The Holographic Hilbert Universe: Constructive Spacetime, Computational-Lapse Gravity, and a Conditional Riemann Critical-Line Rigidity Theorem in HPA–Ω

Haobo Ma (Auric)*
AELF PTE LTD.
#14-02, Marina Bay Financial Centre Tower 1, 8 Marina Blvd, Singapore 018981

December 27, 2025

Abstract

We propose a constructive spacetime framework in the HPA–Ω (Holographic Phase Arithmetic / Omega axiom system) scan–readout paradigm, designed to unify discrete computation and continuous geometry under finite information and finite resolution. The central mechanism is an explicit *scan–projection readout* discipline: operational time is defined as an internal relational scale of phase scanning, while probability is induced by finite-resolution readout kernels (POVMs) rather than imposed as an external axiom. Intrinsic noncommutativity is captured by a Weyl pair, providing a structural source of complementarity and uncertainty.

To fold a one-dimensional instruction stream into higher-dimensional locality, we introduce a *Hilbert Folding Axiom*: at each resolution level n, discrete time steps are mapped by a Hilbert space-filling address to a d-dimensional holographic screen lattice, turning spatial adjacency into a verifiable addressing effect. We define *computational lapse* via routing overhead $\kappa(x)$, and identify the lapse field $\mathcal{N}(x) = \kappa_0/\kappa(x)$ as an operational redshift factor. Using discrepancy certificates and Abel finite-part regularization along the canonical path $r \uparrow 1$, we construct auditable trace observables and a phase potential whose Newtonian limit exhibits a $1/r$ profile.

Finally, under a single explicit bridge assumption—a holographic trace formula (HTF) embedding zeta-zero modes into an Abel-regularized scan trace—we prove a conditional rigidity statement: any nontrivial zero off the critical line would force an interior pole in the unit disk $|r| < 1$, contradicting holomorphy of the geometric side dictated by bounded scan readout. We conclude with reproducible scripts: Hilbert-address locality checks, golden-scan discrepancy bounds, a toy Abel pole-barrier demonstration, an FFT Poisson solver for the phase potential, and a Wigner–Smith time-delay interface for extracting $\kappa_{\mathrm{WS}}(E)$.

Keywords: Hilbert curve; space-filling address; HPA–Ω; scan–projection readout; Weyl pair; POVM; star discrepancy; computational lapse; routing overhead; Abel finite part; holographic trace formula; Riemann hypothesis (conditional).

Conventions. Unless otherwise stated, log denotes the natural logarithm. We use $t \in \mathbb{Z}_{\geq 0}$ for discrete protocol time (ticks), $n \in \mathbb{N}$ for resolution level, and $r \in (0, 1)$ for Abel weights. The *canonical Abel path* refers to the limit process $r \uparrow 1$ with $r \in (0, 1)$, applied to Abel generating functions that are defined (and holomorphic) for $|r| < 1$.

*Email: auric@aelf.io

Contents

1 Introduction

Classical continuum models implicitly assume access to arbitrarily fine resolution. At singularities and at infinity, this assumption becomes not merely inconvenient but structurally unstable: divergences are symptoms of a mismatch between infinite-resolution idealization and the finite information/finite computation available to any operational observer. The HPA–Ω stance is that *operational time and operational probability* must be endogenous to a scan–readout protocol, rather than externally postulated semantics.

Context in the holography literature. The broader motivation is consistent with the holographic principle and the modern "bulk locality from boundary structure" viewpoint [1–6]. The distinguishing emphasis of the present work is constructive and protocol-native: we insist that locality, time, and probability are defined at finite resolution and are auditable within a scan–readout discipline.

Comparison to tensor-network locality constructions. Constructive bulk locality is also pursued in tensor-network approaches, most prominently MERA/entanglement-renormalization constructions and their holographic interpretations [7, 8]. Our Hilbert-folding move is different in spirit: it makes locality an explicit *address effect* at each finite resolution n, and then treats compilation/routing overhead as a physical resource that can be calibrated.

Comparison to complexity–gravity proposals and computational capacity. The idea that computational resources are geometrized in gravity has an extensive literature (e.g. complexity=action/volume and related proposals) [9, 10]. Our "computational lapse" dictionary differs in that it is tied to a concrete, auditable overhead $\kappa(x)$ for implementing screen-local tasks, and it yields direct redshift-type ratios at the protocol level. At a schematic level, κ can be read as a local cost multiplier in any circuit-depth or routing-cost functional for realizing screen-local operations on a given substrate. A quantitative comparison to complexity=volume/action proposals would therefore proceed by fixing a specific cost functional and checking whether the induced lapse factors renormalize local costs in a way compatible with known bulk/boundary variational structures (rather than only at the level of analogies). It is also complementary in scope to recent "complexity equals anything" directions and related developments in alternative complexity measures and deformations [11–16]. For broader discussions of computation as a physical bound, see also [17].

Relation to discrete spacetime programs. Hilbert folding is one particular constructive route from a discrete substrate to an effective geometry. It is complementary to causal-set approaches, where causal order is the primitive and continuum geometry is reconstructed statistically [18].

Modern uncertainty frameworks. On the quantum side, beyond variance-form bounds such as Massar–Spindel, the literature includes entropic uncertainty relations and computable (SDP/polynomial-optimization) uncertainty frontiers [19–22]. Section 6 situates the protocol tradeoff in this broader landscape.

Operator-theoretic RH programs. Our conditional RH mechanism isolates the hard content into a single trace-bridge assumption (HTF), in the tradition of "explicit formula as trace" approaches [23]. This is compatible with, but logically distinct from, the Hilbert–Pólya/quantization viewpoint and the Berry–Keating $H = xp$ heuristic, which aim to produce a concrete spectral object whose mode decomposition encodes the zeros [24]. Recent explicit operator constructions, including prolate-wave operators, exemplify the kind of spectral input

HTF presupposes [25]. To make clear that HTF is not an empty slogan, Appendix E records a fully explicit HTF-style trace identity in the function-field setting (Weil zeta for curves over finite fields), where the analogue of RH is known and the pole-barrier mechanism becomes a transparent spectral-radius statement.

Scan–projection as the primitive interface. In the HPA–Ω closed layer, the observer does not directly access an external continuous time parameter. Instead, the observer accesses a finite-resolution readout of an intrinsic phase scan. Probability is induced by the readout instrument (POVM/effects) at a given resolution ε, and nonclassicality arises from two structural sources: (i) the intrinsic noncommutativity of a Weyl pair in the scan algebra, and (ii) the finite-resolution projection kernel that defines the effective readout statistics.

The missing link: from one-dimensional scan to higher-dimensional locality. To upgrade scan–readout from a deterministic sampling mechanism into a constructive spacetime theory, one must answer a concrete question: *how can a one-dimensional instruction stream produce a verifiable notion of multi-dimensional spatial locality?* This paper proposes an explicit and auditable answer: *Hilbert folding*.

Hilbert folding: space as an address effect. At each finite resolution level n, we define a d-dimensional holographic screen lattice $\Sigma_n = \{0, 1, \ldots, 2^n - 1\}^d$. A discrete Hilbert space-filling address map $H_n : \{0, 1, \ldots, 2^{dn} - 1\} \to \Sigma_n$ is a bijection whose key property is *one-step locality*: consecutive indices map to neighboring lattice sites. We elevate the existence of such an address family to an axiom (Hilbert Folding Axiom, H1). With H1, spatial locality is not a background geometric assumption but a checkable compilation/addressing effect.

Computational-lapse gravity. Once locality is defined on the screen lattice, the implementation question becomes physical: to realize screen-local interactions using a one-dimensional scan substrate (nearest-neighbor gates in scan time), one must pay a routing/compilation overhead. We define a local routing overhead $\kappa(x)$ and introduce the computational lapse field

$$\mathcal{N}(x) = \frac{\kappa_0}{\kappa(x)}.$$

Operationally, $\mathcal{N}$ is the ratio between local relational time and global ticks, and therefore functions as a redshift factor. In the weak-field regime, a Poisson closure for a phase potential yields the expected $1/r$ profile and a Newtonian acceleration template.

A conditional Riemann critical-line rigidity derivation. The final part of the paper addresses the Riemann Hypothesis (RH). We do *not* claim an unconditional proof in classical analytic number theory. Instead, we state an explicit bridge assumption—a holographic trace formula (HTF) that embeds the zero-mode contribution of ζ into an Abel-regularized scan trace consistent with the Ω finite-part convention. Under HTF, we prove a closed-layer rigidity theorem: any off-critical zero ρ forces an interior pole of the spectral-side Abel factor at $r = \mathrm{e}^{-(\rho - \frac{1}{2})}$ with $|r| < 1$, contradicting holomorphy of the geometric side dictated by bounded scan readout. Thus, *given HTF*, all nontrivial zeros satisfy $\mathrm{Re}(\rho) = \frac{1}{2}$.

Contributions and organization. The paper provides:

- a concrete Hilbert-folding axiom that turns 1D scan time into d-dimensional address locality;
- an operational definition of computational lapse as routing overhead, with a redshift/time-dilation dictionary;

- an auditable regularization discipline (Abel first, then finite part) compatible with finite information readout;
- a conditional critical-line rigidity theorem under a single explicit trace-bridge assumption (HTF);
- reproducible scripts implementing Hilbert addressing checks, discrepancy certificates, Abel pole-barrier toys, and Poisson potential solvers.

Section 2 collects protocol objects (scan sequences, discrepancy, Abel regularization) and the discrete Hilbert addressing template. Section 3 states the closed-layer axioms used in the paper, including the Hilbert Folding Axiom. Sections 4–5 develop constructive spacetime from Hilbert folding. Sections 6–7 treat intrinsic quantum structure and computational-lapse gravity. Section 8 sketches a minimal discrepancy-driven dynamical closure. Appendix D gives the conditional RH rigidity theorem. Section 9 provides reproducible numerics and scripts.

2 Preliminaries: scan objects, discrepancy certificates, Hilbert addressing, and Abel regularization

2.1 Kronecker phase scans and Abel regularization

The minimal deterministic scan model is a Kronecker rotation on the circle:

$$x_t = x_0 + t\alpha \pmod 1, \qquad t \in \mathbb{Z}_{\geq 0}, \quad \alpha \in (0,1) \setminus \mathbb{Q}. \tag{1}$$

Given a bounded readout kernel $f : \mathbb{T} \to \mathbb{C}$, the *Abel-regularized orbit sum* is

$$S_f(r) := \sum_{t \geq 0} r^t f(x_t), \qquad 0 < r < 1. \tag{2}$$

For every $|r| < 1$, the series converges absolutely and defines a holomorphic function of r. This "move infinity to the boundary" principle is the closed-layer replacement for informal divergent orbit sums.

Fourier-resolvent form. If f has an absolutely summable Fourier series,

$$f(x) = \sum_{m \in \mathbb{Z}} \widehat{f}(m)\, \mathrm{e}^{2\pi \mathrm{i} m x}, \qquad \sum_{m \in \mathbb{Z}} \left|\widehat{f}(m)\right| < \infty,$$

then substituting into (2) yields the resolvent identity

$$S_f(r) = \sum_{m \in \mathbb{Z}} \widehat{f}(m)\, \mathrm{e}^{2\pi \mathrm{i} m x_0}\, \frac{1}{1 - r\, \mathrm{e}^{2\pi \mathrm{i} m \alpha}}, \qquad |r| < 1. \tag{3}$$

Thus, S_f is holomorphic on the unit disk and admits a controlled boundary expansion as $r \uparrow 1$.

Finite-part extraction. For uniquely ergodic rotations, Abel means converge to the space average for sufficiently regular f. We define the Abel finite part along the canonical path $r \uparrow 1$ by

$$\mathrm{FP} \sum_{t \geq 0} f(x_t) := \lim_{r \uparrow 1} \left(S_f(r) - \frac{\int_0^1 f(x)\, \mathrm{d}x}{1 - r} \right), \tag{4}$$

whenever the limit exists. In the Ω discipline, admissibility of an "infinite" protocol quantity is tied to the existence of such a canonical finite part.

2.2 Star discrepancy and deterministic error certificates

Let $P_N = \{x_0, \dots, x_{N-1}\} \subset [0,1)$ be the scan prefix. The one-dimensional star discrepancy is

$$D_N^*(P_N) \;:=\; \sup_{u\in[0,1]} \left| \frac{1}{N} \#\{t < N : x_t < u\} - u \right|. \tag{5}$$

The discrepancy controls deterministic sampling error via Koksma-type bounds: for functions of bounded variation $\mathrm{Var}(f)$,

$$\left| \frac{1}{N} \sum_{t=0}^{N-1} f(x_t) - \int_0^1 f(x)\,\mathrm{d}x \right| \le \mathrm{Var}(f)\, D_N^*(P_N), \tag{6}$$

see [26, 27].

Golden-branch logarithmic stability. For the golden slope $\alpha = \varphi^{-1}$ with $\varphi = (1+\sqrt{5})/2$, a convenient explicit bound is

$$D_N^*(P_N) \;\le\; \frac{2(2+\log_\varphi N)}{N}, \tag{7}$$

which we will use as an illustrative certificate in Section 9; see Appendix B and [26, 27].

2.3 Discrete Hilbert addressing (space-filling folding at finite resolution)

Fix an integer dimension $d \ge 2$. At resolution $n \in \mathbb{N}$, define the d-dimensional screen lattice

$$\Sigma_n := \{0, 1, \dots, 2^n - 1\}^d, \qquad |\Sigma_n| = 2^{dn}. \tag{8}$$

Definition 2.1 (Discrete Hilbert address map)**.** *A map $H_n : \{0, 1, \dots, 2^{dn} - 1\} \to \Sigma_n$ is a discrete Hilbert address map if it is a bijection and if the ordered list $(H_n(0), H_n(1), \dots, H_n(2^{dn} - 1))$ forms a Hamiltonian path on the grid with one-step locality.*

For standard Hilbert constructions, one-step locality holds in the Manhattan metric:

$$\|H_n(t+1) - H_n(t)\|_1 = 1, \qquad 0 \le t < 2^{dn} - 1. \tag{9}$$

In this paper, H_n is treated as a finite-resolution *addressing primitive*. The continuous Hilbert curve is a classical limiting object, but operational claims are always formulated at finite n.

2.4 Weyl pairs and finite-resolution readout

Let $\mathcal{H}_{\mathrm{eff}}$ be an effective observer Hilbert space. A Weyl pair (U, V) is a pair of unitaries satisfying

$$UV = \mathrm{e}^{2\pi \mathrm{i}\alpha}\, VU, \tag{10}$$

which is the algebraic source of complementarity for scan shift and phase pointer. In the canonical circle representation,

$$(U\psi)(x) = \psi(x+\alpha), \qquad (V\psi)(x) = \mathrm{e}^{2\pi \mathrm{i} x}\psi(x),$$

U implements the Kronecker shift (1).

Finite-resolution readout is encoded by a POVM $\{E_k^{(\varepsilon)}\}_k$ (effects summing to $\mathbf{1}$) and an effective state ω_{eff}:

$$\mathbb{P}_k^{(\varepsilon)} = \omega_{\mathrm{eff}}(E_k^{(\varepsilon)}), \qquad \sum_k E_k^{(\varepsilon)} = \mathbf{1}, \tag{11}$$

see [28–30].

2.5 Routing overhead and computational lapse

Hilbert addressing turns scan time into screen locality, but implementing screen-local interactions using a one-dimensional scan substrate requires compilation. We denote by $\kappa(x)$ a local routing overhead (e.g. minimal swap depth) required to enact a prescribed screen-local operation near lattice site $x \in \Sigma_n$. The *computational lapse* is defined by

$$\mathcal{N}(x) = \frac{\kappa_0}{\kappa(x)}, \tag{12}$$

where κ_0 is a reference overhead in a homogeneous region. This provides an operational redshift dictionary: larger compilation cost corresponds to smaller local proper time per global tick.

2.6 Abel mode factors and an interior pole barrier

The conditional RH mechanism used later is a simple analytic fact about Abel transforms of exponential modes.

Lemma 2.2 (Abel pole barrier for off-critical growth)**.** *Fix $\rho \in \mathbb{C}$ and define $a_t = \mathrm{e}^{(\rho-\frac{1}{2})t}$. For $|r| < 1$,*

$$\sum_{t\geq 0} r^t a_t = \frac{1}{1 - r\,\mathrm{e}^{(\rho-\frac{1}{2})}}.$$

If $\mathrm{Re}(\rho) > \frac{1}{2}$, then the right-hand side has a pole at

$$r_\rho := \mathrm{e}^{-(\rho-\frac{1}{2})}, \qquad |r_\rho| = \mathrm{e}^{-(\mathrm{Re}(\rho)-\frac{1}{2})} < 1.$$

Proof. This is the geometric series identity for the complex ratio $r\,\mathrm{e}^{(\rho-\frac{1}{2})}$. If $\mathrm{Re}(\rho) > \frac{1}{2}$, then $\left|\mathrm{e}^{(\rho-\frac{1}{2})}\right| > 1$, so the reciprocal r_ρ satisfies $|r_\rho| < 1$ and annihilates the denominator. □

Lemma 2.2 will be paired with the holomorphy of bounded scan traces on $|r| < 1$ to obtain a contradiction under the HTF bridge assumption.

2.7 Minimal primitives and terminology map

To streamline terminology, Table 1 lists the small set of primitives used in the closed-layer arguments and their standard mathematical counterparts.

Normalization conventions. The reference overhead κ_0 fixes the zero of the potential $\Phi = \log(\kappa/\kappa_0)$ and should be chosen in a homogeneous reference region or reference band. In the time-delay interface, τ_0 fixes the dimensionless scale of $\kappa_{\mathrm{WS}}(E) = \tau_{\mathrm{WS}}(E)/\tau_0$ (Appendix F).

3 Axioms and layer discipline in HPA–Ω (with a Hilbert-folding extension)

3.1 Closed layer vs. interpretation layer

We adopt a strict two-layer writing discipline:

- **Closed layer**: explicit axioms, regularization conventions, and theorems provable from them.
- **Interpretation layer**: physical narratives (black holes, wormholes, agency, etc.) that do not enter the proof chain.

All formal results in this paper are stated in the closed layer, with assumptions clearly declared.

HPA–Ω term / symbol	Standard object / reading
scan ticks t	discrete time index (protocol time)
scan unitary U_{scan}	unitary shift/rotation implementing scan dynamics
phase pointer V and Weyl relation	noncommutative Weyl pair (complementarity source)
readout kernel / POVM $\{E_k^{(\varepsilon)}\}$	generalized measurement (effects at resolution ε)
effective state ω_{eff}	normal state; in finite dimensions $\omega_{\mathrm{eff}}(E) = \mathrm{Tr}(\rho E)$
Hilbert folding H_n	discrete space-filling address map (Hamiltonian path locality)
routing overhead $\kappa(x; G_{\mathrm{phys}}, \pi_n, \mathcal{G})$	compilation/routing depth for a fixed local task family on a specified hardware graph and placement
time-delay overhead proxy $\kappa_{\mathrm{WS}}(E)$	Wigner–Smith total delay normalized by τ_0 (energy/scale-dependent experimental proxy; distinct from spatial $\kappa(x)$)
computational lapse $\mathcal{N}(x) = \kappa_0/\kappa(x)$	ratio of local relational time to global ticks (redshift factor)
computational potential $\Phi(x) = -\log \mathcal{N}(x)$	dimensionless potential; $\phi_N = -c^2\Phi$ in Newtonian units
star discrepancy D_N^*	deterministic equidistribution error certificate
Abel regularization $S(r) = \sum a_t r^t$	holomorphic generating function on $\lvert r\rvert < 1$
finite part $\mathrm{FP}_{r\uparrow 1}$	constant-term extraction along the canonical Abel path
HTF	trace-bridge assumption (explicit-formula-as-trace in Abel coordinates)

Table 1: Minimal primitives and terminology map used throughout the paper.

3.2 Scan–projection readout and induced probability (O5)

Axiom 3.1 (O5: scan–projection readout and induced measure)**.** *At finite resolution $\varepsilon > 0$, the observer accesses a family of readout effects $\{E_k^{(\varepsilon)}\}_k$ (a POVM) on an effective Hilbert space $\mathcal{H}_{\mathrm{eff}}$, satisfying $\sum_k E_k^{(\varepsilon)} = \mathbf{1}$. Given an effective state ω_{eff}, the outcome probabilities are induced by*

$$\mathbb{P}_k^{(\varepsilon)} = \omega_{\mathrm{eff}}\left(E_k^{(\varepsilon)}\right).$$

Operational (discrete) time is represented by scan ticks $t \in \mathbb{Z}_{\geq 0}$ indexing the internal scan.

This axiom encodes the principle that probability is not an external semantic postulate but an interface object induced by finite-resolution readout.

3.3 Scan algebra and Weyl pair noncommutativity (O6)

Axiom 3.2 (O6: scan algebra with a Weyl pair)**.** *On $\mathcal{H}_{\mathrm{eff}}$ there exist unitary operators U_{scan} and V satisfying the Weyl relation*

$$U_{\mathrm{scan}} V = \mathrm{e}^{2\pi \mathrm{i} \alpha} V U_{\mathrm{scan}}, \qquad \alpha \in (0,1) \setminus \mathbb{Q}.$$

The noncommutativity is intrinsic and provides a structural source of complementarity and uncertainty for scan shift and phase pointer readout.

