record:** explicitly treated as infrastructure/tooling, not as an independent simulation.
* **V1-vol4-flip-count-simulator-errorbars**
  + A technical branch devoted to refining error bars and statistical diagnostics for the main Flip Count Simulator.
  + Its role is purely methodological; the actual flip-count maps used throughout Vol.4/Vol.5 come from the parent sim (V1-vol4-flip-count-simulator).
  + **Status in this record:** folded into the Flip Count Simulator entry as “error-bar extension,” not counted as a separate simulation.

Leaving these out as stand-alone entries keeps the attachment focused on simulations that either:

* test a distinct theoretical claim, or
* generate data that many other sims rely on.

**7.4.3 Summary of Part I and Transition to Part II**

With all sections completed, Part I of the simulation attachment now provides:

* a **complete catalogue** of the V1-era simulations that matter for the theory’s first published form;
* a clear **pass / obstruction / pending** status for each;
* a narrative showing how they collectively support the **V1 formalism**, foreshadow the **V2 engine**, and are stitched together conceptually by the **Bridge**;
* and an explicit list of **next-step tasks** (pointer classicalization, engine-native MICC/ParentGate tests) that the V2 simulation attachment will take up.

Simulation Attachment – Part II will begin from exactly this point:

* it will treat the V1 simulations recorded here as **baseline evidence**,
* it will implement the engine-level extensions and completions indicated in 7.4.1,
* and it will cross-reference back to this document wherever a V2 simulation can be seen as a direct refinement or re-expression of one of the V1 sims.

Together, the two attachments will form a single, continuous simulation record for the first integrated V1+V2+Bridge version of Absolute Relativity.