3.4 Canonical regularization path: Abel first, then finite part (R1)

Axiom 3.3 (R1: Abel first, then finite part)**.** *Infinite-horizon scan traces are defined by Abel regularization on $\lvert r\rvert < 1$ and are compared/renormalized only along the canonical path $r \uparrow 1$. When a universal divergence of the form $c_{-1}/(1-r)$ is present, the admissible protocol value is the finite-part constant term extracted along this path.*

3.5 Hilbert folding (H1): a constructive spacetime extension

Axiom 3.4 (H1: Hilbert Folding Axiom)**.** *For each resolution level $n \in \mathbb{N}$ and each fixed dimension $d \geq 2$, there exists a discrete Hilbert address map*

$$H_n : \{0, 1, \ldots, 2^{dn} - 1\} \to \Sigma_n = \{0, 1, \ldots, 2^n - 1\}^d$$

that is a bijection and satisfies one-step locality (9)*. The observer-accessible boundary degrees of freedom at resolution n are organized on the screen lattice Σ_n with nearest-neighbor locality, and scan ticks address them by $x_t^{(n)} := H_n(t \bmod 2^{dn})$.*

H1 makes "space" an explicit finite-resolution object, derived from scan time by an auditable address map. Implementation overhead (routing) then becomes a physical resource that can be measured and compared across regions.

Remark (mathematical existence). The existence of discrete Hilbert orders with one-step locality is a standard constructive fact in the space-filling curve literature [31, 32]. The axiom content of H1 is the physical identification of observer-accessible boundary degrees of freedom with such an addressable lattice at each resolution.

4 Hilbert folding: from tick time to holographic screen locality

4.1 Addressing at finite resolution

Fix $d \geq 2$. At resolution level n, the screen lattice Σ_n in (8) has 2^{dn} sites. By Axiom 3.4, there exists a bijective address map H_n with one-step locality (9). We interpret H_n as a *compiler-visible* and *auditable* specification of how scan ticks visit screen degrees of freedom.

Because H_n is a bijection, it admits an inverse

$$H_n^{-1} : \Sigma_n \to \{0, 1, \ldots, 2^{dn} - 1\},$$

and therefore supports two primitive operations:

- **forward addressing** (tick $\to$ site): $t \mapsto H_n(t)$;
- **reverse addressing** (site $\to$ tick): $x \mapsto H_n^{-1}(x)$.

This invertibility is essential: locality is not only a geometric notion but a compilable addressing relation.

4.2 Locality as a verifiable adjacency constraint

The defining local constraint (9) implies that the scan visits screen sites along a nearest-neighbor path:

$$H_n(0) \sim H_n(1) \sim \cdots \sim H_n(2^{dn} - 1),$$

where $\sim$ denotes ℓ^1 nearest-neighbor adjacency on the grid. Thus, any scan-layer update that couples consecutive ticks corresponds to a screen-local propagation.

Conversely, implementing a prescribed screen-local interaction (nearest neighbors on Σ_n) on a one-dimensional scan substrate typically requires *routing*: degrees of freedom must be swapped/moved in scan order so that neighboring sites become neighboring ticks. This is the source of computational overhead $\kappa(x)$ and, ultimately, computational lapse (Section 7).

4.3 A constructive view of "space"

Within this framework, "space" is defined at each resolution level n as the graph $(\Sigma_n, \sim)$, where $\sim$ is a chosen nearest-neighbor relation (e.g. ℓ^1). The scan is not embedded *into* a pre-existing manifold; rather, the manifold is an *effective* description of how the address-induced neighborhood structure behaves under coarse graining.

Two consequences are immediate:

- **Locality is protocol-relative.** The same underlying scan dynamics can induce different effective spatial neighborhoods under different admissible address maps (different compilation conventions).
- **Continuum geometry is an emergent coarse-grained object.** The continuous Hilbert curve $H : [0,1] \to [0,1]^d$ provides a classical scaling limit of H_n under appropriate normalization, but operational claims are always made at finite n and finite readout resolution.

A quantitative locality fact (Hölder regularity). For the continuous Hilbert curve $H : [0,1] \to [0,1]^d$, one has Hölder continuity with exponent $1/d$: there exists a constant C_d such that

$$\|H(t) - H(s)\|_\infty \le C_d\, |t-s|^{1/d} \qquad (s,t \in [0,1]),$$

see, e.g., [32]. This provides a quantitative backbone for the "finite resolution" viewpoint: small parameter increments induce small spatial displacements after coarse graining, with a deterministic scale law.

4.4 Algorithmic realization (2D) and locality check

For $d = 2$, a standard bitwise construction provides a concrete H_n and is easy to verify by brute force. Appendix A records a pure-Python implementation and checks (9) for orders up to $n = 8$ in milliseconds. This implementation is included only as an auditable reference construction consistent with the standard literature [32].

5 Constructive spacetime: coarse-grained geometry from folded scan dynamics

5.1 Locality as a compilation/addressing effect

At resolution n, the physical adjacency graph is the screen lattice $(\Sigma_n, \sim)$. The scan order provides an *addressing* of this graph, but it does not in general preserve all local interactions for free. Instead, the cost of implementing a screen-local gate using scan-nearest-neighbor primitives depends on the local routing overhead κ. This yields a natural operational meaning of "curvature" or "gravitational potential" in terms of inhomogeneous compilation cost.

In particular, regions in which κ is larger are regions in which local operations take more global ticks to realize. This is the structural origin of the lapse field (12) and of gravitational redshift in this framework (Section 7).

5.2 Why a non-differentiable address still yields smooth effective geometry

The continuous Hilbert curve is nowhere differentiable, and even the discrete address path is highly tortuous at fine scales. However, physical measurement is always performed with finite resolution. Let $w^{(\varepsilon)}$ be a readout kernel at scale ε (or, equivalently, a POVM element family $E^{(\varepsilon)}$). The effective observable is a coarse-grained field Φ_ε obtained by smoothing over many lattice sites.

In this regime, discrete differential operators on Σ_n (graph Laplacian, discrete gradients) can approximate continuum operators provided ε is large compared with the lattice spacing 2^{-n} and the readout kernel is sufficiently regular. Thus, classical smooth geometry is an *effective* description of coarse-grained fields, not a microscopic property of the address curve itself.

5.3 Scale as resolution: Hilbert self-similarity and canonical time coding

Hilbert addressing is strictly hierarchical: H_{n+1} is built recursively from H_n by subdividing the lattice into 2^d subcubes and connecting appropriately rotated/reflected subpaths. This induces a natural notion of "coarse graining": the map $n \mapsto n-1$ groups 2^d sites into a block and projects Σ_n to Σ_{n-1}.

On the time side, canonical integer codings (Ostrowski/Zeckendorf for the golden branch) provide a multi-scale decomposition of scan time. Together, these two hierarchical structures support a renormalization viewpoint in which:

- resolution level n acts as a discrete spatial scale;
- Abel weight r acts as an analytic radial coordinate (regularizing long tails before taking the boundary limit);
- coarse-grained dynamics is defined by pushing forward the scan/readout protocol through these scale maps.

5.4 Admissible address maps and robustness under coarse graining

Hilbert folding (Axiom 3.4) commits to a concrete finite-resolution addressing primitive, but the broader scan–fold paradigm admits alternative address families. This matters because the addressing enters the compilation problem and therefore the overhead field κ (Section 7.1.1). At the same time, physical predictions are formulated at finite readout resolution, so only coarse-grained features of protocol fields are observable. This subsection records a minimal robustness statement in the closed layer.

Definition 5.1 (Admissible address family (coarse-grained locality class))**.** *An address family $\{A_n\}_{n\geq 1}$ with $A_n : \{0,1,\dots,2^{dn}-1\} \to \Sigma_n$ is called admissible if each A_n is a bijection and if there exist constants $C_d > 0$ and $c_d > 0$ such that, after rescaling Σ_n to the unit cube by $\tilde{x} := 2^{-n}x$, the induced discrete path obeys a uniform Hölder-type locality bound*

$$\left\|2^{-n}A_n(t) - 2^{-n}A_n(s)\right\|_\infty \;\leq\; C_d \left|\frac{t-s}{2^{dn}}\right|^{1/d}, \qquad s,t \in \{0,\dots,2^{dn}-1\},$$

and a uniform non-degeneracy bound in the reverse direction

$$\left|\frac{t-s}{2^{dn}}\right| \;\leq\; c_d \left\|2^{-n}A_n(t) - 2^{-n}A_n(s)\right\|_\infty^d .$$

Continuous Hilbert/Peano-type constructions satisfy such bi-Hölder bounds in their classical forms; see [32].

Remark 5.2 (Hilbert vs. Z-order (Morton) locality)**.** *Different admissible families can have substantially different clustering properties at finite resolution, and therefore different compilation overhead landscapes. In spatial indexing, Hilbert orders are known to provide stronger clustering/locality than Z-order/Morton-type orders for a wide range of window queries and locality metrics; see [33]. In the present framework, this translates into a principled sensitivity channel: the map choice changes the distribution of address separations and therefore changes κ and $\mathcal{N}$ in substrate models where compilation is dominated by scan-order routing.*

Definition 5.3 (Neighbor index separation and a scan-chain overhead proxy)**.** *Fix an admissible address map* $A_n : \{0, 1, \ldots, 2^{dn}-1\} \to \Sigma_n$ *and write its inverse as* $A_n^{-1} : \Sigma_n \to \{0, 1, \ldots, 2^{dn}-1\}$. *For a screen-neighbor edge* $\{x, y\} \in E_{\text{screen}}$ *define the* scan-order separation

$$\Delta_{A_n}(x, y) := \left| A_n^{-1}(x) - A_n^{-1}(y) \right|.$$

Define the associated local scan-chain overhead proxy by

$$\widetilde{\kappa}_{A_n}(x) := \max_{y \sim x} \Delta_{A_n}(x, y),$$

interpreted as the largest scan-order separation among screen neighbors of x.

Proposition 5.4 (Scan-chain lower bound from address separations)**.** *Assume the hardware graph is the unit-weight path on vertices* $\{0, 1, \ldots, 2^{dn}-1\}$ *and the placement is* $\pi_n(x) = A_n^{-1}(x)$. *Then implementing any primitive interaction between screen neighbors* $\{x, y\} \in E_{\text{screen}}$ *using adjacent primitives on the chain requires at least* $\Delta_{A_n}(x, y)$ *global ticks. Consequently, for any local update task family* $\mathcal{G}_x$ *that requires at least one interaction between* x *and each of its neighbors, the compilation overhead satisfies*

$$\kappa(x) \;\geq\; \widetilde{\kappa}_{A_n}(x),$$

up to fixed constant factors determined by the primitive gate set and scheduling convention.

Proof. On a path graph with unit weights, information cannot propagate faster than one edge per tick. Thus any interaction between sites placed at chain positions $A_n^{-1}(x)$ and $A_n^{-1}(y)$ requires at least their graph distance, which equals $|A_n^{-1}(x) - A_n^{-1}(y)|$. The local-update bound follows by taking the maximum over the required neighbor interactions. □

Remark 5.5 (Address-family dependence of $N(x)$ on a scan chain)**.** *In the GR dictionary of Section 7, the lapse* N *is identified with the operational field* $\mathcal{N} = \kappa_0/\kappa$, *hence also with the potential* $\Phi = -\log N = \log(\kappa/\kappa_0)$. *In the scan-chain specialization* $\pi_n(x) = A_n^{-1}(x)$, *Proposition 5.4 shows that address-order neighbor separations control the overhead landscape and therefore directly control the lapse landscape. This makes the dependence of* $N(x)$ *on the choice of address family (Hilbert vs. Morton/Z-order, etc.) an experimentally testable effect: Tables 3 and 12 report the induced proxy statistics and the corresponding redshift ratios for fixed substrate model, with only the address family varied. Since all physical comparisons are formulated either as lapse ratios or as potentials (which are invariant under global tick reparameterizations; Remark 7.5), the sensitivity to address choice is a genuine spatial inhomogeneity rather than a coordinate artifact.*

Remark 5.6 (Address choice as a protocol gauge (diffeomorphism analogy))**.** *In continuum gravity, diffeomorphism invariance expresses the fact that physical predictions do not depend on the choice of coordinates. In the present constructive setting, an address family plays a role closer to a* protocol gauge choice*: it is a concrete discrete chart that fixes how scan time is folded into screen locality and therefore fixes a concrete compilation problem on a given substrate. Accordingly, at fixed microscopic substrate the lapse landscape can depend on the address choice (Remark 5.5). Operational invariance is recovered only after specifying the readout resolution: Definition 5.8 formalizes when two protocol realizations are indistinguishable at scale* ε, *and Proposition 5.7 then bounds the induced changes in coarse-grained potentials and lapse ratios. In this sense, diffeomorphism-style "gauge freedom" is replaced here by an explicit, testable universality statement at finite resolution.*

5.4.1 A stability estimate for coarse-grained potentials

Let $w^{(\varepsilon)}$ be a nonnegative readout kernel at scale ε with $\int w^{(\varepsilon)} = 1$. For any field F define its coarse-graining by convolution $F_\varepsilon := w^{(\varepsilon)} * F$. If Φ is defined as the Poisson potential of a protocol-defined source density σ (Section 8.4), then small changes in σ imply small changes in Φ in standard Sobolev norms.

Proposition 5.7 (Stability of Poisson potentials under source perturbations)**.** *Let $\Omega \subset \mathbb{R}^3$ be a domain with fixed boundary conditions (periodic, Dirichlet, or suitable decay at infinity). Let σ_1, σ_2 be two source densities and let Φ_1, Φ_2 solve*

$$-\Delta\Phi_i = 4\pi G\,\sigma_i$$

with the same boundary convention. Then

$$\|\nabla(\Phi_1 - \Phi_2)\|_{L^2(\Omega)} \;\le\; 4\pi G\,\|\sigma_1 - \sigma_2\|_{H^{-1}(\Omega)}\,.$$

Proof. Set $\delta\Phi = \Phi_1 - \Phi_2$ and $\delta\sigma = \sigma_1 - \sigma_2$. Then $-\Delta\delta\Phi = 4\pi G\,\delta\sigma$. Testing against $\delta\Phi$ and integrating by parts yields

$$\|\nabla\delta\Phi\|^2_{L^2(\Omega)} = 4\pi G\,\langle\delta\sigma, \delta\Phi\rangle \le 4\pi G\,\|\delta\sigma\|_{H^{-1}(\Omega)}\,\|\nabla\delta\Phi\|_{L^2(\Omega)}\,,$$

hence the claimed estimate. □

Definition 5.8 (Resolution-ε universality class (protocol equivalence))**.** *Fix a boundary convention on Ω and a readout kernel $w^{(\varepsilon)}$ of width ε with $\int w^{(\varepsilon)} = 1$. Consider two protocol realizations (address family, routing scheme, hardware graph G_{phys}, placement π_n, horizon and regularization budgets, etc.) that induce source densities σ_1, σ_2 in the sense of Definition 8.7. Define the coarse-grained sources $\sigma_{i,\varepsilon} := w^{(\varepsilon)} * \sigma_i$. For a tolerance $\delta > 0$, we say the two realizations are* (ε, δ)-equivalent *if*

$$\|\sigma_{1,\varepsilon} - \sigma_{2,\varepsilon}\|_{H^{-1}(\Omega)} \;\le\; \delta.$$

The collection of realizations mutually (ε, δ)-equivalent forms an operational universality class at readout resolution ε.

Proposition 5.9 (Coarse-grained invariance of potentials and lapse ratios in a universality class)**.** *Fix a boundary convention on Ω such that a Poincaré-type inequality holds in the chosen gauge (e.g. periodic with zero-mean gauge, or Dirichlet with $\Phi|_{\partial\Omega} = 0$), and let C_P be the corresponding constant:*

$$\|f\|_{L^2(\Omega)} \le C_P\,\|\nabla f\|_{L^2(\Omega)}\,.$$

Let two protocol realizations be (ε, δ)-equivalent in the sense of Definition 5.8. Let $\Phi_{i,\varepsilon}$ denote the Poisson potentials of the coarse-grained sources:

$$-\Delta\Phi_{i,\varepsilon} = 4\pi G\,\sigma_{i,\varepsilon}$$

with the same boundary convention and gauge choice, and define the corresponding coarse-grained lapse fields $N_{i,\varepsilon} := \mathrm{e}^{-\Phi_{i,\varepsilon}}$. Then:

1. ***Energy-norm invariance.***

$$\|\nabla(\Phi_{1,\varepsilon} - \Phi_{2,\varepsilon})\|_{L^2(\Omega)} \;\le\; 4\pi G\,\delta.$$

2. ***Pointwise invariance after readout smoothing.*** *For any nonnegative normalized kernel $\widetilde{w}^{(\varepsilon)}$ used to read out the potential at scale ε, define $\widetilde{\Phi}_{i,\varepsilon} := \widetilde{w}^{(\varepsilon)} * \Phi_{i,\varepsilon}$ and $\widetilde{N}_{i,\varepsilon} := \mathrm{e}^{-\widetilde{\Phi}_{i,\varepsilon}}$. Then*

$$\left\|\widetilde{\Phi}_{1,\varepsilon} - \widetilde{\Phi}_{2,\varepsilon}\right\|_{L^\infty(\Omega)} \;\le\; 4\pi G\,C_P\,\left\|\widetilde{w}^{(\varepsilon)}\right\|_{L^2(\Omega)}\,\delta,$$

and therefore for all $x \in \Omega$ one has the multiplicative lapse bound

$$\exp(-\Delta_\varepsilon) \;\le\; \frac{\widetilde{N}_{1,\varepsilon}(x)}{\widetilde{N}_{2,\varepsilon}(x)} \;\le\; \exp(\Delta_\varepsilon), \qquad \Delta_\varepsilon := 4\pi G\, C_P \left\|\widetilde{w}^{(\varepsilon)}\right\|_{L^2(\Omega)} \delta.$$

Proof. By Definition 5.8, one has the coarse-grained source bound

$$\|\sigma_{1,\varepsilon} - \sigma_{2,\varepsilon}\|_{H^{-1}(\Omega)} \le \delta.$$

Applying Proposition 5.7 to the pair $(\sigma_{1,\varepsilon}, \sigma_{2,\varepsilon})$ yields the energy-norm bound. For the pointwise statement, set $\delta\Phi_\varepsilon := \Phi_{1,\varepsilon} - \Phi_{2,\varepsilon}$ and note

$$\widetilde{\Phi}_{1,\varepsilon} - \widetilde{\Phi}_{2,\varepsilon} = \widetilde{w}^{(\varepsilon)} * \delta\Phi_\varepsilon.$$

Young's inequality gives $\left\|\widetilde{w}^{(\varepsilon)} * \delta\Phi_\varepsilon\right\|_{L^\infty} \le \left\|\widetilde{w}^{(\varepsilon)}\right\|_{L^2} \|\delta\Phi_\varepsilon\|_{L^2}$. The Poincaré inequality bounds $\|\delta\Phi_\varepsilon\|_{L^2} \le C_P \|\nabla\delta\Phi_\varepsilon\|_{L^2}$. Combining with the energy estimate yields the stated L^∞ bound. The multiplicative lapse bound follows from $\widetilde{N}_{i,\varepsilon} = \exp(-\widetilde{\Phi}_{i,\varepsilon})$. □

Remark 5.10 (Practical robustness criterion at resolution ε)**.** *If a change of address map (or other protocol parameters) modifies the source density only at scales below the readout resolution, then the coarse-grained sources $\sigma_{1,\varepsilon}$ and $\sigma_{2,\varepsilon}$ are close, and Proposition 5.7 implies that the coarse-grained potentials $\Phi_{1,\varepsilon}$ and $\Phi_{2,\varepsilon}$ (hence $\mathcal{N}_{i,\varepsilon} = \mathrm{e}^{-\Phi_{i,\varepsilon}}$) are close as well. Thus, sensitivity of physical conclusions to the address map can be made quantitative: it is controlled by the norm of the induced source perturbation after coarse graining. Equivalently, within the universality class of Definition 5.8, the coarse-grained lapse field $N_\varepsilon = \mathcal{N}_\varepsilon$ is insensitive to microscopic address/graph details beyond their impact on the coarse-grained source.*

6 Intrinsic quantumness: Weyl pairs, induced probability, and uncertainty as a protocol tradeoff

6.1 Weyl pairs as the algebraic source of complementarity

By Axiom 3.2, the scan algebra contains a Weyl pair (U_{scan}, V) satisfying

$$U_{\mathrm{scan}} V = \mathrm{e}^{\mathrm{i}\theta} V U_{\mathrm{scan}}, \qquad \theta := 2\pi\alpha \not\equiv 0 \pmod{2\pi}.$$

This obstructs simultaneous diagonalization: one cannot prepare a state that is arbitrarily close to an eigenstate of both scan shift and phase pointer. In the scan–readout semantics, this is the structural origin of complementarity.

6.2 Readout instruments and the derived Born form

At resolution ε, the readout channel is encoded by a POVM $\{E_k^{(\varepsilon)}\}$ (Axiom 3.1). The probability assignment is induced by the effective state:

$$\mathbb{P}_k^{(\varepsilon)} = \omega_{\mathrm{eff}}\left(E_k^{(\varepsilon)}\right).$$

In finite-dimensional approximations, this becomes the Born form $\mathbb{P}_k = \mathrm{Tr}(\rho E_k)$. Thus, probability is a derived interface object controlled by the readout resolution and the instrument design, rather than an external stochastic postulate [28–30].

Born form from effect-additivity (Gleason–Busch). The appearance of $\mathrm{Tr}(\rho E)$ is not an extra modeling choice once one commits to POVM-based readout and noncontextual additivity on effects. In a finite-dimensional $\mathcal{H}_{\mathrm{eff}}$, every normal state ω_{eff} has the form $\omega_{\mathrm{eff}}(E) = \mathrm{Tr}(\rho E)$ for a density operator ρ. More generally, the following theorem shows that *any* consistent probability assignment on effects is necessarily of this form.

Theorem 6.1 (Gleason–Busch representation for POVMs (informal statement))**.** *Let $\mathcal{H}$ be a complex Hilbert space and let μ be a map assigning probabilities to effects E ($0 \le E \le \mathbf{1}$) such that:*

- *$\mu(\mathbf{1}) = 1$ and $\mu(E) \ge 0$;*
- *for any finite POVM $\{E_k\}_k$ with $\sum_k E_k = \mathbf{1}$, one has $\sum_k \mu(E_k) = 1$ (equivalently, μ is finitely additive on coexistent effects).*

Then there exists a density operator ρ such that $\mu(E) = \mathrm{Tr}(\rho E)$ for all effects E.

Reference. This is a POVM/effect-space version of Gleason's theorem; see [30] for a simple proof and discussion of hypotheses. □

6.3 A variance-form uncertainty relation (Massar–Spindel)

Following [34], for a normalized state $|\psi\rangle$ define the unitary variances

$$\Delta_U^2 := 1 - |\langle\psi| U |\psi\rangle|^2 , \qquad \Delta_V^2 := 1 - |\langle\psi| V |\psi\rangle|^2 .$$

Theorem 6.2 (Weyl-pair uncertainty (variance form))**.** *Let U, V be unitary operators satisfying $UV = \mathrm{e}^{\mathrm{i}\theta} VU$ with the phase chosen so that $0 \le \theta \le \pi$, and set $A = \tan(\theta/2)$. Then for any normalized state $|\psi\rangle$,*

$$(1 + 2A)\, \Delta_U^2\, \Delta_V^2 + A^2(\Delta_U^2 + \Delta_V^2) \;\ge\; A^2. \tag{13}$$

Reference. This is Theorem 1 of [34]; see also the derivation in [35, Appendix]. □

6.4 Entropic uncertainty at finite resolution

Variance bounds such as (13) capture one aspect of complementarity. Finite-resolution readout naturally also supports *entropic* uncertainty relations, which are often tighter and directly operational in information-theoretic tasks. For two projective measurements corresponding to orthonormal bases $\{|x_i\rangle\}$ and $\{|z_j\rangle\}$, the Maassen–Uffink bound states

$$H(X) + H(Z) \;\ge\; -2 \log c, \qquad c := \max_{i,j} |\langle x_i | z_j\rangle| ,$$

where $H(\cdot)$ is the Shannon entropy of the induced outcome distribution [19]. With quantum side information (memory), the bound strengthens to a conditional-entropy form [20].

6.5 SDP and polynomial-optimization uncertainty frontiers

Beyond analytic inequalities, modern approaches compute tight uncertainty tradeoffs for fixed finite measurement families using convex optimization. For finite observables and POVMs, measurement-uncertainty regions can be characterized and approximated by semidefinite programs [21]. More recently, state polynomial optimization provides systematic hierarchies yielding uncertainty relations and tight constants from polynomial constraints on states [22]. These frameworks provide a quantitative route from a specified instrument family $\{E_k^{(\varepsilon)}\}_k$ to testable uncertainty frontiers. This complements the variance-form Weyl-pair bound.

6.6 Protocol interpretation: address sharpness vs. global context

Equation (13) admits a direct operational translation in the scan–readout setting:

- making the phase pointer channel sharp (small Δ_V) requires a more selective readout instrument, which increases the cost of maintaining global coherence under scan dynamics;
- making scan access "cheap" (small Δ_U, near invariance under U_{scan}) reduces distinguishability across ticks and forces longer horizons or deeper protocols to accumulate reliable information.

Thus, uncertainty is reinterpreted as a *resource lower bound* reflecting the incompatibility between sharpening local addresses and retaining global contextual information under finite-resolution readout.

Coupling to routing overhead

In the scan–fold framework, sharpening "where" and "when" is not only an algebraic tradeoff (Weyl-pair complementarity), but can also become a compilation tradeoff once the readout instrument and locality primitive are embedded into a constrained substrate. Concretely, any protocol refinement that requires more selective screen-local addressing or denser local couplings increases the required routing dilation/congestion on the hardware graph (Section 7.1.1), hence increases the local overhead field $\kappa(x)$. In the lapse dictionary (Section 7.1), this decreases $N(x) = \kappa_0/\kappa(x)$. In scan-chain realizations this dependence is directly controlled by address-order separations of screen neighbors (Proposition 5.4). A fully quantitative inequality linking a chosen POVM family $\{E_k^{(\varepsilon)}\}_k$ to a lower bound on κ requires specifying an explicit compilation model for implementing that instrument. In this paper we treat this as an auditable interface. Once a concrete instrument family and routing model are fixed, one can compute the induced uncertainty region (Δ_U, Δ_V) and the resulting κ landscape and compare them.

6.7 Readout regularity and bandwidth: controlled approximation of differential operators

To connect coarse-grained closed-layer fields to continuum differential operators (as used in the Poisson template), the readout must suppress sub-lattice scales and provide sufficient regularity. We record a concrete sufficient condition in a standard band-limited setting.

Definition 6.3 (Kernel-form readout and effective bandwidth)**.** *Let the screen be embedded as a periodic box and let $h := 2^{-n}$ denote the lattice spacing at resolution n. We say a readout POVM at resolution ε admits a* kernel form *if the induced coarse-grained scalar field can be written as a convolution*

$$F_\varepsilon = w^{(\varepsilon)} * F, \qquad w^{(\varepsilon)} \geq 0, \quad \int w^{(\varepsilon)} = 1,$$

for a nonnegative normalized kernel $w^{(\varepsilon)}$ of spatial width ε (equivalently, a low-pass transfer function $\widehat{w}^{(\varepsilon)}$). We call K_ε an effective bandwidth *if $\widehat{w}^{(\varepsilon)}(k)$ is negligible for $|k| > K_\varepsilon$ (for example, compactly supported or rapidly decaying), so that F_ε is effectively band-limited to frequencies $|k| \lesssim K_\varepsilon$.*

Proposition 6.4 (Band-limited error bounds for discrete gradients and Laplacians)**.** *Consider a d-dimensional periodic box and a band-limited field f whose Fourier support satisfies $\widehat{f}(k) = 0$ for $|k| > K$. Let ∇_h and Δ_h denote the standard centered finite-difference gradient and Laplacian on the grid of spacing h. Then there exists a dimension-dependent constant $C_d > 0$ such that, at grid points,*

$$\|\nabla_h f - \nabla f\|_{L^2} \;\leq\; C_d\,(Kh)^2\,\|\nabla f\|_{L^2}\,, \qquad \|\Delta_h f - \Delta f\|_{L^2} \;\leq\; C_d\,(Kh)^2\,\|\Delta f\|_{L^2}\,.$$

In particular, if the readout kernel enforces an effective bandwidth $K_\varepsilon \asymp 1/\varepsilon$ (Definition 6.3) and the scale separation $h \ll \varepsilon$ holds, then the discretization error is $O((h/\varepsilon)^2)$ in these norms.

Proof sketch. On a periodic domain, discrete difference operators are Fourier multipliers with symbols that approximate the continuum symbols: for example, in one dimension the centered difference derivative has symbol $\frac{\sin(kh)}{h}$ while the continuum derivative has symbol k. For $|k| \le K$ one has $\left|\frac{\sin(kh)}{h} - k\right| \le c\,(Kh)^2\,|k|$ and similarly for the second derivative symbol, yielding the stated bounds after summing over Fourier modes. □

Remark 6.5 (Interpretation as a POVM regularity condition)**.** *Proposition 6.4 isolates concrete readout requirements: the POVM must act as a low-pass filter at scale ε (bandwidth $K_\varepsilon \lesssim 1/\varepsilon$) and the lattice spacing must satisfy $h \ll \varepsilon$. Under these conditions, coarse-grained closed-layer operators built from discrete gradients/Laplacians approximate their continuum counterparts with a controlled error, so the continuum Poisson template of Section 8.4 becomes quantitatively meaningful.*

7 Computational-lapse gravity: routing overhead, redshift, and a phase-potential Newtonian limit

7.1 Routing overhead and the lapse field

Hilbert folding defines screen locality, but physical implementation depends on how screen-local operations are compiled into scan-nearest-neighbor primitives. To make the "routing overhead" κ precise and falsifiable, we fix a minimal compilation model.

Assumption 7.1 (Hardware graph, primitive gate set, and scheduling)**.** *At resolution n, the observer-accessible degrees of freedom are labeled by screen sites $x \in \Sigma_n$ (Axiom 3.4). Physical primitives are executed on a* hardware interaction graph *$G_{\mathrm{phys}} = (V_{\mathrm{phys}}, E_{\mathrm{phys}})$ equipped with integer edge weights $w(e) \in \mathbb{Z}_{>0}$ interpreted as the number of global scan ticks required to execute a fixed primitive two-body operation supported on $e \in E_{\mathrm{phys}}$. A placement map $\pi_n : \Sigma_n \to V_{\mathrm{phys}}$ assigns each screen site to a physical vertex. In addition to two-body primitives on edges, we allow arbitrary single-site primitives on vertices at unit tick cost (absorbed into the tick definition) and, when needed, bounded-cost measurement/reset primitives as part of the readout interface (Axiom 3.1).*

Scheduling constraint (disjoint-support parallelism). *Time is discretized in global scan ticks $t \in \mathbb{Z}_{\ge 0}$. During any tick, one may execute in parallel any collection of primitives whose supports are vertex-disjoint (equivalently, two-body primitives are executed on a matching of G_{phys}, possibly together with arbitrary single-site primitives on vertices not engaged by a two-body primitive in that tick). If a two-body primitive on an edge e has weight $w(e) > 1$, it occupies its endpoints for $w(e)$ consecutive ticks under this rule.*

Routing primitive. *Whenever a screen-local task requires a two-body interaction between degrees of freedom placed far apart on G_{phys}, routing is implemented by sequences of local swaps/permutations built from the same two-body primitives (standard swap-network compilation).*

Definition 7.2 (Local compilation overhead)**.** *Fix a reference* local update task *$\mathcal{G}_x$ supported in a bounded screen neighborhood of x (e.g. a prescribed set of nearest-neighbor gates on $(\Sigma_n, \sim)$ around x). Let $\mathrm{Depth}_{G_{\mathrm{phys}}}(\mathcal{G}_x)$ denote the minimal number of global scan ticks required to implement $\mathcal{G}_x$ using the primitive gate set and disjoint-support scheduling constraint of Assumption 7.1, together with any intermediate swaps/routing permitted on G_{phys}. Define*

$$\kappa(x) := \mathrm{Depth}_{G_{\mathrm{phys}}}(\mathcal{G}_x).$$

When needed, we write $\mathcal{G} := \{\mathcal{G}_x\}_{x\in\Sigma_n}$ *and* $\kappa(x) = \kappa(x; G_{\text{phys}}, \pi_n, \mathcal{G})$ *to emphasize the dependence on the hardware graph, placement, and task family.*

Definition 7.2 makes $\kappa(x)$ an operationally auditable quantity once G_{phys}, π_n, and the task family $\mathcal{G}_x$ are fixed. It also allows graph-theoretic bounds that connect κ to standard notions of circuit depth and routing congestion.

We define the computational lapse by (12):

$$\mathcal{N}(x) = \frac{\kappa_0}{\kappa(x)}.$$

Proposition 7.3 (Operational derivation of the lapse time-scaling)**.** *Let* $t \in \mathbb{Z}_{\geq 0}$ *denote global scan ticks, and fix a reference overhead* $\kappa_0 > 0$ *corresponding to a homogeneous region in which the local update task completes in* κ_0 *ticks. Define the* local relational time $\tau_{\text{loc}}(t; x)$ *to be proportional to the number of completed local update cycles at* x *by time* t*, normalized so that* $\tau_{\text{loc}}(t; x) = t$ *when* $\kappa(x) = \kappa_0$*. Then, in the continuous notation used throughout the paper,*

$$\mathrm{d}\tau_{\text{loc}} = \mathcal{N}(x)\,\mathrm{d}t = \frac{\kappa_0}{\kappa(x)}\,\mathrm{d}t. \tag{14}$$

Proof. Define the completed-cycle count

$$C_x(t) := \left\lfloor \frac{t}{\kappa(x)} \right\rfloor,$$

which is an operationally auditable integer derived from the compilation depth $\kappa(x)$. By Definition 7.2, one local update cycle at x requires $\kappa(x)$ global ticks, so $C_x(t)$ is exactly the number of completed local cycles by tick t. Define relational time by

$$\tau_{\text{loc}}(t; x) := \kappa_0\, C_x(t). \tag{15}$$

Then $\tau_{\text{loc}}(t; x) = t$ when $\kappa(x) = \kappa_0$, since $C_x(t) = \lfloor t/\kappa_0 \rfloor$ and coarse-grained time ignores the $O(1)$ endpoint discrepancy. Moreover,

$$0 \;\leq\; \frac{\kappa_0}{\kappa(x)}\, t \;-\; \tau_{\text{loc}}(t; x) \;<\; \kappa_0,$$

so

$$\frac{\tau_{\text{loc}}(t; x)}{t} = \frac{\kappa_0}{\kappa(x)} + O\!\left(\frac{1}{t}\right) \qquad (t \to \infty).$$

Passing to the continuous notation (coarse-grained in t) yields (14). □

Remark 7.4 (Finite-horizon redshift test and quantitative error bound)**.** *The lapse ratio prediction* (16) *is directly testable by counting completed cycles. For two sites* x_1, x_2 *and horizon* t*, define the measured ratio*

$$R_{\text{meas}}(t) := \frac{\tau_{\text{loc}}(t; x_1)}{\tau_{\text{loc}}(t; x_2)} = \frac{C_{x_1}(t)}{C_{x_2}(t)}.$$

Whenever $t > \kappa(x_2)$ *one has the deterministic bounds*

$$\frac{\frac{t}{\kappa(x_1)} - 1}{\frac{t}{\kappa(x_2)}} \;\leq\; R_{\text{meas}}(t) \;\leq\; \frac{\frac{t}{\kappa(x_1)}}{\frac{t}{\kappa(x_2)} - 1},$$

so $R_{\text{meas}}(t) \to \kappa(x_2)/\kappa(x_1)$ *with an explicit* $O(1/t)$ *finite-horizon error controlled by* $\kappa(x_1), \kappa(x_2)$*. This is the discrete (auditable) counterpart of the differential statement* $\mathrm{d}\tau_1/\mathrm{d}\tau_2 = \kappa(x_2)/\kappa(x_1)$*.*

Equation (14) is therefore not an external relativistic postulate: it is a closed-layer consequence of (i) the compilation model and (ii) the definition of relational time as auditable progress of a standard local update task. In standard relativity language, it matches the role of the lapse function in a static slicing with vanishing shift [36–38].

Static-metric dictionary. In a static spacetime with vanishing shift, the line element can be written as

$$\mathrm{d}s^2 = -N(x)^2\, c^2\, \mathrm{d}t^2 + h_{ij}(x)\, \mathrm{d}x^i \mathrm{d}x^j,$$

so for a stationary worldline ($\mathrm{d}x^i = 0$) one has $\mathrm{d}\tau = N(x)\, \mathrm{d}t$. Thus, (14) identifies the GR lapse N with the operational field $\mathcal{N}$, up to the calibration that turns scan ticks into coordinate time.

Remark 7.5 (Calibration and reparameterization)**.** *The global tick parameter t is a protocol coordinate. Rescaling $t \mapsto t' := a\, t$ (or, more generally, applying a monotone reparameterization) rescales the lapse by $N'(x) = N(x)\, \mathrm{d}t/\mathrm{d}t'$. All closed-layer observables used below are formulated either in terms of the relational time τ_{loc} itself or in terms of lapse ratios (redshifts), which are invariant under such coordinate reparameterizations. To attach physical units, one fixes a calibration map from ticks to coordinate time, e.g. $t_{\mathrm{phys}} := \Delta t\, t$ where Δt is the duration (in seconds) of a chosen primitive reference tick in the substrate, or $\Delta t := \tau_0$ in a time-delay realization (Appendix F). The reference overhead κ_0 sets the zero of the potential $\Phi = \log(\kappa/\kappa_0)$ and can be fixed by choosing a homogeneous reference region (or reference energy band) in which the standard local task completes in κ_0 ticks.*

Remark 7.6 (Micro-to-macro map and scope)**.** *To avoid ambiguity about what is postulated and what is derived, we summarize the dictionary used in this section.*

- ***Auditable micro input.*** *One fixes (i) a hardware graph (G_{phys}, w) and placement π_n (Assumption 7.1); (ii) a bounded-local task family $\mathcal{G} = \{\mathcal{G}_x\}$ (Definition 7.2); and (iii) calibration choices such as $(\kappa_0, \Delta t)$ or (κ_0, τ_0) (Remark 7.5).*
- ***Derived closed-layer observables.*** *Given (i)–(iii), the overhead field κ induces the lapse field $\mathcal{N} = \kappa_0/\kappa$, hence the relational time scaling* (14)*, redshift ratios* (16)*, and the potential $\Phi = -\log \mathcal{N}$ (Definition 7.11).*
- ***Minimal macroscopic closure (Newtonian-limit template).*** *To connect Φ to forces, one additionally specifies (iv) an auditable defect/source interface σ at a chosen horizon and readout scale, with a boundary convention, and (v) a macroscopic coupling G fixed by calibration (Section 8.4 and Remark 8.17). Given (iv)–(v), the Dirichlet principle yields Poisson closure for Φ, reproducing the weak-field template of GR at leading order (Remark 8.18).*
- ***Limitations.*** *This paper does not derive the full Einstein dynamics from the microscopic compilation model. Beyond the static weak-field template, additional constitutive input is required to relate defect currents to protocol fields and to determine an effective shift/spatial-metric evolution; Section 7.6 records the corresponding constraint-level templates.*

Proposition 7.7 (Clock-task equivalence does not change the lapse field)**.** *Let $\mathcal{G}_x$ and $\mathcal{G}'_x$ be two bounded-local update task families on the same screen neighborhood such that there exists a constant $C > 0$ with*

$$\mathrm{Depth}_{G_{\mathrm{phys}}}(\mathcal{G}'_x) = C\, \mathrm{Depth}_{G_{\mathrm{phys}}}(\mathcal{G}_x) \qquad \text{for all } x \in \Sigma_n.$$

Let κ, κ' be the corresponding overhead fields and choose $\kappa'_0 := C\, \kappa_0$. Then the associated lapse fields coincide: $\mathcal{N}'(x) = \mathcal{N}(x)$ for all x, and hence all redshift ratios are unchanged.

Proof. By definition, $\kappa'(x) = C\, \kappa(x)$ for all x. Therefore $\mathcal{N}'(x) = \kappa'_0/\kappa'(x) = (C\kappa_0)/(C\kappa(x)) = \kappa_0/\kappa(x) = \mathcal{N}(x)$. □

7.1.1 Graph-theoretic bounds for κ

Definition 7.2 admits immediate lower bounds in terms of *dilation* (distance expansion) and *congestion* (edge load) of any routing realizing the required screen-local couplings on the hardware graph. These are standard notions in network routing and parallel compilation.

Definition 7.8 (Dilation and congestion (routing complexity measures))**.** *Let*

$$E_{\text{screen}} := \{\{x, y\} \subset \Sigma_n : \ x \sim y\}$$

denote the nearest-neighbor edge set of the screen lattice. Consider a collection of required screen-local interactions (edges) $\mathcal{E} \subseteq E_{\text{screen}}$ to be implemented during a local update. A routing *assigns to each $\{x, y\} \in \mathcal{E}$ a path P_{xy} in G_{phys} connecting $\pi_n(x)$ to $\pi_n(y)$. The* dilation *of the routing is*

$$\text{dil}(\mathcal{E}) := \max_{\{x,y\}\in\mathcal{E}} \sum_{e\in P_{xy}} w(e),$$

and the congestion *is*

$$\text{cong}(\mathcal{E}) := \max_{e\in E_{\text{phys}}} \sum_{\{x,y\}\in\mathcal{E}: e\in P_{xy}} 1,$$

interpreted as the maximum number of routed interactions whose chosen paths traverse a given hardware edge.

Proposition 7.9 (Universal lower bounds)**.** *For any local update task whose required interaction set is $\mathcal{E}$ one has*

$$\kappa(x) \ \geq \ \text{dil}(\mathcal{E}), \qquad \kappa(x) \ \geq \ \text{cong}(\mathcal{E}),$$

up to fixed constant factors determined by the primitive gate set and scheduling convention.

Proof. No implementation can transmit information between $\pi_n(x)$ and $\pi_n(y)$ faster than the weighted shortest-path distance, yielding the dilation bound. Each tick uses any hardware edge at most once (parallel disjoint-edge constraint), so an edge traversed by k routed interactions forces at least k time slices, yielding the congestion bound. □

Proposition 7.10 (Existence of schedules with depth $O(\text{cong} + \text{dil})$)**.** *Under standard store-and-forward routing/scheduling models on graphs, there exist randomized schedules that realize all interactions in $\mathcal{E}$ within*

$$O(\text{cong}(\mathcal{E}) + \text{dil}(\mathcal{E}))$$

ticks, with high probability, and corresponding deterministic bounds with additional polylogarithmic overhead; see [39].

Propositions 7.9–7.10 provide a principled map from the protocol field $\kappa(x)$ to well-defined graph-theoretic complexity measures with provable bounds. They also identify what must be specified to make κ experimentally meaningful: the hardware graph, the placement map, and the local interaction task family.

Scan-chain specialization (explicit κ in terms of address separation). The simplest substrate model is the one-dimensional scan chain, in which G_{phys} is the path graph on vertices $\{0, 1, \ldots, 2^{dn} - 1\}$ with unit weights $w(e) \equiv 1$ and primitive operations allowed only between adjacent ticks. In this case, a natural placement is $\pi_n(x) := H_n^{-1}(x)$ (Section 4.1). For a single required interaction between screen neighbors $\{x, y\} \in E_{\text{screen}}$, any implementation by adjacent swaps satisfies the tight bound

$$\text{Depth}_{\text{chain}}(\{x, y\}) \ \geq \ \left|H_n^{-1}(x) - H_n^{-1}(y)\right|,$$

since information cannot traverse faster than one edge per tick on a path. Conversely, swapping along the unique shortest path on the chain yields an implementation within $O(\Delta_{xy})$ ticks, where $\Delta_{xy} := |H_n^{-1}(x) - H_n^{-1}(y)|$. Thus, in the scan-chain model, the lapse profile is quantitatively controlled by address-order separation of screen neighbors, making the address map choice directly testable.

Canonical architecture scalings. As an order-of-magnitude guide, Definition 7.2 supports immediate scaling estimates on standard graphs:

- **Matched lattice (near-constant lapse).** If G_{phys} contains a copy of the screen adjacency graph with unit weights and π_n respects that adjacency locally (identity placement at scale ε), then the local update task $\mathcal{G}_x$ can be scheduled in $O(1)$ depth by a constant-round edge-coloring/matching decomposition, hence $\kappa(x) = O(1)$ and $\mathcal{N}$ is approximately uniform.
- **Hypercube (logarithmic diameter).** If G_{phys} is a hypercube on $|V_{\text{phys}}| = 2^{dn}$ vertices, its diameter is dn, so any two sites can be made adjacent via a path of length at most dn. For bounded-degree local tasks this yields the conservative estimate $\kappa(x) = O(dn)$, i.e. logarithmic in the total number of sites.
- **Scan chain (maximal distortion).** On the path graph, the diameter is $2^{dn} - 1$, so neighbor pairs in the screen graph that are far in scan order yield κ of order their index separation, producing exponentially large overheads in the worst case.

These estimates clarify what the lapse field is sensitive to: the interplay between the required screen-local interaction pattern and the geometric/graph-theoretic constraints of the underlying primitive substrate.

Time-delay interface for a measurable overhead proxy. A practical way to interpret and measure routing overhead is as an operational delay. If an experiment provides a local total delay $\tau_{\text{WS}}(E)$ (e.g. via Wigner–Smith time delay at energy E), then a dimensionless overhead proxy can be defined by

$$\kappa_{\text{WS}}(E) := \frac{\tau_{\text{WS}}(E)}{\tau_0}, \qquad \mathcal{N}_{\text{WS}}(E) := \frac{\kappa_0}{\kappa_{\text{WS}}(E)},$$

for a chosen reference tick duration τ_0. This energy/scale-dependent proxy is not the same object as the spatial fields $\kappa(x)$ and $\mathcal{N}(x)$ defined by a compilation task family; see Appendix F.

7.2 Redshift as an overhead ratio

Given two locations x_1, x_2, (14) yields the redshift ratio

$$\frac{\mathrm{d}\tau_1}{\mathrm{d}\tau_2} = \frac{\mathcal{N}(x_1)}{\mathcal{N}(x_2)} = \frac{\kappa(x_2)}{\kappa(x_1)}. \tag{16}$$

Thus, inhomogeneous compilation cost is directly observable as a relative slowing of local relational time. This provides a concrete operational semantics for "gravitational time dilation" within the scan–fold–readout framework [40, 41].

7.3 Computational potential and a GR matching dictionary

Definition 7.11 (Computational potential)**.** *Define the computational potential*

$$\Phi(x) := -\log \mathcal{N}(x) = \log\!\left(\frac{\kappa(x)}{\kappa_0}\right). \tag{17}$$

The potential Φ is directly measurable once κ is operationally specified, and it converts multiplicative redshift into an additive field:

$$\frac{\mathcal{N}(x_1)}{\mathcal{N}(x_2)} = \mathrm{e}^{-(\Phi(x_1)-\Phi(x_2))}.$$

Schwarzschild matching (static vacuum template). In classical GR, for a static spherically symmetric vacuum solution in Schwarzschild coordinates, the lapse is

$$N_{\mathrm{Schw}}(r) = \sqrt{1 - \frac{2GM}{c^2 r}},$$

so the corresponding overhead profile in the present dictionary is

$$\kappa_{\mathrm{Schw}}(r) = \frac{\kappa_0}{N_{\mathrm{Schw}}(r)}.$$

This provides a direct *fitting interface*: any measured redshift profile $\mathcal{N}(r)$ can be converted into an inferred $\kappa(r)$, and compared against the Schwarzschild template at the level of a single parameter M [36, 37, 42].

Weak-field expansion. For $r \gg 2GM/c^2$, one has

$$\Phi_{\mathrm{Schw}}(r) = -\log N_{\mathrm{Schw}}(r) = -\frac{1}{2}\log\left(1 - \frac{2GM}{c^2 r}\right) = \frac{GM}{c^2 r} + O\left(\frac{G^2M^2}{c^4 r^2}\right),$$

so $\mathcal{N}(r) = \mathrm{e}^{-\Phi(r)}$ reproduces the standard first-order gravitational redshift/time-dilation scaling.

Closed-form one-parameter fit. Given two radii r_1, r_2 and a measured lapse ratio $R := \mathcal{N}(r_1)/\mathcal{N}(r_2)$, the Schwarzschild formula yields

$$R^2 = \frac{1 - \frac{2GM}{c^2 r_1}}{1 - \frac{2GM}{c^2 r_2}},$$

so (provided $1/r_1 \neq R^2/r_2$) one obtains

$$M = \frac{c^2(1 - R^2)}{2G\left(\frac{1}{r_1} - \frac{R^2}{r_2}\right)}. \tag{18}$$

7.4 Calibration example: Earth-to-GPS gravitational redshift

In the weak-field regime, the relative clock-rate difference between two stationary radii $r_1 < r_2$ is

$$\frac{\mathrm{d}\tau(r_2)}{\mathrm{d}\tau(r_1)} - 1 \;\approx\; \frac{GM}{c^2}\left(\frac{1}{r_1} - \frac{1}{r_2}\right). \tag{19}$$

Conversely, a measured rate difference immediately yields a one-parameter mass extraction in this approximation:

$$M \;\approx\; \frac{c^2}{G}\,\frac{\left(\frac{\mathrm{d}\tau(r_2)}{\mathrm{d}\tau(r_1)} - 1\right)}{\left(\frac{1}{r_1} - \frac{1}{r_2}\right)},$$

consistent with the exact inversion (18) when the redshift is not linearized. For Earth, taking $GM \simeq 3.986 \times 10^{14}\,\mathrm{m}^3/\mathrm{s}^2$, $c = 299\,792\,458\,\mathrm{m/s}$, $r_1 \simeq 6.371 \times 10^6\,\mathrm{m}$ and $r_2 \simeq r_1 + 2.02 \times 10^7\,\mathrm{m}$ (GPS altitude), (19) predicts a gravitational rate increase of

$$\Delta\tau \approx 45.7\ \mu\mathrm{s/day},$$

consistent with the standard GPS gravitational redshift budget (before including the kinematic special-relativistic correction) [43].

7.5 Discrepancy as a source and a Poisson closure for a phase potential

To connect lapse and forces in a minimal way, we introduce a coarse-grained *mismatch density* $\sigma(x)$, intended to be an auditable proxy derived from discrepancy/regularization budgets (Section 8.4). At the level of effective fields, the canonical closure is a Poisson equation for the scalar phase potential Φ:

$$-\Delta\Phi = 4\pi G\,\sigma, \tag{20}$$

where G is Newton's constant (or, more generally, the macroscopic coupling constant fixed by calibration). With $\sigma = \rho/c^2$ (mass-energy density divided by c^2), (20) is equivalent to the standard Newtonian form for $\phi_N := -c^2\Phi$:

$$\Delta\phi_N = 4\pi G\,\rho.$$

The associated phase-pressure (force) field is

$$\mathbf{P}_\Phi := -\nabla\Phi. \tag{21}$$

In these conventions, the weak-field gravitational acceleration template is

$$\mathbf{a} = -\nabla\phi_N = c^2\nabla\Phi = -c^2\,\mathbf{P}_\Phi.$$

Newtonian template. In $\mathbb{R}^3$, the Green function of $-\Delta$ implies that a localized source produces

$$\Phi(r) \approx \Phi_0 + \frac{GM}{c^2 r}, \tag{22}$$

and therefore an inverse-square acceleration template $|\mathbf{a}| \sim GM/r^2$. Section 9 includes a reproducible FFT-based Poisson solver demonstrating the emergence of an approximate $1/r$ profile on a periodic grid.

Remarks on closure. Equation (20) follows as an Euler–Lagrange equation both on the discrete screen graph (Proposition 8.13) and in the continuum (Proposition 8.15). It also matches the covariant weak-field limit of Einstein's equations (Remark 8.18), and therefore serves as the standard Newtonian-limit template for static weak fields [36–38, 42, 44]. In the present framework, its closed-layer role is to provide a quantitative and falsifiable bridge: given a protocol-defined source density σ (from discrepancy/defect budgets) one computes Φ and predicts redshift and force profiles via (17) and (21). For extended discussions and variational embeddings within the HPA–Ω program, see [40, 45].

7.6 Beyond static lapse: drift/shift, defect currents, and constraint templates

The closed-layer constructions above are formulated in a static slicing with vanishing shift, which suffices for the auditable redshift dictionary and the Newtonian-limit Poisson template. For time-dependent protocol flows, the defect-density interface already provides the minimal conservation structure: a source $\sigma(t,x)$ and a defect current $J(t,x)$ satisfying the continuity equation $\partial_t\sigma + \nabla\cdot J = 0$ (Definition 8.11).

In GR language, a general ADM line element reads

$$\mathrm{d}s^2 = -N^2c^2\,\mathrm{d}t^2 + h_{ij}\big(\mathrm{d}x^i + N^i\mathrm{d}t\big)\big(\mathrm{d}x^j + N^j\mathrm{d}t\big),$$

where the lapse N and shift N^i act as Lagrange multipliers enforcing the Hamiltonian and momentum constraints. In the present dictionary, N continues to be identified with the local cycle-completion rate $\mathcal{N} = \kappa_0/\kappa$. A nontrivial effective shift is naturally interpreted as a protocol-level drift of screen-local degrees of freedom induced by compilation schedules and

routing, i.e. by systematic transport of defect budgets across the screen. In this sense, routing overhead contributes not only to the scalar lapse landscape but also to transport costs controlling feasible defect currents.

A fully closed dynamic model requires one additional input beyond the static Poisson closure: a constitutive principle relating J to the protocol fields (or, equivalently, a dynamical penalty/constraint for transport on the hardware graph). Once such an input is fixed, the continuity constraint and the static Dirichlet principle (Section 8.4) provide a canonical route to a time-dependent effective geometry: one enforces charge conservation at the protocol level while determining $\Phi(t,\cdot)$ at each time slice from the instantaneous defect density by a variational principle.

ADM constraint templates (for orientation). On each time slice, the ADM constraints relate intrinsic geometry, extrinsic curvature, and matter sources. In standard notation, they can be written schematically as

$$R^{(3)} + K^2 - K_{ij}K^{ij} \;=\; \frac{16\pi G}{c^4}\, T_{\mu\nu} n^\mu n^\nu, \tag{23}$$

$$D_j\Big(K^{ij} - h^{ij}K\Big) \;=\; \frac{8\pi G}{c^4}\, T_{\mu\nu} n^\mu h^{\nu i}, \tag{24}$$

where $R^{(3)}$ is the scalar curvature of h_{ij}, K_{ij} is the extrinsic curvature, D is the Levi–Civita connection of h_{ij}, and n^μ is the unit normal to the slice [36–38]. The static Poisson closure used above corresponds to the weak-field, slow-motion regime in which the shift is negligible, $K_{ij} \approx 0$, and $T_{00} \approx \rho c^2$, so that (23) reduces at leading order to $\Delta\phi_N = 4\pi G\rho$ (Remark 8.18).

Momentum-constraint analogue and open input. In the protocol language, the defect current $J(t,x)$ (Definition 8.11) is the minimal conserved flux accompanying the source $\sigma(t,x)$. To match the spirit of (24) beyond the static template, one must provide an additional constitutive or variational input that couples a shift/drift field (or an extrinsic-curvature surrogate) to the conserved current. This paper does not fix such an input; rather, it isolates the point at which new physics must enter to go beyond the scalar Poisson closure.

Limitations and non-claims. The present section provides an operational definition of the lapse field and a calibrated Newtonian-limit template once a defect/source interface is specified. It does not establish a micro-to-macro derivation of full Einstein dynamics, the full ADM constraint algebra, or a unique time-dependent closure for (h_{ij}, N^i) from compilation/routing microphysics.

8 Minimal dynamical closure: least-discrepancy flow and an auditable arrow of time

8.1 Accumulated mismatch as a Lyapunov candidate

For a scan prefix P_N, define the accumulated mismatch

$$E_N := N\, D_N^*(P_N).$$

For bounded-type irrational scans (notably the golden branch), discrepancy certificates imply $E_N = O(\log N)$, so mismatch grows slowly and remains auditable at large horizons. In contrast, protocols that effectively introduce exponential-gain modes (Appendix D) destroy Abel admissibility and drive E_N beyond sustainable bounds.

Remark 8.1 (Sensitivity away from the golden branch)**.** *The mismatch growth rate is quantitatively controlled by the continued-fraction data of the scan slope α. For Kronecker scans, standard certificates bound $D_N^*(P_N)$ in terms of partial quotients (Appendix B); bounded-type slopes yield the logarithmic $O((\log N)/N)$ rate, while large partial quotients produce bursts of irregularity at intermediate horizons. When the defect density σ is constructed from occupation bias (Example 8.8), such discrepancy bursts translate into source perturbations at the readout scale. Propagation to effective geometry is then controlled by the Poisson stability estimate (Proposition 5.7): coarse-grained perturbations of σ induce proportionate perturbations of $\nabla\Phi$ (hence of lapse ratios) in Sobolev norms.*

This motivates a minimal closed-layer arrow-of-time statement: in the space of admissible protocols, sustainable evolution must flow toward configurations that keep mismatch growth controlled under fixed resource budgets.

8.2 A least-discrepancy variational template

Let $\mathcal{P}$ denote protocol parameters (scan slope, readout resolution, address map selection within the admissible family, routing scheme, truncation budget, etc.). A minimal auditable cost functional at horizon T takes the schematic form

$$\mathcal{E}_T(\mathcal{P}) = \mathrm{CertDisc}_T(\mathcal{P}) \;+\; \lambda\,\mathrm{Cost}(\mathcal{P}) \;+\; \mu\,\mathrm{RegTail}_T(\mathcal{P}), \tag{25}$$

where:

- $\mathrm{CertDisc}_T$ is a deterministic discrepancy-based certificate (e.g. $\mathrm{Var}(f)\,D_N^*$);
- Cost is an implementation overhead cost (routing depth, memory, circuit size), including the lapse profile $\mathcal{N}$;
- $\mathrm{RegTail}_T$ controls truncation/regularization tail errors under the Abel-first convention.

One may then define dynamics as a (possibly damped) gradient flow on protocol space decreasing $\mathcal{E}_T$. In the broader HPA–Ω program, this template is elevated to an Ω action principle with explicit Fisher-information and gravitational terms [40]. For the present paper, (25) serves only as the minimal closed-layer bridge between discrepancy certificates, routing overhead, and effective gravitational templates.

8.3 An explicit least-discrepancy evolution and an H-theorem

To answer the "minimal deviation" question quantitatively, we record an explicit evolution equation that is sufficient for a closed-layer monotonicity statement.

Definition 8.2 (Least-discrepancy gradient flow (continuous time))**.** *Assume the protocol space is modeled as an open subset $\mathcal{U} \subset \mathbb{R}^m$ with coordinate vector $\mathcal{P} \in \mathcal{U}$, and assume $\mathcal{E}_T : \mathcal{U} \to \mathbb{R}$ is differentiable. Fix a* mobility *(preconditioner) field $M(\mathcal{P}) \in \mathbb{R}^{m\times m}$ such that $M(\mathcal{P})$ is symmetric and positive semidefinite for all $\mathcal{P}$. The least-discrepancy evolution is the gradient flow*

$$\frac{\mathrm{d}\mathcal{P}}{\mathrm{d}\tau} = -\,M(\mathcal{P})\,\nabla_{\mathcal{P}}\mathcal{E}_T(\mathcal{P}), \tag{26}$$

where τ is an auxiliary (protocol) time parameter indexing updates in protocol space.

Proposition 8.3 (H-theorem (continuous flow))**.** *Let $\mathcal{P}(\tau)$ satisfy* (26) *and suppose $\mathcal{E}_T$ is continuously differentiable along the trajectory. Then the "entropy" functional $H(\tau) := \mathcal{E}_T(\mathcal{P}(\tau))$ is nonincreasing:*

$$\frac{\mathrm{d}H}{\mathrm{d}\tau} = -\,\langle \nabla_{\mathcal{P}}\mathcal{E}_T(\mathcal{P}),\, M(\mathcal{P})\,\nabla_{\mathcal{P}}\mathcal{E}_T(\mathcal{P})\rangle \;\le\; 0.$$

Moreover, $\mathrm{d}H/\mathrm{d}\tau = 0$ holds if and only if $M(\mathcal{P})\,\nabla_{\mathcal{P}}\mathcal{E}_T(\mathcal{P}) = 0$ (a stationary protocol under the chosen mobility).

Proof. Differentiate $H(\tau) = \mathcal{E}_T(\mathcal{P}(\tau))$ and use (26):

$$\frac{\mathrm{d}H}{\mathrm{d}\tau} = \left\langle \nabla_{\mathcal{P}} \mathcal{E}_T(\mathcal{P}),\, \frac{\mathrm{d}\mathcal{P}}{\mathrm{d}\tau} \right\rangle = -\left\langle \nabla_{\mathcal{P}} \mathcal{E}_T(\mathcal{P}),\, M(\mathcal{P})\, \nabla_{\mathcal{P}} \mathcal{E}_T(\mathcal{P}) \right\rangle \le 0,$$

since $M(\mathcal{P})$ is positive semidefinite. □

Definition 8.4 (Auditable discrete-time update (proximal/gradient step))**.** *In practice the protocol is updated in discrete steps $k \in \mathbb{Z}_{\ge 0}$. A minimal auditable update rule compatible with* (26) *is*

$$\mathcal{P}_{k+1} = \mathcal{P}_k - \eta_k\, M_k\, \nabla_{\mathcal{P}} \mathcal{E}_T(\mathcal{P}_k), \tag{27}$$

where $\eta_k > 0$ is a step size and $M_k \succeq 0$ is a chosen preconditioner (for example, a diagonal scaling reflecting heterogeneous audit costs of changing different protocol knobs).

Proposition 8.5 (Sufficient condition for monotonic decrease (discrete time))**.** *Assume $\mathcal{E}_T$ has an L-Lipschitz gradient on a convex subset containing the iterates (i.e. $\mathcal{E}_T$ is L-smooth). If $M_k = I$ and $0 < \eta_k \le 1/L$, then the discrete update* (27) *satisfies*

$$\mathcal{E}_T(\mathcal{P}_{k+1}) \le \mathcal{E}_T(\mathcal{P}_k)$$

for every step k. More generally, for symmetric $M_k \succeq 0$, the same conclusion holds whenever $\eta_k \|M_k\|_2 \le 1/L$.

Proof. For an L-smooth function, the standard descent lemma gives

$$\mathcal{E}_T(\mathcal{P}_{k+1}) \le \mathcal{E}_T(\mathcal{P}_k) \;+\; \langle \nabla \mathcal{E}_T(\mathcal{P}_k),\, \mathcal{P}_{k+1} - \mathcal{P}_k \rangle \;+\; \frac{L}{2} \|\mathcal{P}_{k+1} - \mathcal{P}_k\|_2^2.$$

Substituting $\mathcal{P}_{k+1} - \mathcal{P}_k = -\eta_k M_k \nabla \mathcal{E}_T(\mathcal{P}_k)$ yields a negative quadratic form whenever $\eta_k \|M_k\|_2 \le 1/L$. □

Remark 8.6 (When the H-theorem can fail)**.** *Monotonicity is guaranteed only under regularity and step-size control (Propositions 8.3–8.5). In particular, $\mathcal{E}_T$ is generally nonconvex in realistic protocol spaces, and large steps or poorly conditioned preconditioners can increase $\mathcal{E}_T$ transiently. This is not a contradiction: it simply means the protocol update rule is itself part of the auditable specification of dynamics.*

8.4 From auditable defect budgets to a Poisson closure

Section 7 uses a phase potential $\Phi = -\log \mathcal{N}$ and a Poisson closure as the Newtonian-limit template. Here we record a closed-layer derivation of this closure from a minimal energy principle.

Scope of the closure. The purpose of this subsection is to provide a minimal and auditable route from protocol-level defect budgets to an effective *scalar* potential. It should be read as a weak-field/static (Hamiltonian-constraint) template: once a source convention and boundary condition are fixed, Poisson closure follows from the Dirichlet principle. A full time-dependent recovery of the ADM system requires additional constitutive input coupling conserved defect currents to an effective shift/drift sector (Section 7.6).

8.4.1 Defect charge measures and continuity constraints

To connect discrepancy/regularization budgets to a field source, it is convenient to treat mismatch as a signed "defect charge" distributed over the screen.

Definition 8.7 (Defect charge and source density)**.** *Let $\Omega \subset \mathbb{R}^3$ denote a coarse-grained region of the screen continuum limit. A protocol at horizon T induces an* auditable defect charge measure *Σ_T on Ω such that $\Sigma_T(B)$ equals the net mismatch budget assigned to a measurable region $B \subseteq \Omega$. Assume Σ_T is finite and absolutely continuous with respect to volume measure, so that there exists a density $\sigma_T \in L^1(\Omega)$ with*

$$\Sigma_T(B) = \int_B \sigma_T(x)\, \mathrm{d}^3x.$$

In the static closure, we write $\sigma := \sigma_T$ at a fixed horizon and suppress T.

Example 8.8 (A canonical auditable choice: coarse-grained occupation bias)**.** *At resolution n, let $x_t^{(n)} \in \Sigma_n$ denote the addressed site at tick t and let*

$$\nu_T^{(n)} := \frac{1}{T} \sum_{t=0}^{T-1} \delta_{x_t^{(n)}}$$

be the empirical occupation measure on Σ_n up to horizon T. Let $u_n := |\Sigma_n|^{-1} \sum_{x \in \Sigma_n} \delta_x$ be the uniform reference measure and let $w^{(\varepsilon)}$ be a nonnegative readout kernel of width ε used to coarse-grain lattice data into a continuum density. Then a concrete auditable defect measure is obtained by smoothing the signed occupation bias $\nu_T^{(n)} - u_n$ and scaling by a calibration constant:

$$\Sigma_T := \gamma_\sigma \left(w^{(\varepsilon)} * (\nu_T^{(n)} - u_n)\right), \qquad \sigma_T = \frac{\mathrm{d}\Sigma_T}{\mathrm{d}^3x}.$$

On a periodic box, this construction automatically yields $\int_\Omega \sigma_T \, \mathrm{d}^3x = 0$ (zero mean), matching the solvability condition of Remark 8.10.

Remark 8.9 (A discrete implementable source at resolution n)**.** *Let $N_T^{(n)}(x) := \#\{0 \le t < T : x_t^{(n)} = x\}$ be the occupation count at site $x \in \Sigma_n$. The signed occupation bias*

$$b_T^{(n)}(x) := \frac{N_T^{(n)}(x)}{T} - \frac{1}{|\Sigma_n|}$$

satisfies $\sum_{x \in \Sigma_n} b_T^{(n)}(x) = 0$. A concrete discrete source for the graph Poisson equation (29) is then

$$\sigma_n(x) := \gamma_\sigma \, b_T^{(n)}(x),$$

and the continuum source density σ may be obtained by smoothing σ_n at readout scale ε.

Remark 8.10 (Conservation/solvability constraints)**.** *If Ω is treated as periodic (as in the FFT Poisson experiment), solvability of $-\Delta\Phi = 4\pi G\sigma$ requires the zero-mean constraint $\int_\Omega \sigma \, \mathrm{d}^3x = 0$, corresponding to fixing the Laplacian zero mode. On non-compact domains with decay at infinity, or on bounded domains with Dirichlet boundary conditions, no such global neutrality is required.*

For time-dependent protocol flows, one may encode conservation of auditable defect charge by a continuity equation.

Definition 8.11 (Continuity equation (dynamic defect conservation))**.** *Let $\sigma(t, x)$ be a time-dependent source density and let $J(t, x)$ be a defect current. We say that defect charge is conserved if, in the distributional sense on $(0, T) \times \Omega$,*

$$\partial_t \sigma + \nabla \cdot J = 0.$$

8.4.2 Dirichlet principle and the Poisson equation

We now isolate the minimal variational structure that yields Poisson closure.

Discrete screen formulation (graph Dirichlet principle). At finite resolution n, the screen lattice $(\Sigma_n, \sim)$ is a finite graph with nearest-neighbor edge set E_{screen} (Definition 7.8). A discrete source $\sigma_n : \Sigma_n \to \mathbb{R}$ may be taken as the coarse-grained defect charge assigned to each site (Definition 8.7), with the neutrality constraint $\sum_{x\in\Sigma_n} \sigma_n(x) = 0$ in the periodic convention (Remark 8.10).

Definition 8.12 (Screen graph Laplacian)**.** *For a function $\Phi : \Sigma_n \to \mathbb{R}$, define the (unscaled) graph Laplacian*

$$(\Delta_{\Sigma_n}\Phi)(x) := \sum_{y\in\Sigma_n:\, y\sim x} (\Phi(y) - \Phi(x)).$$

Proposition 8.13 (Discrete Dirichlet principle and graph Poisson equation)**.** *Fix $G > 0$ and a discrete source $\sigma_n : \Sigma_n \to \mathbb{R}$. Consider the discrete Dirichlet functional on potentials $\Phi : \Sigma_n \to \mathbb{R}$,*

$$\mathcal{J}_n[\Phi] := \sum_{\{x,y\}\in E_{\text{screen}}} \frac{1}{8\pi G} \big(\Phi(x) - \Phi(y)\big)^2 \; - \sum_{x\in\Sigma_n} \sigma_n(x)\, \Phi(x), \tag{28}$$

with boundary convention (periodic/Dirichlet) fixed. If Φ is a stationary point of $\mathcal{J}_n$ under arbitrary variations $\eta : \Sigma_n \to \mathbb{R}$ compatible with the boundary convention, then Φ satisfies the graph Poisson equation

$$-\Delta_{\Sigma_n}\Phi = 4\pi G\, \sigma_n. \tag{29}$$

Proof. Let $\Phi_\epsilon = \Phi + \epsilon\eta$. Differentiating (28) at $\epsilon = 0$ gives

$$0 = \frac{\mathrm{d}}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} \mathcal{J}_n[\Phi_\epsilon] = \sum_{x\in\Sigma_n} \eta(x) \left(\frac{1}{4\pi G} \sum_{y\sim x} \big(\Phi(x) - \Phi(y)\big) - \sigma_n(x) \right),$$

since each undirected edge contributes to exactly its two endpoints. Because η is arbitrary, the bracket must vanish for each x, which is equivalent to (29) using Definition 8.12. □

Remark 8.14 (Continuum limit and calibration)**.** *On a regular grid of spacing $h = 2^{-n}$, the standard scaled discrete Laplacian is $h^{-2}\Delta_{\Sigma_n}$, which converges to the continuum Laplacian under smoothness/coarse-graining assumptions. In the present framework, the overall coefficient G is fixed by calibration at finite resolution (Remark 8.17), so the discrete and continuum conventions may be treated uniformly once a boundary convention and scale map are specified.*

Proposition 8.15 (Dirichlet functional and Poisson Euler–Lagrange equation)**.** *Fix a coupling constant $G > 0$. Let $\Omega \subset \mathbb{R}^3$ be a domain and let $\sigma \in L^1(\Omega)$ be a source density (Definition 8.7). Consider the functional on admissible potentials Φ with suitable boundary conditions,*

$$\mathcal{J}[\Phi] := \int_\Omega \left(\frac{1}{8\pi G}\, |\nabla\Phi(x)|^2 - \sigma(x)\, \Phi(x) \right) \mathrm{d}^3x. \tag{30}$$

If Φ is a stationary point of $\mathcal{J}$ under compactly supported variations (or variations compatible with the boundary condition), then Φ satisfies the Poisson equation

$$-\Delta\Phi = 4\pi G\, \sigma \tag{31}$$

in the weak sense; if Φ is sufficiently smooth, then (31) *holds pointwise.*

Proof. Let η be a smooth test function compatible with the boundary condition and consider $\Phi_\epsilon = \Phi + \epsilon\eta$. Differentiating (30) at $\epsilon = 0$ and integrating by parts yields

$$0 = \frac{\mathrm{d}}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} \mathcal{J}[\Phi_\epsilon] = \int_\Omega \left(\frac{1}{4\pi G}\, \nabla\Phi\cdot\nabla\eta - \sigma\,\eta \right) \mathrm{d}^3x = \int_\Omega \left(-\frac{1}{4\pi G}\, \Delta\Phi - \sigma \right) \eta\, \mathrm{d}^3x,$$

which is exactly the weak formulation of (31). □

Remark 8.16 (Minimal closed-layer input for Poisson closure)**.** *Propositions 8.13–8.15 show that the Poisson closure for the phase potential is not an additional axiom once the defect-density interface is fixed. Beyond Axioms 3.1–3.2 and Hilbert folding (Axiom 3.4), the only additional closed-layer inputs are: (i) an auditable defect charge density σ (or σ_n) at a chosen horizon and readout scale (Definition 8.7); (ii) a boundary/gauge convention (periodic/Dirichlet/decay), including neutrality when required for solvability (Remark 8.10); and (iii) a macroscopic coupling constant G fixed by calibration (Remark 8.17). Given these, the Dirichlet principle uniquely determines Φ (up to the standard Laplacian zero mode in the periodic convention), and the force template follows by $\mathbf{P}_\Phi = -\nabla\Phi$ (Section 7.5).*

Remark 8.17 (Dimensional analysis and calibration)**.** *In the GR matching dictionary of Section 7, $\Phi = -\log\mathcal{N}$ is dimensionless. Defining the Newtonian potential $\phi_N := -c^2\Phi$ gives ϕ_N the standard physical units. Then* (31) *becomes the classical Poisson equation $\Delta\phi_N = 4\pi G\rho$ upon setting $\rho := c^2\sigma$. Operationally, the coefficient G is fixed by calibration against a reference redshift profile (e.g. Earth-to-GPS) once σ is defined from auditable protocol budgets.*

Remark 8.18 (Weak-field limit from general covariance)**.** *In general relativity, Einstein's equation is $G_{\mu\nu} = (8\pi G/c^4)\,T_{\mu\nu}$. In the weak, static, slow-motion limit with $T_{00} \approx \rho c^2$ and negligible stresses, one may write $g_{00} = -(1 + 2\phi_N/c^2) + O(c^{-4})$ and $g_{0i} = 0$. Linearizing the field equations yields the Newtonian Poisson equation $\Delta\phi_N = 4\pi G\rho$ (see, e.g., [36–38, 42]). In a static slicing with vanishing shift, the lapse is $N = \sqrt{-g_{00}} = 1 + \phi_N/c^2 + O(c^{-4})$, hence*

$$\Phi = -\log N = -\frac{\phi_N}{c^2} + O(c^{-4}).$$

Therefore $-\Delta\Phi = 4\pi G\,\rho/c^2 + O(c^{-4}) = 4\pi G\,\sigma + O(c^{-4})$, matching (31) *at leading order.*

9 Reproducible numerics and scripts

This section provides optional reproducible scripts illustrating intermediate mechanisms discussed in the paper. No closed-layer theorem depends on computation; the scripts are included to make the constructions auditable and testable.

9.1 Reproducibility protocol

All experiments are implemented in Python 3.10+. The Poisson solver uses `numpy` if available (FFT mode); if not, it falls back to a pure-Python iterative periodic solver (slower, but dependency-free). All other scripts use only the Python standard library. Run the following from this paper directory:

- `python3scripts/exp_hilbert_locality.py`
- `python3scripts/exp_golden_discrepancy_bound.py`
- `python3scripts/exp_abel_pole_barrier.py`
- `python3scripts/exp_poisson_fft.py`
- `python3scripts/exp_wigner_smith_kappa.py`
- `python3scripts/exp_address_family_sensitivity.py`
- `python3scripts/exp_kappa_redshift_toy.py`
- `python3scripts/exp_1p1d_routing_worked_example.py`

The scripts optionally write small LaTeX row files into `sections/generated/`. If those files are present, the tables below will include them.

Determinism and randomness

All scripts are deterministic as written. The only use of pseudorandomness is in the "shuffled baseline" of `scripts/exp_address_family_sensitivity.py`, and the RNG seeds are fixed in code (2D: `seed = 123456 + order`; 3D: `seed = 654321 + order`). Thus, regenerating tables does not require passing external seeds.

Generated artifacts (what each script writes)

Each script writes into `sections/generated/` using fixed filenames; tables include them when present.

- **Hilbert locality (Experiment A).**
  `exp_hilbert_locality.py` writes `hilbert_locality_rows.tex`.
  Defaults: orders $n = 1, \ldots, 8$.
- **Address-family sensitivity (Experiments A', A", A"', A3D).**
  `exp_address_family_sensitivity.py` writes
  `address_kappa_proxy_rows.tex`,
  `address_neighbor_model_sensitivity_rows.tex`,
  `address_kappa_proxy_fit_rows.tex`,
  `address_kappa_proxy_3d_rows.tex`,
  and (where used) `address_neighbor_separation_rows.tex`.
  Defaults: 2D orders $n = 1, \ldots, 8$ with Manhattan and Chebyshev neighborhoods. 3D orders $n = 1, \ldots, 5$. Shuffled baseline seeds fixed as stated above.
- **Golden discrepancy (Experiment B).** `exp_golden_discrepancy_bound.py` writes `golden_discrepancy_rows.tex`. *Defaults:* $\alpha = 1/\varphi$, $x_0 = 0.1$, and N in {100, 300, 1000, 3000, 10000, 30000}.
- **Abel pole barrier (Experiment C).** `exp_abel_pole_barrier.py` writes `abel_barrier_rows.tex`. *Defaults:* $\gamma = 1$ and $T_{\max} = 5000$. β in {0.5, 0.6}, with r sampled on a fixed grid around the threshold.
- **Poisson FFT (Experiments D and D').** `exp_poisson_fft.py` writes `poisson_rows.tex` and `poisson_scaling_rows.tex`. *Defaults:* FFT mode uses $N = 64$ and $r_{\max} = 12$ (fallback: $N = 24$). The scaling table uses N in {32, 48, 64} (FFT) plus a modest Dirichlet-Jacobi comparison at $N = 32$, with window band starting at $r = 3$.
- **Wigner–Smith interface (Experiment E).**
  `exp_wigner_smith_kappa.py` writes `wigner_smith_rows.tex`.
  Defaults: toy Breit–Wigner with $E_0 = 1$ and $\gamma = 0.2$. Energy band $[0, 2]$ sampled at $n = 17$ points. $\tau_0 = 1$, smoothing window = 1 (off).
- **Scan-chain redshift toy (Experiment F).**
  `exp_kappa_redshift_toy.py` writes `kappa_redshift_toy_rows.tex`.
  Defaults: order $n = 8$ and horizon $t_{\max} = 2 \times 10^8$ ticks. Maps: Hilbert and Z-order.
- **1+1D worked example (Appendix G).** `exp_1p1d_routing_worked_example.py` writes `1p1d_error_rows.tex`, `1p1d_scaling_fit_rows.tex`, and `1p1d_curve_rows.tex`. *Defaults:* weak-field target with $M = 0.05$ on r in $[1, 8]$, and $\kappa_0 = 10^9$. Orders $n = 6, \ldots, 11$ for scaling (representative curve samples at order $n = 9$).

9.2 Experiment A: Hilbert address locality (2D)

We verify the one-step locality property (9) for the standard 2D discrete Hilbert map up to order $n = 8$.

order n	$\min_t \|H_n(t+1) - H_n(t)\|_1$	$\max_t \|H_n(t+1) - H_n(t)\|_1$
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	1
8	1	1

Table 2: Hilbert address one-step locality check for orders $n = 1, \ldots, 8$.

9.3 Experiment A′: address-family sensitivity on a scan chain (2D)

We quantify how the choice of address family changes scan-order separations of screen neighbors. For an address map A_n and a screen-neighbor edge $\{x, y\}$, define the separation $\Delta_{A_n}(x, y) := |A_n^{-1}(x) - A_n^{-1}(y)|$ (Definition 5.3). On a scan chain, Δ_{A_n} lower bounds the depth required to enact neighbor interactions (Proposition 5.4). We report statistics of the induced local proxy $\widetilde{\kappa}_{A_n}(x) = \max_{y \sim x} \Delta_{A_n}(x, y)$ across all sites in Σ_n for Hilbert vs. Morton/Z-order, together with a deterministic shuffled baseline. Unless otherwise stated, the neighborhood relation $y \sim x$ here refers to the 4-neighbor (Manhattan) screen adjacency.

9.4 Experiment A″: neighborhood-model robustness (fixed order)

To assess robustness of the proxy to adjacency conventions, we recompute the local proxy at a fixed order,

$$\widetilde{\kappa}_{A_n}(x) = \max_{y \sim x} \Delta_{A_n}(x, y),$$

using both the Manhattan 4-neighbor adjacency and the Chebyshev 8-neighbor adjacency (including diagonals).

9.5 Experiment A‴: finite-size trend fit for high quantiles

To summarize finite-size scaling, we fit $\log_2 p_{99}$ and $\log_2(\max)$ as affine functions of n over the computed range $n = 1, \ldots, 8$ (least squares), and report the fitted slopes and R^2.

9.6 Experiment A_{3D}: address sensitivity in three dimensions (scan-chain proxy)

As a minimal dimensional robustness check, we repeat the same scan-chain proxy construction on a 3D screen lattice $\Sigma_n = \{0, \ldots, 2^n - 1\}^3$ with 6-neighbor (Manhattan) adjacency, comparing Morton/Z-order against a shuffled baseline. We report the local proxy $\widetilde{\kappa}_{A_n}(x) = \max_{y \sim x} |A_n^{-1}(x) - A_n^{-1}(y)|$ statistics across all sites.

9.7 Experiment B: golden-branch star discrepancy and an explicit bound

We compute $D_N^*(P_N)$ for the golden scan and compare it against the explicit bound (7).

order n	map	mean	p_{50}	p_{90}	p_{99}	max
1	Hilbert	2.00	1	3	3	3
1	Z-order	2.00	2	2	2	2
1	Shuffled	2.50	2	3	3	3
2	Hilbert	5.25	3	11	13	13
2	Z-order	4.25	3	6	6	6
2	Shuffled	9.56	11	12	15	15
3	Hilbert	12.94	7	43	53	53
3	Z-order	10.31	6	22	22	22
3	Shuffled	34.33	33	51	62	62
4	Hilbert	30.25	11	75	211	213
4	Z-order	24.08	11	86	86	86
4	Shuffled	147.07	146	217	243	243
5	Hilbert	66.92	13	179	819	853
5	Z-order	53.64	22	171	342	342
5	Shuffled	565.18	565	822	962	1010
6	Hilbert	142.42	19	301	2869	3413
6	Z-order	115.10	22	342	1366	1366
6	Shuffled	2319.87	2309	3403	3856	4054
7	Hilbert	295.62	19	341	5201	13653
7	Z-order	240.49	22	342	5462	5462
7	Shuffled	9228.63	9138	13536	15521	16341
8	Hilbert	604.24	19	681	13133	54613
8	Z-order	493.86	22	683	10923	21846
8	Shuffled	37072.80	36807	54171	62123	65327

Table 3: Address-family sensitivity on the 2D screen: statistics of the local scan-chain overhead proxy $\widetilde{\kappa}_{A_n}$ for Hilbert, Morton/Z-order, and a shuffled baseline.

9.8 Experiment C: Abel pole barrier toy model

We illustrate Lemma 2.2 using a toy mode $u_t = \mathrm{e}^{(\beta-\frac{1}{2})t}\cos(\gamma t)$. When $\beta > \frac{1}{2}$, the Abel weight must satisfy $r < \mathrm{e}^{-(\beta-\frac{1}{2})}$ to suppress exponential gain; otherwise the partial sums exhibit threshold blow-up.

9.9 Experiment D: Poisson phase potential via FFT

We solve $-\Delta\Phi = 4\pi\rho$ on a periodic grid and check that a localized source produces an approximate $1/r$ potential profile before periodic images dominate.

9.10 Experiment D′: finite-size and source-width robustness of the $1/r$ window

To quantify how cleanly a periodic finite grid reproduces an intermediate $1/r$ regime, we measure the flatness of $r\langle\Phi\rangle$ over a fixed radius band and report the relative RMS deviation as a simple error proxy, for multiple grid sizes and for a point source vs. a small cube-smeared source. For comparison, we also include a Dirichlet (zero boundary) iterative solve at a modest size.

map	$p_{99}^{(4)}$	$p_{99}^{(8)}$	ratio	$\max^{(4)}$	$\max^{(8)}$	ratio
Hilbert	13133	13136	1.000	54613	54613	1.000
Z-order	10923	10929	1.001	21846	32769	1.500
Shuffled	62123	62955	1.013	65327	65444	1.002

Table 4: Sensitivity of the overhead proxy to the neighborhood model at a fixed resolution (script default: order $n = 8$). Here (4) denotes Manhattan 4-neighbors and (8) denotes Chebyshev 8-neighbors.

map	slope for $\log_2 p_{99}$	R^2	slope for $\log_2(\max)$	R^2
Hilbert	1.751	0.987	2.015	1.000
Z-order	1.856	0.997	1.939	0.999
Shuffled	2.027	1.000	2.041	1.000

Table 5: Finite-size trend fit for high quantiles of the Manhattan proxy $\widetilde{\kappa}_{A_n}$.

9.11 Experiment E: Wigner–Smith time delay and $\kappa_{\mathrm{WS}}(E)$

To connect computational lapse to measurable delays, we include a minimal implementation of the Wigner–Smith time-delay matrix

$$Q(E) = -\mathrm{i}S(E)^\dagger \frac{\mathrm{d}S}{\mathrm{d}E},$$

the total delay $\tau_{\mathrm{WS}}(E) = \mathrm{Tr}Q(E)$, and the dimensionless overhead proxy $\kappa_{\mathrm{WS}}(E) = \tau_{\mathrm{WS}}(E)/\tau_0$ for a chosen reference tick duration τ_0. The script `scripts/exp_wigner_smith_kappa.py` provides an interface and includes a 1-channel Breit–Wigner toy model; users may replace it with a physical model or data for the scattering matrix $S(E)$.

9.12 Experiment F: a scan-chain redshift toy from a computed $\kappa(x)$ landscape

We demonstrate the redshift ratio (16) on a fully discrete scan-chain toy model. Fix a finite-resolution address map A_n (Hilbert or Morton/Z-order) and place screen sites on a scan chain by $\pi_n(x) = A_n^{-1}(x)$. Define a site-local overhead proxy by

$$\kappa(x) := \widetilde{\kappa}_{A_n}(x) = \max_{y \sim x} \left| A_n^{-1}(x) - A_n^{-1}(y) \right|,$$

as in Definition 5.3. Interpreting one local "clock cycle" at x as requiring $\kappa(x)$ global ticks, the lapse dictionary predicts

$$\frac{\mathrm{d}\tau_1}{\mathrm{d}\tau_2} = \frac{\kappa(x_2)}{\kappa(x_1)}.$$

The script below selects representative low/median/high overhead sites and verifies the ratio by counting completed cycles over a long horizon.

10 Conclusion

We presented a constructive spacetime framework in the HPA–Ω scan–readout paradigm. The key move is to treat finite-resolution readout as primitive: operational time and probability are induced by scan ticks and POVM instruments, while nonclassicality arises from a Weyl pair in the scan algebra.

order n	map	mean	p_{50}	p_{90}	p_{99}	max
1	Z-order	4.00	4	4	4	4
1	Shuffled	5.00	5	7	7	7
2	Z-order	18.88	14	28	28	28
2	Shuffled	37.38	37	51	59	59
3	Z-order	92.83	55	220	220	220
3	Shuffled	309.24	305	439	489	503
4	Z-order	426.14	220	1756	1756	1756
4	Shuffled	2487.73	2473	3499	3933	4069
5	Z-order	1852.72	220	7022	14044	14044
5	Shuffled	20068.47	19897	28014	31372	32656

Table 6: 3D scan-chain proxy statistics for Morton/Z-order and a shuffled baseline (orders $n = 1, \ldots, 5$).

N	$D_N^*(P_N)$	bound $2(2 + \log_\varphi N)/N$	ratio
100	0.0207764	0.231399	0.090
300	0.00575141	0.092353	0.062
1000	0.00134748	0.0327098	0.041
3000	0.000692067	0.0124253	0.056
10000	0.000244267	0.00422798	0.058
30000	8.00778e-05	0.00156153	0.051

Table 7: Exact star discrepancy for the golden-branch Kronecker sequence and the explicit logarithmic bound.

To obtain higher-dimensional locality from a one-dimensional scan, we introduced the Hilbert Folding Axiom, making space an explicit finite-resolution address graph rather than a background continuum. This exposes routing/compilation overhead as a physical resource and yields the computational lapse field $\mathcal{N} = \kappa_0/\kappa$ as an operational redshift factor. A minimal Poisson closure for a phase potential provides a falsifiable Newtonian-limit template with an approximate $1/r$ profile.

On the arithmetic side, we isolated a single bridge assumption (HTF) that embeds zeta-zero modes into an Abel-regularized scan trace. Under HTF and unit-disk holomorphy mandated by bounded scan readout, we proved a conditional rigidity theorem: off-critical zeros would force interior poles, contradicting holomorphy, hence $\mathrm{Re}(\rho) = \frac{1}{2}$ for all nontrivial zeros. This conditional rigidity statement is recorded as an arithmetic appendix (Appendix D).

Reproducible scripts are provided to audit the discrete Hilbert addressing property, discrepancy certificates, Abel threshold behavior, and Poisson potential numerics.

Limitations and open directions. The gravity dictionary developed here is operational and calibrated, but it is presently matched to the weak-field/static template: a full micro-to-macro derivation of the Einstein dynamics (including a dynamical closure for shift/drift and spatial-metric evolution) is not established in this paper. On the arithmetic side, the rigidity consequence is conditional: the remaining mathematical task is to construct a nontrivial HTF bridge compatible with the Abel-first/finite-part convention and bounded scan readout, beyond the explicit function-field model recorded in Appendix E. On the experimental side, the Wigner–Smith interface provides an actionable delay-based proxy $\kappa_{\mathrm{WS}}(E)$, but mapping it to a

β	r	$\lvert S(r;T_{\max})\rvert$
0.500	0.8000	7.321034e-01
0.500	0.8800	6.369817e-01
0.500	0.9000	6.134388e-01
0.500	0.9050	6.075852e-01
0.500	0.9200	5.901151e-01
0.500	0.9500	5.556554e-01
0.500	0.9800	5.219656e-01
0.600	0.8000	6.320969e-01
0.600	0.8800	5.302520e-01
0.600	0.9000	5.058303e-01
0.600	0.9050	1.910336e+00
0.600	0.9200	1.193817e+36
0.600	0.9500	5.656057e+105
0.600	0.9800	1.823025e+173

Table 8: Toy Abel-threshold behavior at fixed $T_{\max}$.

r	$\langle\Phi\rangle$	$r\langle\Phi\rangle$
1	+0.704947	+0.704947
2	+0.335675	+0.671351
3	+0.203919	+0.611756
4	+0.131873	+0.527493
5	+0.082613	+0.413063
6	+0.051132	+0.306791
7	+0.031001	+0.217009
8	+0.016342	+0.130732
9	+0.004706	+0.042356
10	-0.003518	-0.035177

Table 9: Representative radial averages from the FFT Poisson solver (periodic box).

spatial overhead field $\kappa(x)$ requires a concrete identification of scattering channels with localized protocol tasks at a stated resolution (Appendix F).

N	boundary/solver	source	window	count	mean($r\langle\Phi\rangle$)	rel. RMS
20	periodic-Jacobi	cube-0	3–5	3	+4.3128e-01	2.228e-01
20	periodic-Jacobi	cube-1	3–5	3	+4.3283e-01	2.194e-01
24	periodic-Jacobi	cube-0	3–6	4	+4.6478e-01	2.481e-01
24	periodic-Jacobi	cube-1	3–6	4	+4.6574e-01	2.457e-01
20	Dirichlet-Jacobi	cube-0	3–5	3	+6.0764e-01	1.104e-01
20	Dirichlet-Jacobi	cube-1	3–5	3	+6.0714e-01	1.094e-01

Table 10: Finite-size and source-width robustness proxy for the $1/r$ window.

E	$\tau_{\mathrm{WS}}(E)$	$\kappa_{\mathrm{WS}}(E)$
0.125	0.263063	0.263063
0.25	0.358906	0.358906
0.375	0.518752	0.518752
0.5	0.815577	0.815577
0.625	1.46489	1.46489
0.75	3.31311	3.31311
0.875	9.52232	9.52232
1	14.3369	14.3369
1.125	9.52232	9.52232
1.25	3.31311	3.31311
1.375	1.46489	1.46489
1.5	0.815577	0.815577
1.625	0.518752	0.518752
1.75	0.358906	0.358906
1.875	0.263063	0.263063

Table 11: Representative values from the Wigner–Smith toy model, providing an auditable interface for $\kappa_{\mathrm{WS}}(E)$.

order n	map	x_1	$\kappa(x_1)$	x_2	$\kappa(x_2)$	predicted	measured
8	Hilbert	(0,255)	1	(2,7)	19	19.000000	19.000001
8	Hilbert	(0,255)	1	(127,0)	54613	54613.000000	54614.964500
8	Hilbert	(2,7)	19	(127,0)	54613	2874.368421	2874.471600
8	Z-order	(0,0)	2	(0,3)	22	11.000000	11.000000
8	Z-order	(0,0)	2	(0,127)	21846	10923.000000	10924.186148
8	Z-order	(0,3)	22	(0,127)	21846	993.000000	993.107822

Table 12: Scan-chain toy redshift verification from a computed overhead proxy $\kappa(x)$ (Hilbert vs Morton/Z-order).

A Discrete Hilbert addressing and one-step locality

The Hilbert curve is a classical space-filling construction mapping an interval continuously onto a square (and, more generally, onto higher-dimensional cubes) [31, 32]. For the purposes of this paper, we use only its *finite-resolution* discrete approximation: a bijection between indices $\{0, \dots, 2^{dn} - 1\}$ and lattice sites $\{0, \dots, 2^n - 1\}^d$ that visits each lattice site exactly once.

One-step locality. Standard Hilbert orders are constructed so that consecutive indices correspond to lattice neighbors (in ℓ^1), i.e. (9). This property is crucial for interpreting the address map as a locality-preserving scan schedule.

Hölder scaling (continuum limit). The continuous Hilbert curve is Hölder continuous with exponent $1/d$ [32], which quantitatively expresses locality preservation under coarse graining. In the present paper, this scaling is invoked only as a standard mathematical property supporting the finite-resolution interpretation; all operational claims remain at finite n.

Implementation (2D). For $d = 2$, a compact bitwise algorithm maps a Hilbert index to (x, y) coordinates. We include a pure-Python reference implementation in `scripts/exp_hilbert_locality.py`, which also brute-force verifies that $\|H_n(t+1) - H_n(t)\|_1 = 1$ for all t and for orders up to $n = 8$. The script emits a small LaTeX row file for Table 2.

B Notes on star discrepancy, Denjoy–Koksma, and the golden-branch bound

We recall the one-dimensional star discrepancy

$$D_N^*(P_N) = \sup_{u \in [0,1]} \left| \frac{1}{N} \#\{t < N : x_t < u\} - u \right|$$

for $P_N = \{x_0, \dots, x_{N-1}\} \subset [0, 1)$, and the Koksma inequality (6) controlling sampling error for bounded-variation observables [26, 27].

Rotation sequences and bounded-type slopes. For an irrational rotation $x_t = x_0 + t\alpha \pmod 1$, discrepancy behavior is governed by the continued fraction of α. When the partial quotients are uniformly bounded (bounded type), Denjoy–Koksma estimates at convergent times yield logarithmic mismatch stability; see [26, 46, 47].

A continued-fraction certificate. A standard bound for Kronecker sequences controls star discrepancy in terms of the continued-fraction data of α: if p_m/q_m is a convergent and $q_m \le N < q_{m+1}$, then one has

$$D_N^*(P_N) \;\le\; \frac{1 + \sum_{k=1}^{m} a_k}{N},$$

where a_k are the partial quotients of α; see, e.g., [26, Ch. 2] or [27, Ch. 1]. For bounded type, $\sum_{k=1}^{m} a_k = O(m) = O(\log N)$, giving the familiar $O((\log N)/N)$ rate.

Golden branch. For the golden slope $\alpha = \varphi^{-1} = [0; 1, 1, 1, \dots]$, all partial quotients satisfy $a_k = 1$ and the convergent denominators are Fibonacci numbers, hence q_m grows exponentially like φ^m. Specializing the continued-fraction certificate above therefore yields an explicit logarithmic bound of order $(\log N)/N$; (7) is a convenient uniform certificate. The script `scripts/exp_golden_discrepancy_bound.py` computes $D_N^*(P_N)$ exactly from its definition and compares it against (7) for representative N.

Higher-dimensional discrepancy and the inverse problem. The discussion in the main text emphasizes one-dimensional deterministic certificates as a minimal auditable interface. In dimension $d \geq 2$, the theory of star discrepancy becomes substantially more delicate, with sharp bounds depending on d, the point-set class, and the norm used to measure irregularities; see, e.g., [48, 49]. One structural fact relevant to any "deterministic error certificate" narrative is the *inverse discrepancy* problem: how large N must be (as a function of d and ε) to guarantee $D_N^* \leq \varepsilon$. In particular, the inverse of the star discrepancy can scale at least linearly with the dimension in general settings [50]. Recent work also relates discrepancy growth to additive energy and other arithmetic structure of sequences, providing additional mechanisms by which protocols can exhibit large irregularities [51].

A sensitivity channel for coarse-grained field estimation. The closed-layer role of discrepancy is to provide deterministic, non-asymptotic error certificates for finite-horizon readout. For one-dimensional scan samples, Koksma-type inequalities bound the error of estimating a bounded-variation observable by $\mathrm{Var}(f)\, D_N^*(P_N)$ (cf. (6)). When a protocol source density σ is built from occupation bias and then coarse-grained by a kernel $w^{(\varepsilon)}$ (Example 8.8), the same discrepancy controls the deviation of the coarse-grained empirical density from its uniform reference at fixed horizon and resolution. Propagation to the phase potential and lapse is then controlled by the Poisson stability estimate (Proposition 5.7), which bounds the induced perturbation of $\nabla\Phi$ in terms of the H^{-1} norm of the source perturbation. This provides a concrete sensitivity pipeline from "off-golden" scan irregularities to coarse-grained geometric prediction errors.

C Abel finite parts and unit-disk analyticity (notes)

Abel regularization replaces divergent infinite-horizon expressions by holomorphic generating functions on the unit disk. Given a bounded sequence $(a_t)_{t\geq 0}$, define

$$\mathcal{A}_a(r) := \sum_{t\geq 0} a_t r^t, \qquad |r| < 1.$$

This is holomorphic for $|r| < 1$ and is computable at every $r \in (0, 1)$. The only universal singular behavior allowed by absolute convergence is at the boundary $r \uparrow 1$.

Finite-part template. If $\mathcal{A}_a(r)$ admits an expansion

$$\mathcal{A}_a(r) = \frac{c_{-1}}{1-r} + c_0 + c_1(1-r) + \cdots \qquad (r \uparrow 1),$$

then the Abel finite part is defined by $\mathrm{FP}_{r\uparrow 1}\mathcal{A}_a(r) := c_0$. This is the canonical constant-term extraction used in Convention 3.3; see [52,53] for classical treatments of Abelian summation and finite-part asymptotics.

Rotation resolvent criterion. For rotation orbits with a sufficiently regular kernel f (absolutely summable Fourier coefficients), the resolvent identity (3) shows that $\mathcal{A}(r)$ is holomorphic on $|r| < 1$ and that all possible singularities occur only on the boundary $|r| = 1$. This is the analytic backbone of the interior pole-barrier argument in Lemma 2.2.

Proposition C.1 (Canonical pole subtraction and scheme stability for rotation kernels)**.** *Let $\alpha \in (0, 1) \setminus \mathbb{Q}$ and let $x_t = x_0 + t\alpha \pmod 1$. Let $f : \mathbb{T} \to \mathbb{C}$ have an absolutely summable Fourier series $\sum_{m\in\mathbb{Z}} \left|\widehat{f}(m)\right| < \infty$ and define the Abel orbit sum $S_f(r) = \sum_{t\geq 0} r^t f(x_t)$ for $|r| < 1$. Then:*

1. ***Unique universal pole.*** *One has the decomposition*

$$S_f(r) = \frac{\widehat{f}(0)}{1-r} \; + \; H_f(r),$$

where H_f is holomorphic on $|r| < 1$ and extends continuously to $r \uparrow 1$. In particular, no subleading divergences of the form $\log(1-r)$ occur in this kernel class.

2. ***Finite-part existence and uniqueness.*** *The Abel finite part in* (4) *exists and equals $H_f(1)$.*

3. ***Scheme dependence is reduced to a fixed constant.*** *If a modified subtraction scheme uses*

$$\lim_{r\uparrow 1}\left(S_f(r) - \frac{\widehat{f}(0)}{1-r} - g(r)\right)$$

for some function g holomorphic in a neighborhood of $r = 1$, then the resulting value equals $H_f(1) - g(1)$. Thus, within the admissible kernel class and with the Abel-first canonical path fixed, the only ambiguity is an additive constant corresponding to an explicitly chosen counterterm $g(1)$; Convention 3.3 fixes this ambiguity by taking $g \equiv 0$.

Proof. By (3),

$$S_f(r) = \sum_{m\in\mathbb{Z}} \widehat{f}(m)\, \mathrm{e}^{2\pi \mathrm{i} m x_0}\, \frac{1}{1 - r\,\mathrm{e}^{2\pi \mathrm{i} m\alpha}}.$$

The $m = 0$ term equals $\widehat{f}(0)/(1-r)$. For $m \neq 0$, irrationality of α implies $\mathrm{e}^{2\pi \mathrm{i} m\alpha} \neq 1$, so the denominators do not vanish at $r = 1$; absolute summability of $\widehat{f}(m)$ then gives uniform convergence and holomorphy of the remaining sum on $|r| < 1$ and continuity at $r \uparrow 1$. This yields the claimed decomposition with $H_f(1)$ finite and excludes subleading divergences in this class. The scheme-stability statement is immediate from holomorphy of g at $r = 1$. □

Remark C.2 (Inadmissibility as a protection against subleading divergences)**.** *If one enlarges the kernel class or the protocol object so that $S_f(r)$ develops subleading divergences at $r \uparrow 1$ (for example, logarithmic terms), then the canonical finite-part limit in* (4) *need not exist. In the Ω layer discipline, such quantities are rejected as inadmissible: the closed layer only assigns protocol values when the Abel-first/finite-part convention yields a well-defined finite part along the canonical path.*

C.1 Relation to zeta/theta regularization in QFT and Casimir-type calculations

Abel-first finite parts are closely related to standard analytic regularization methods in QFT, including zeta-function regularization and heat-kernel/theta regularization. We record the correspondence and the (limited) scheme dependence.

Discrete Abel weight vs. Laplace/heat-kernel weights. Writing $r = \mathrm{e}^{-s}$ with $\mathrm{Re}(s) > 0$ turns an Abel sum $\sum_{t\geq 0} a_t r^t$ into a discrete Laplace transform $\sum_{t\geq 0} a_t \mathrm{e}^{-st}$. This is the discrete-time analogue of the heat-kernel regularization $\mathrm{Tr}(\mathrm{e}^{-tA})$ and the associated Mellin transform/zeta regularization $\zeta_A(s) = \mathrm{Tr}(A^{-s})$ in spectral problems [54–56]. In both settings, the renormalized value is obtained by extracting a finite part (constant term) after subtracting the universal divergent piece.

Finite-part extraction and renormalization conditions. In the present framework, the canonical prescription is the Abel-first path $r \uparrow 1$ and the constant-term extraction $\mathrm{FP}_{r\uparrow 1}$ (Convention 3.3). In QFT/Casimir calculations, different zeta/heat-kernel subtraction schemes correspond to different choices of local counterterms (renormalization conditions). Proposition C.1 makes the analogous statement in our admissible kernel class: modifying the subtraction by a holomorphic counterterm $g(r)$ shifts the finite part by an explicit additive constant $g(1)$. Thus, whenever the Abel finite part is well-defined in the sense of this appendix, the only remaining scheme dependence is a controlled finite ambiguity corresponding to a chosen (local) counterterm, exactly as in analytic renormalization.

When can finite parts differ? Finite parts can differ if one compares:

- different subtraction choices (different g), which shift the finite part by a constant; or
- different regularization families (e.g. Abel vs. a non-equivalent cutoff) outside the admissible class, where subleading divergences (e.g. logarithms) can appear and the Abel finite part may fail to exist (Remark C.2).

In the closed-layer Ω discipline, such differences are not treated as paradoxes: the protocol value is defined only after fixing the canonical path and the subtraction convention, and any additional finite renormalization is an explicit, auditable modeling choice (cf. HTF-Types/HTF-Renorm in Assumption D.6). For comparisons with physics literature using analytic renormalization, see, e.g., [57] for representative Casimir/QFT discussions.

D Arithmetic appendix: holographic trace formulas and a conditional Riemann critical-line rigidity theorem

D.1 Geometric-side holomorphy on the unit disk

In the closed layer, Abel-regularized traces are primary objects.

Lemma D.1 (Bounded scan readout implies unit-disk holomorphy)**.** *Let U be a unitary on $\mathcal{H}_{\mathrm{eff}}$, let ω_{eff} be an effective state, and let A be a bounded operator. Define the scan-readout sequence*

$$a_t := \omega_{\mathrm{eff}}\left(U^t A U^{-t}\right), \qquad t \in \mathbb{Z}_{\geq 0}.$$

Then (a_t) is bounded, and the Abel generating function

$$G(r) := \sum_{t\geq 0} a_t r^t$$

converges absolutely for $|r| < 1$ and is holomorphic on the unit disk.

Proof. Since U is unitary, $\|U^t A U^{-t}\| = \|A\|$. For any state ω_{eff} on a C^*-algebra one has $|\omega_{\mathrm{eff}}(B)| \leq \|B\|$, hence $|a_t| \leq \|A\|$ for all t. Absolute convergence for $|r| < 1$ and holomorphy of the resulting power series are immediate. □

Corollary D.2 (Geometric-side growth bounds on the unit disk)**.** *In the setting of Lemma D.1, for every $r \in \mathbb{C}$ with $|r| < 1$ one has the explicit bound*

$$|G(r)| \;\leq\; \frac{\|A\|}{1-|r|}. \tag{32}$$

More generally, for every integer $k \geq 0$ the k-th derivative exists and satisfies

$$\left|G^{(k)}(r)\right| \;\leq\; \frac{k!\,\|A\|}{(1-|r|)^{k+1}}.$$

Proof. By Lemma D.1, $|a_t| \le \|A\|$. Thus, for $|r| < 1$,

$$|G(r)| \le \sum_{t\ge 0} |a_t|\ |r|^t \le \|A\| \sum_{t\ge 0} |r|^t = \frac{\|A\|}{1-|r|}.$$

Termwise differentiation is justified by absolute convergence on compact subsets of $|r| < 1$, and the derivative bound follows from $\sum_{t\ge k} t(t-1)\cdots(t-k+1)\,|r|^{t-k} \le k!/(1-|r|)^{k+1}$. □

Lemma D.1 provides the geometric-side analyticity constraint mandated by bounded scan readout (Axioms 3.1–3.2). This unit-disk holomorphy will be paired with the spectral-side mode structure under HTF.

Remark D.3 (Concrete scan models realizing the holomorphy condition)**.** *Lemma D.1 is intentionally minimal: it isolates the unit-disk analyticity that follows from bounded readout, independent of any arithmetic content. A concrete realization consistent with the Weyl-pair setting is obtained by taking $\mathcal{H}_{\text{eff}} = L^2(\mathbb{T})$, letting U be the irrational rotation shift $(U\psi)(x) = \psi(x+\alpha)$, choosing A as a bounded multiplication operator $(A\psi)(x) = f(x)\psi(x)$ with $f \in L^\infty(\mathbb{T})$, and letting ω_{eff} be any normal state (e.g. a vector state). Then $a_t = \omega_{\text{eff}}(U^t A U^{-t})$ is a bounded scan-readout sequence, and the Abel generating function $G(r)$ is holomorphic on $|r| < 1$. In any HTF candidate, this holomorphy verification is therefore straightforward once $(U, \omega_{\text{eff}}, A)$ are fixed; the nontrivial content is the existence of a trace identity whose spectral mode decomposition contains the zeta data.*

Definition D.4 (Abel-regularized scan trace)**.** *In the setting of Lemma D.1, we call*

$$G(r) = \sum_{t\ge 0} \omega_{\text{eff}}\left(U^t A U^{-t}\right) r^t$$

the Abel-regularized scan trace *associated to* $(U, \omega_{\text{eff}}, A)$*. It is holomorphic on* $|r| < 1$*.*

D.2 HTF as a bridge package (structured assumptions)

The Riemann Hypothesis in this framework is treated as a *protocol consequence* of a trace identity that embeds arithmetic spectral data into scan traces. We isolate the needed input as a single assumption.

Remark D.5 (A fully explicit model exists in the function-field setting)**.** *Before stating HTF, we emphasize that the logical structure required here is not a purely formal slogan. Appendix E records an unconditional HTF-style trace identity for Weil zeta functions of curves over finite fields, where the analogue of RH is known. That example shows concretely how a resolvent-mode decomposition produces interior poles when a spectral radius exceeds* 1*, and how unit-disk holomorphy forces a "critical-line" modulus constraint.*

Assumption D.6 (HTF: holographic trace formula bridge (Id, Types, Modes, Renorm, Non-Cancel))**.** *There exists a choice of $(U, \omega_{\text{eff}}, A)$ and an Abel-regularized scan trace G in the sense of Definition D.4, together with a spectral term $\mathcal{S}$ and a regular term $\mathcal{A}_\infty$ on the unit disk, such that:*

morphic identity). *The following identity holds on $|r| < 1$ as an equality of meromorphic functions:*

$$G(r) = \mathcal{S}(r) + \mathcal{A}_\infty(r), \qquad |r| < 1, \tag{33}$$

(analytic types). *$G(r)$ is holomorphic on $|r| < 1$ by Lemma D.1. The regular term $\mathcal{A}_\infty(r)$ is holomorphic on $|r| < 1$ and collects archimedean/renormalization contributions compatible with the Abel-first/finite-part convention.*

resolvent modes). *The spectral term $\mathcal{S}(r)$ is meromorphic on $|r| < 1$ and admits a decomposition*

$$\mathcal{S}(r) = \sum_{\rho} c_\rho \, \frac{1}{1 - r\,\mathrm{e}^{(\rho - \frac{1}{2})}} \; + \; \mathcal{H}(r), \tag{34}$$

where the sum ranges over nontrivial zeros ρ of ζ, the coefficients c_ρ are complex numbers, and $\mathcal{H}(r)$ is holomorphic on $|r| < 1$. The sum is understood with a fixed summability/renormalization prescription (for example, symmetric pairing of zeros and an Abel-first limiting procedure) chosen so that the right-hand side defines a meromorphic function on $|r| < 1$.

renormalization). *The chosen summability/renormalization prescription is fixed once and for all (as part of the HTF package) and defines $\mathcal{S}(r)$ as a meromorphic function on $|r| < 1$ whose principal parts at its poles agree with the principal parts of the corresponding resolvent-mode factors in* (34)*. In particular, the prescription does not remove interior poles by subtracting their principal parts.*

ole cancellation). *Whenever $\mathrm{Re}(\rho) > \frac{1}{2}$ (equivalently $|r_\rho| < 1$), the point*

$$r_\rho = \mathrm{e}^{-(\rho - \frac{1}{2})}$$

is an actual pole of $\mathcal{S}(r)$, i.e. the principal part of $\mathcal{S}$ at $r = r_\rho$ is nonzero. Equivalently, the total residue at $r = r_\rho$ contributed by all zero modes mapping to r_ρ is nonzero.

Remark D.7 (On "cancellation hiding poles")**.** *In a meromorphic identity on $|r| < 1$ of the form $G = \mathcal{S} + \mathcal{A}_\infty$, the regular term $\mathcal{A}_\infty$ is holomorphic by HTF-Types and therefore cannot cancel any interior pole. Hence any interior pole of the right-hand side must be canceled* within *$\mathcal{S}$ itself. But cancellation between different pole locations is impossible: principal parts at distinct points are independent. The only way for a pole to be "hidden" is therefore a* same-point *cancellation (the total principal part at that pole vanishes). HTF-NonCancel excludes exactly this degenerate possibility, and it is automatically satisfied in explicit resolvent-trace models in which residues are fixed nonzero spectral multiplicities (Appendix E gives a fully explicit instance).*

Assumption D.6 is a protocol-level abstraction of the classical "explicit formula as trace" paradigm [23, 58, 59], rewritten in Abel r-coordinates with the Ω canonical path; see also the companion protocol trace manuscript [60]. The present paper does not attempt to construct such an identity; rather, it isolates the analytic hypotheses and proves the rigidity consequence *if* HTF holds.

Remark D.8 (Logical separation and why the argument is not circular)**.** *The rigidity step proved below is purely complex-analytic: unit-disk holomorphy of the geometric-side Abel trace G (a boundedness consequence) is incompatible with interior poles produced by exponential-growth modes. HTF does* not *assume RH; it assumes the existence of a trace identity whose spectral side contains zeta-zero resolvent factors in the concrete Abel coordinate r. The conditional theorem should therefore be read as a* rigidity constraint on admissible trace bridges*: any HTF candidate compatible with bounded scan readout must be consistent with the critical-line location of all zero modes.*

Remark D.9 (HTF as a concrete verification checklist)**.** *Assumption D.6 is formulated so that its components can be checked modularly in any proposed operator/dynamical model: (i) the geometric-side boundedness and unit-disk holomorphy of G are ensured by the scan-readout interface (Lemma D.1); (ii) the regular term $\mathcal{A}_\infty$ must be holomorphic and compatible with the Abel-first/finite-part discipline (Convention 3.3 and Appendix C); (iii) the spectral term must admit a meromorphic mode decomposition with resolvent factors that produce interior poles for off-critical growth, and those poles must not be canceled. Appendix E provides a fully explicit model where this checklist is satisfied unconditionally (with the Frobenius spectrum in place of zeta zeros).*

Remark D.10 (A fully explicit HTF instance in the function-field setting)**.** *In the setting of zeta functions of curves over finite fields, trace identities of the form* (33) *are explicit and unconditional, with spectral modes given by Frobenius eigenvalues. Appendix E records a concrete model showing how the pole-barrier mechanism becomes a transparent spectral-radius statement.*

D.3 Relation to explicit formulas and operator-trace frameworks

Abel coordinate r and Laplace coordinate s. Writing $r = \mathrm{e}^{-s}$ with $\mathrm{Re}(s) > 0$ converts Abel weights into a Laplace-like coordinate. The zero-mode factor in (34) becomes

$$\frac{1}{1 - r\,\mathrm{e}^{(\rho-\frac{1}{2})}} \;=\; \frac{1}{1 - \mathrm{e}^{-(s-(\rho-\frac{1}{2}))}}.$$

Thus, an off-critical zero with $\mathrm{Re}(\rho) > \frac{1}{2}$ produces a pole at $r_\rho = \mathrm{e}^{-(\rho-\frac{1}{2})}$ strictly inside the unit disk (Lemma 2.2). This is the analytic content of the "pole barrier" mechanism: in Abel/Laplace coordinates, exponential-growth modes correspond to interior resolvent singularities.

Resolvent and determinant viewpoint (operator-theoretic intuition). Factors of the form $(1 - r\lambda)^{-1}$ are resolvent kernels. If an operator F has eigenvalues $\{\lambda_j\}$ and the trace $\mathrm{Tr}((I - rF)^{-1})$ is meaningful (possibly after regularization), then formally

$$\mathrm{Tr}\big((I - rF)^{-1}\big) = \sum_j \frac{1}{1 - r\lambda_j}.$$

The function-field model in Appendix E is exactly of this form with F the normalized Frobenius action. In an operator-trace instantiation of HTF, (34) should be interpreted as asserting that a suitably regularized resolvent trace contains zeta-zero modes as resolvent singularities in the Abel coordinate r (equivalently, in the Laplace coordinate s). Related determinant regularizations can be viewed through $\log\det(I - rF) = -\sum_{n\geq 1} \frac{r^n}{n}\mathrm{Tr}(F^n)$ when such expansions are justified.

Test-function viewpoint (explicit-formula dictionary). The Abel trace is the generating function

$$G(r) = \sum_{t\geq 0} a_t r^t,$$

and under $r = \mathrm{e}^{-s}$ it becomes the discrete Laplace transform

$$G(\mathrm{e}^{-s}) = \sum_{t\geq 0} a_t\,\mathrm{e}^{-st}.$$

In this language, the factor $\big(1 - \mathrm{e}^{-(s-(\rho-\frac{1}{2}))}\big)^{-1}$ is exactly the Laplace image of the geometric mode $t \mapsto \mathrm{e}^{(\rho-\frac{1}{2})t}$. Classical explicit formulas can be viewed as trace identities evaluated against a family of test functions; HTF specializes to a concrete "geometric-series" test family adapted to the Abel-first discipline.

Connection to classical explicit formulas and Connes' trace framework. In analytic number theory, the Weil explicit formula relates test-function transforms of primes to test-function transforms of the nontrivial zeros of ζ; see, e.g., [58, 59]. Connes' semilocal trace formula provides an operator-theoretic realization of this paradigm in noncommutative geometry, in which the zero spectrum appears through trace identities [23]. HTF should be understood as the assertion that a specific Abel-regularized scan trace $G(r)$ admits such a trace realization in a form compatible with the Ω Abel-first discipline.

Relation to prolate-wave operator constructions. Recent work constructs explicit operators whose spectral data encode zeta zeros in novel ways, including prolate wave operators [25]. These constructions exemplify the operator-theoretic Hilbert–Pólya route that HTF presupposes: one seeks a concrete spectral object and a trace identity whose mode decomposition contains the zeros [24]. From the present closed-layer perspective, HTF packages the existence of such a trace bridge into a single auditable assumption, after which the rigidity consequence follows from unit-disk holomorphy.

D.4 Interior poles from off-critical zeros

Theorem D.11 (Riemann critical-line rigidity (conditional))**.** *Assume Axioms 3.1–3.2, Convention 3.3, and the HTF bridge Assumption D.6 (HTF-Id/Types/Modes/Renorm/NonCancel). If the geometric-side trace $G(r)$ is holomorphic on the unit disk $|r| < 1$, then every nontrivial zero ρ of ζ satisfies*

$$\mathrm{Re}(\rho) = \frac{1}{2}.$$

Proof. Suppose, for contradiction, that there exists a nontrivial zero ρ with $\mathrm{Re}(\rho) > \frac{1}{2}$. By Lemma 2.2, the corresponding mode factor $\left(1 - r\,\mathrm{e}^{(\rho-\frac{1}{2})}\right)^{-1}$ has a pole at $r_\rho = \mathrm{e}^{-(\rho-\frac{1}{2})}$ with $|r_\rho| < 1$. Under Assumption D.6, the point $r = r_\rho$ is an actual pole of $\mathcal{S}(r)$. Since $\mathcal{A}_\infty(r)$ is holomorphic on $|r| < 1$, the right-hand side of (33) has an interior pole at $r = r_\rho$.

On the other hand, by the boundedness/holomorphy of Abel traces on $|r| < 1$, the left-hand side $G(r)$ is holomorphic on the entire unit disk and has no interior poles. This contradicts (33) as an equality of meromorphic functions on $|r| < 1$. Hence no zero can satisfy $\mathrm{Re}(\rho) > \frac{1}{2}$.

Finally, the functional equation symmetry $\rho \mapsto 1 - \rho$ implies that if a zero with $\mathrm{Re}(\rho) < \frac{1}{2}$ existed, then $1 - \rho$ would be a zero with real part $> \frac{1}{2}$, which has been excluded. Therefore all nontrivial zeros lie on the critical line. □

Remark D.12 (Status of the result)**.** *Theorem D.11 is a closed-layer* protocol consequence *of HTF and unit-disk holomorphy. The mathematical difficulty is isolated into a single task: construct an HTF bridge compatible with the Abel-first/finite-part convention and with bounded scan readout.*

E An explicit HTF model: Frobenius trace and Weil zeta for curves over finite fields

This appendix records a fully explicit instance of an HTF-style trace bridge in a setting where the analogue of RH is known: zeta functions of smooth projective curves over finite fields. The purpose is not to re-prove the Weil conjectures, but to exhibit a concrete trace identity in which spectral modes appear through resolvent factors and the pole-barrier mechanism becomes an elementary spectral-radius statement.

E.1 Weil zeta and Frobenius eigenmodes

Let C be a smooth projective geometrically connected curve of genus g over $\mathbb{F}_q$. Define the point counts $N_n := \#C(\mathbb{F}_{q^n})$ and the Weil zeta function

$$Z_C(u) := \exp\left(\sum_{n\geq 1} \frac{N_n}{n}\, u^n\right).$$

It is a rational function of the form

$$Z_C(u) = \frac{P_C(u)}{(1-u)(1-qu)}, \qquad P_C(u) = \prod_{j=1}^{2g} (1 - \alpha_j u)\,, \tag{35}$$

where the α_j are algebraic numbers encoding the Frobenius spectrum on the first cohomology of C (see, e.g., [61]). The nontrivial zeros of Z_C occur at $u = \alpha_j^{-1}$. The Weil RH statement for curves is the modulus condition

$$|\alpha_j| = q^{1/2} \qquad (1 \le j \le 2g),$$

equivalently $\left|\alpha_j^{-1}\right| = q^{-1/2}$.

E.2 A resolvent trace generating function and interior poles

Define the normalized eigenvalues

$$\lambda_j := \frac{\alpha_j}{q^{1/2}}, \qquad 1 \le j \le 2g.$$

Consider the formal power series

$$G(r) := \sum_{n \ge 0} \left(\sum_{j=1}^{2g} \lambda_j^n \right) r^n, \qquad r \in \mathbb{C}. \tag{36}$$

Whenever the sequence $\sum_j \lambda_j^n$ is bounded, the series (36) is holomorphic on the unit disk $|r| < 1$ (cf. Lemma D.1). On the other hand, for $|r| < 1$ one has the resolvent identity

$$G(r) = \sum_{j=1}^{2g} \frac{1}{1 - r\lambda_j}, \tag{37}$$

which is meromorphic with poles at $r = \lambda_j^{-1}$. Thus:

- if $|\lambda_j| > 1$ for some j, then $r = \lambda_j^{-1}$ is a pole strictly inside $|r| < 1$;
- if $G(r)$ is holomorphic on $|r| < 1$, then necessarily $|\lambda_j| \le 1$ for all j;
- applying the same argument to the dual spectrum (or using the functional equation symmetry in (35)) forces $|\lambda_j| = 1$ for all j.

E.3 Relation to the pole-barrier mechanism

The identity (37) is a concrete trace bridge in the same structural form as Assumption D.6: the geometric-side holomorphy of an Abel-type trace on $|r| < 1$ is incompatible with interior poles created by exponential-growth spectral modes. In the function-field setting, the spectral data are explicit (Frobenius eigenvalues), and the "critical line" statement becomes the unit-modulus constraint $|\lambda_j| = 1$, i.e. $|\alpha_j| = q^{1/2}$. This provides a fully worked model demonstrating that an HTF package can be realized as a genuine trace identity rather than as a purely heuristic slogan: the resolvent identity gives the mode decomposition (HTF-Modes), the unit-disk holomorphy criterion is explicit, and interior poles are genuine unless the corresponding mode coefficient vanishes (HTF-NonCancel).

E.4 A concrete genus-one example (explicit poles and noncancellation)

We now instantiate the discussion on an explicit genus-one curve, where all objects can be written down concretely.

Elliptic curve over $\mathbb{F}_5$. Let $q = 5$ and consider the elliptic curve

$$E:\ y^2 = x^3 + x + 1 \qquad \text{over } \mathbb{F}_5.$$

Direct point counting gives $\#E(\mathbb{F}_5) = 9$, hence the Frobenius trace is

$$a := q + 1 - \#E(\mathbb{F}_5) = 6 - 9 = -3.$$

The Weil zeta numerator is therefore

$$P_E(u) = 1 - au + qu^2 = 1 + 3u + 5u^2.$$

The Frobenius eigenvalues $\alpha_\pm$ are the roots of $T^2 - aT + q = 0$, i.e.

$$\alpha_\pm = \frac{a \pm \sqrt{a^2 - 4q}}{2} = \frac{-3 \pm \mathrm{i}\sqrt{11}}{2}, \qquad |\alpha_\pm| = \sqrt{5}.$$

The normalized eigenvalues are

$$\lambda_\pm := \frac{\alpha_\pm}{\sqrt{q}} = \frac{-3 \pm \mathrm{i}\sqrt{11}}{2\sqrt{5}}, \qquad |\lambda_\pm| = 1, \qquad \lambda_- = \overline{\lambda_+} = \lambda_+^{-1}. \tag{38}$$

Geometric-side boundedness and unit-disk holomorphy. In genus $g = 1$ one has $2g = 2$ modes, and the coefficient sequence in (36) becomes

$$b_n := \sum_{j=1}^{2} \lambda_j^n = \lambda_+^n + \lambda_-^n = 2\,\mathrm{Re}(\lambda_+^n), \qquad n \geq 0.$$

Since $|\lambda_\pm| = 1$, the coefficients are bounded by $|b_n| \leq 2$ for all n. Therefore the Abel series $G(r) = \sum_{n\geq 0} b_n r^n$ is holomorphic on $|r| < 1$ (cf. Lemma D.1).

Spectral-side resolvent decomposition and explicit poles. The resolvent identity (37) is completely explicit here:

$$G(r) = \frac{1}{1 - r\lambda_+} + \frac{1}{1 - r\lambda_-}. \tag{39}$$

Hence G is meromorphic with simple poles at

$$r_\pm = \lambda_\pm^{-1} = \overline{\lambda_\pm}, \qquad |r_\pm| = 1.$$

In particular, there are *no* poles in the open unit disk, matching the geometric-side holomorphy.

Noncancellation is explicit. The principal part of $(1 - r\lambda_\pm)^{-1}$ at $r = r_\pm$ is

$$\frac{1}{1 - r\lambda_\pm} = -\frac{1}{\lambda_\pm} \cdot \frac{1}{r - r_\pm} + (\text{holomorphic}),$$

so the residue is $-1/\lambda_\pm \neq 0$. Because $r_+ \neq r_-$, the poles are at distinct points and cannot cancel. Thus HTF-NonCancel holds in this explicit model in the strongest possible sense: each off-unit-circle eigenmode would force a genuine interior pole.

Pole barrier as a spectral-radius statement. If one replaces (38) by a hypothetical mode with $|\lambda| > 1$, then $r = \lambda^{-1}$ lies strictly inside $|r| < 1$ and (39) would contain an interior pole. Equivalently, the coefficient sequence would grow like $|b_n| \sim |\lambda|^n$, and the Abel series would have radius of convergence $1/|\lambda| < 1$. Therefore, demanding unit-disk holomorphy forces $|\lambda| \leq 1$, and applying the same argument to the dual mode (or using the functional equation symmetry) forces $|\lambda| = 1$. This is exactly the pole-barrier mechanism in its simplest fully explicit form.

F Wigner–Smith time delay as an interface for measuring computational lapse

If computational lapse is interpreted as an operational time-delay density, then scattering theory provides a natural measurable proxy. Given a unitary scattering matrix $S(E)$ as a function of energy E, the Wigner–Smith time-delay matrix is

$$Q(E) = -\mathrm{i}\, S(E)^\dagger \frac{\mathrm{d}S}{\mathrm{d}E}. \tag{40}$$

A common scalar summary is the total delay

$$\tau_{\mathrm{WS}}(E) = \mathrm{Tr} Q(E). \tag{41}$$

In the present framework, $\tau_{\mathrm{WS}}(E)$ can be interpreted as a measurable time-delay proxy for routing overhead at energy/scale E. Choosing a reference tick duration τ_0 defines a dimensionless overhead proxy

$$\kappa_{\mathrm{WS}}(E) := \frac{\tau_{\mathrm{WS}}(E)}{\tau_0}, \qquad \mathcal{N}_{\mathrm{WS}}(E) := \frac{\kappa_0}{\kappa_{\mathrm{WS}}(E)}.$$

Standard references for time delay include [62, 63].

Remark F.1 (What $\kappa_{\mathrm{WS}}(E)$ is (and is not))**.** *The proxy $\kappa_{\mathrm{WS}}(E)$ is an* energy/scale-dependent *delay observable derived from a scattering measurement. It is not, by itself, the same object as the spatial compilation field $\kappa(x; G_{\mathrm{phys}}, \pi_n, \mathcal{G})$ (Definition 7.2). Connecting the two requires an additional experimental/modeling identification specifying how the measured scattering channel probes localized degrees of freedom and how the energy scale E corresponds to a protocol resolution band.*

F.1 Protocol: from measured S-parameters to κ_{WS}

The definition (40) is operational once $S(E)$ is available as a calibrated function of energy (or frequency). In many experiments one measures S as a function of angular frequency ω or frequency f. Writing $E = \hbar\omega$, one has

$$\frac{\mathrm{d}S}{\mathrm{d}E} = \frac{1}{\hbar}\frac{\mathrm{d}S}{\mathrm{d}\omega},$$

so an equivalent formulation is $Q(\omega) = -(\mathrm{i}/\hbar)\, S(\omega)^\dagger(\mathrm{d}S/\mathrm{d}\omega)$. The following steps define a minimal, auditable pipeline.

Step 1: acquire and calibrate S. Measure the complex scattering matrix elements on a grid of energies/frequencies. Calibration should remove systematic phase delays from the measurement chain and fix a reference plane so that S represents the sample/device.

Step 2: diagnose (approximate) unitarity and losses. Equation (40) is standard for unitary S (lossless elastic scattering). In realistic settings absorption, dissipation, or imperfect calibration can make S non-unitary. Operationally one should report a unitarity diagnostic (e.g. $\|S^\dagger S - I\|$ in multi-channel settings or $|S(E)|$ in the 1-channel setting) and either: (i) restrict attention to bands where unitarity holds to a stated tolerance, or (ii) interpret τ_{WS} as an effective delay proxy under a stated loss model.

Step 3: set derivative resolution and regularize. Time-delay estimates differentiate noisy data, so $\mathrm{d}S/\mathrm{d}E$ must be regularized. A minimal method is a central finite difference on a uniform grid: for sampled energies E_k with step ΔE,

$$\left.\frac{\mathrm{d}S}{\mathrm{d}E}\right|_{E_k} \approx \frac{S(E_{k+1}) - S(E_{k-1})}{2\Delta E}.$$

In the 1-channel case one may equivalently unwrap the phase $\delta(E) = \arg S(E)$ and compute $\tau_{\mathrm{WS}}(E) = \mathrm{d}\delta/\mathrm{d}E$, which avoids branch-cut artifacts at $\pm\pi$. If smoothing is used (moving average, local polynomial fit), the window size should be reported because it sets the effective resolution.

Step 4: compute Q and τ_{WS}. With a derivative estimate in hand, evaluate $Q(E)$ by (40) and take the trace (41). In multi-channel settings, one may additionally inspect eigenvalues of $Q(E)$ (proper delay times) and report their distribution; the trace is the simplest scalar compression.

Step 5: normalize to κ_{WS} and report stability. Choose a reference tick duration τ_0 and define $\kappa_{\mathrm{WS}}(E) = \tau_{\mathrm{WS}}(E)/\tau_0$. Because τ_{WS} is a derivative estimate, uncertainty scales roughly like $\delta\tau_{\mathrm{WS}} \sim \delta S/\Delta E$ (or $\delta\delta/\Delta E$ in the phase method). A minimal report should therefore include the sampling step ΔE (or Δf), the smoothing window, and a basic stability check under moderate changes of ΔE and smoothing window.

Script interface. The script `scripts/exp_wigner_smith_kappa.py` implements the central-difference/phase-unwrapping pipeline in a dependency-free way. It includes a 1-channel Breit–Wigner toy model for $S(E)$ and supports replacing it by a physical model or experimental data source.

A one-channel Breit–Wigner model (analytic check). For a single-channel elastic scatterer, one may write $S(E) = \mathrm{e}^{\mathrm{i}\delta(E)}$ and the Wigner–Smith delay reduces to the phase derivative $\tau_{\mathrm{WS}}(E) = \mathrm{d}\delta/\mathrm{d}E$. A minimal exactly unitary resonance model is the Breit–Wigner form

$$S(E) = \frac{E - E_0 - \mathrm{i}\gamma/2}{E - E_0 + \mathrm{i}\gamma/2}, \tag{42}$$

whose phase rises by π across the resonance. In this case one obtains the Lorentzian delay profile

$$\tau_{\mathrm{WS}}(E) = \frac{\gamma}{(E - E_0)^2 + (\gamma/2)^2}. \tag{43}$$

The toy implementation in `scripts/exp_wigner_smith_kappa.py` computes $\tau_{\mathrm{WS}}(E)$ numerically via a phase-unwrapped finite difference and provides an auditable interface for replacing (42) by a physical $S(E)$.

F.2 A minimal device-level realization (Hamiltonian, lead coupling, and tick calibration)

To make the Wigner–Smith interface concrete at the level requested by experimental/numerical tests, we record a minimal open-system model that produces (42) and therefore an explicit κ_{WS}.

Device Hamiltonian and coupling (one level, one channel). Let the device have a single internal mode $|0\rangle$ with Hamiltonian

$$H_{\mathrm{dev}} = E_0 \, |0\rangle\langle 0| \, .$$

Couple this mode to a single propagation channel (lead) in the standard wide-band approximation so that the retarded self-energy is purely imaginary, $\Sigma(E) \approx -\mathrm{i}\,\gamma/2$ with $\gamma > 0$. Equivalently, the effective non-Hermitian Hamiltonian is

$$H_{\mathrm{eff}} := H_{\mathrm{dev}} - \mathrm{i}\,\frac{\gamma}{2}\,|0\rangle\langle 0|\,.$$

Then the on-shell scattering matrix is unitary and equals the Breit–Wigner form (42). In this realization, the linewidth γ is the coupling-induced decay rate (inverse lifetime) of the internal mode.

Tick period and the dimensionless overhead proxy. To compare with compilation overhead (which is measured in discrete ticks), fix a reference tick duration τ_0 in seconds. Operationally, τ_0 is the duration of one primitive substrate tick, i.e. the time required to execute a chosen reference primitive on the physical platform (or, in a wave network, the calibrated delay of a single standard cell). Then

$$\kappa_{\mathrm{WS}}(E) = \frac{\tau_{\mathrm{WS}}(E)}{\tau_0}$$

is a dimensionless, directly measurable delay/overhead proxy.

Resonance-height normalization (closed form). For (43) one has, at the resonance energy $E = E_0$,

$$\tau_{\mathrm{WS}}(E_0) = \frac{4}{\gamma}, \qquad \kappa_{\mathrm{WS}}(E_0) = \frac{4}{\gamma\,\tau_0}. \tag{44}$$

Thus, in this minimal model the overhead proxy is proportional to the inverse linewidth. This provides a clean experimental/numerical handle: γ can be extracted from the measured phase jump or delay peak width, and (44) then yields $\kappa_{\mathrm{WS}}(E_0)$.

F.3 A testable map from $\kappa_{\mathrm{WS}}(E)$ to spatial $\kappa(x)$ (calibrated interface)

The spatial field $\kappa(x)$ is defined by compilation depth on a hardware graph (Definition 7.2), whereas $\kappa_{\mathrm{WS}}(E)$ is defined by a scattering delay measurement. To relate them in a concrete platform one must specify a *localization map* (how a spatial location x is addressed by a scattering channel) and a *band map* (how an energy/frequency band corresponds to a protocol resolution/time scale). We record a minimal calibrated interface that is sufficient for quantitative tests.

Assumption F.2 (Localized resonance probe (calibrated delay-to-depth identification))**.** *For each coarse-grained spatial location x (or for each probe region centered at x at readout scale ε), there exists an experimentally realizable scattering configuration with:*

- *a localized resonance centered at energy $E_0(x)$ with extracted linewidth $\gamma(x)$;*
- *a fixed platform tick duration τ_0 (calibration);*
- *a fixed local-update clock task family used to define $\kappa(x)$.*

The calibrated identification is

$$\kappa(x) \;\approx\; \kappa_{\mathrm{WS}}(E_0(x)) \;=\; \frac{4}{\gamma(x)\,\tau_0}, \tag{45}$$

up to a location-independent constant factor that depends only on the chosen clock task normalization (cf. Proposition 7.7) and on the probe geometry.

Remark F.3 (Quantitative test form)**.** *Under Assumption F.2, the lapse dictionary predicts*

$$N(x) = \frac{\kappa_0}{\kappa(x)} \;\approx\; \frac{\kappa_0}{\kappa_{\mathrm{WS}}(E_0(x))} = \frac{\kappa_0\,\gamma(x)\,\tau_0}{4},$$

so lapse ratios are directly expressed in terms of measurable linewidth ratios:

$$\frac{N(x_1)}{N(x_2)} \approx \frac{\gamma(x_1)}{\gamma(x_2)}.$$

This gives a concrete experimental/numerical falsifiability channel: one extracts $\gamma(x)$ (or, more generally, $\tau_{\mathrm{WS}}(E)$) from measured S-data and compares the induced lapse field (or its ratios) against the gravitational templates of Section 7.

F.4 Candidate experimental platforms (example)

The Wigner–Smith interface is most natural in platforms where a scattering description is already standard. One concrete example is a microwave cavity / microwave network experiment in which $S(f)$ is measured directly as complex S-parameters over a frequency band; the group delay and resonance widths are extracted from phase evolution and delay peaks. Another common arena is mesoscopic transport, where multi-channel scattering matrices and time delays arise naturally. In either case, the minimal protocol is the same: measure S on a frequency grid, calibrate, differentiate with stated regularization, and report τ_{WS} and κ_{WS} with uncertainty diagnostics.

F.5 Relation to spatial overhead $\kappa(x)$

The compilation field $\kappa(x)$ is defined from a spatial task family and a hardware graph, whereas $\kappa_{\mathrm{WS}}(E)$ is defined from a spectral/time-delay measurement. To map between them in a concrete system one must specify how the scattering channel probes localized degrees of freedom and how the energy scale E corresponds to a protocol resolution band. Operationally, one may use $\kappa_{\mathrm{WS}}(E)$ as a calibrated overhead proxy and compare redshift-type ratios across energies/bands, or use it as an input to fit an effective lapse profile once a spatial identification is supplied by the experimental model.

Directly measurable observables. In laboratory settings where a scattering description is available, the most direct measurable objects are the scattering matrix elements (e.g. microwave/mesoscopic S-parameters) as functions of energy/frequency. From these one extracts:

- the phase $\arg S(E)$ (or multi-channel eigenphases) and its energy derivative (group delay);
- resonance locations E_0 and linewidths γ via fits of phase jumps or delay peaks;
- in the multi-channel case, the full time-delay matrix $Q(E)$ and its trace $\tau_{\mathrm{WS}}(E) = \mathrm{Tr}Q(E)$.

Fixing a reference tick duration τ_0 turns the measured delay into the dimensionless overhead proxy $\kappa_{\mathrm{WS}}(E) = \tau_{\mathrm{WS}}(E)/\tau_0$, and hence into a lapse proxy $\mathcal{N}_{\mathrm{WS}}(E) = \kappa_0/\kappa_{\mathrm{WS}}(E)$.

G A complete 1+1D worked example: weighted scan-chain lapse matching

This appendix provides a fully explicit 1+1-dimensional worked example (one spatial lattice dimension plus scan time) that closes the micro-to-macro loop required for quantitative testing: *addressing → hardware graph → routing/compilation depth $\kappa(x)$ → lapse $N(x) = \kappa_0/\kappa(x)$ → coarse-grained readout and comparison against a weak-field template.*

G.1 Microscopic specification: address, hardware graph, primitives, and a local clock task

Screen (1D) at resolution n. Let $\Sigma_n := \{0, 1, \dots, L-1\}$ with $L := 2^n$ and nearest-neighbor adjacency $i \sim i \pm 1$ (with endpoints treated as boundaries).

Addressing and placement. Fix the trivial address map $A_n(t) = t$ and the identity placement

$$\pi_n(i) := i \in V_{\text{phys}}.$$

In this example, addressing is explicit and routing is driven entirely by the (weighted) hardware constraint rather than by a nontrivial folding permutation.

Hardware interaction graph. Take the weighted path graph

$$V_{\text{phys}} = \{0, 1, \dots, L-1\}, \qquad E_{\text{phys}} = \big\{\{i, i+1\} : 0 \le i \le L-2\big\},$$

with integer edge weights $w_{i+\frac{1}{2}} := w(\{i, i+1\}) \in \mathbb{Z}_{>0}$ interpreted as tick costs (Assumption 7.1).

Local clock task (two-edge update). Fix a bounded-local task family $\mathcal{G} = \{\mathcal{G}_i\}_{i \in \Sigma_n}$ as follows. For interior sites $1 \le i \le L-2$, the task $\mathcal{G}_i$ consists of executing a prescribed two-body primitive on the left edge $\{i-1, i\}$ and on the right edge $\{i, i+1\}$ once each (the specific gate is irrelevant for depth accounting). Because the two edges share the vertex i, the disjoint-support scheduling constraint forces sequential execution, so the minimal depth is exactly

$$\kappa(i) = \text{Depth}_{G_{\text{phys}}}(\mathcal{G}_i) = w_{i-\frac{1}{2}} + w_{i+\frac{1}{2}}, \qquad 1 \le i \le L-2. \tag{46}$$

This gives a completely explicit and auditable $\kappa(i)$.

Lapse and potential. Fix a reference overhead $\kappa_0 > 0$ and define

$$N(i) := \frac{\kappa_0}{\kappa(i)}, \qquad \Phi(i) := \log\!\left(\frac{\kappa(i)}{\kappa_0}\right) = -\log N(i).$$

By Proposition 7.3, $N(i)$ is the operational lapse linking global scan ticks to local relational time in this concrete model.

G.2 Weak-field target and construction of a matching hardware profile

Target lapse (weak-field Schwarzschild template along a radial line). To compare against a standard weak-field template used throughout Section 7, we take the Schwarzschild lapse

$$N_{\text{Schw}}(r) = \sqrt{1 - \frac{2M}{r}} \qquad (c = G = 1), \tag{47}$$

restricted to a radial interval $r \in [r_{\min}, r_{\max}]$ with $r_{\min} > 2M$ so that N_{Schw} is real and close to 1. In the worked numerics below we use a weak-field choice with $2M/r_{\min} \ll 1$.

Grid embedding. Embed the 1D screen sites into the radial interval by

$$r_i := r_{\min} + i\,h, \qquad h := \frac{r_{\max} - r_{\min}}{L-1}, \qquad 0 \le i \le L-1,$$

and midpoints $r_{i+\frac{1}{2}} := r_i + \frac{h}{2}$ for $0 \le i \le L-2$.

Target overhead and midpoint edge weights. Define the target overhead field

$$\kappa_{\mathrm{tar}}(r) := \frac{\kappa_0}{N_{\mathrm{Schw}}(r)}.$$

We choose the hardware edge weights by midpoint sampling:

$$w_{i+\frac{1}{2}} := \left\lfloor \frac{1}{2}\,\kappa_{\mathrm{tar}}(r_{i+\frac{1}{2}}) + \frac{1}{2} \right\rfloor, \qquad 0 \le i \le L-2. \tag{48}$$

This choice is constructive, integer-valued (as required by Assumption 7.1), and yields a second-order accurate site overhead by symmetric averaging.

Proposition G.1 (Second-order matching)**.** *Assume $\kappa_{\mathrm{tar}} \in C^2([r_{\min}, r_{\max}])$. Define $\kappa(i)$ for interior sites by (46)–(48). Define the induced lapse by $N(i) = \kappa_0/\kappa(i)$. Then for interior sites $1 \le i \le L-2$,*

$$\kappa(i) = \kappa_{\mathrm{tar}}(r_i) + O(h^2) + O(1), \qquad N(i) = N_{\mathrm{Schw}}(r_i) + O(h^2) + O\Big(\frac{1}{\kappa_0}\Big),$$

where the $O(1)$ term comes from integer rounding of edge weights and is uniformly bounded independent of h. In particular, at fixed κ_0 large enough that rounding is negligible, the mismatch decays as $O(h^2)$ under refinement.

Proof sketch. Ignoring the integer rounding in (48), one has

$$w_{i\pm\frac{1}{2}} \approx \frac{1}{2}\,\kappa_{\mathrm{tar}}(r_i \pm h/2), \qquad \kappa(i) = w_{i-\frac{1}{2}} + w_{i+\frac{1}{2}} \approx \frac{1}{2}\Big(\kappa_{\mathrm{tar}}(r_i - h/2) + \kappa_{\mathrm{tar}}(r_i + h/2)\Big).$$

Taylor expansion shows the symmetric average equals $\kappa_{\mathrm{tar}}(r_i) + \frac{h^2}{8}\kappa''_{\mathrm{tar}}(r_i) + O(h^4)$, hence the $O(h^2)$ term. Rounding each edge weight introduces an error bounded by $1/2$ tick per edge, hence an $O(1)$ error in $\kappa(i)$. Finally, $N(i) = \kappa_0/\kappa(i)$ and smoothness of N_{Schw} on $[r_{\min}, r_{\max}]$ transfer the scaling to lapse errors, with an additional $O(1/\kappa_0)$ quantization term under fixed physical calibration. □

G.3 Readout at finite resolution

To model finite-resolution measurement, we read out coarse-grained fields using a nonnegative normalized kernel $w^{(\varepsilon)}$ of width ε (cf. Section 5.2 and Definition 5.8). On the 1D grid, a concrete discrete implementation is the windowed average

$$\Phi_\varepsilon(i) := \sum_{j=0}^{L-1} w_{ij}^{(\varepsilon)}\,\Phi(j), \qquad \sum_j w_{ij}^{(\varepsilon)} = 1, \quad w_{ij}^{(\varepsilon)} \ge 0,$$

with $w_{ij}^{(\varepsilon)}$ supported mainly on $|r_j - r_i| \lesssim \varepsilon$. The induced coarse lapse is $N_\varepsilon(i) = \exp(-\Phi_\varepsilon(i))$.

G.4 Numerical pipeline and generated tables

The script `scripts/exp_1p1d_routing_worked_example.py` implements the construction (48), computes $\kappa(i)$, produces the induced lapse $N(i)$, and compares it to the weak-field target (47). It reports finite-size errors across multiple resolutions and a simple log–log scaling fit.

order n	sites L	h	$\max_i \lvert\Delta N_i\rvert$	RMS($\lvert\Delta N\rvert$)	rel. RMS
6	64	0.111111	1.270e-04	2.535e-05	2.642e-05
7	128	0.0551181	3.668e-05	6.691e-06	6.982e-06
8	256	0.027451	9.882e-06	1.718e-06	1.794e-06
9	512	0.0136986	2.566e-06	4.351e-07	4.545e-07
10	1024	0.00684262	6.536e-07	1.095e-07	1.144e-07
11	2048	0.00341964	1.647e-07	2.748e-08	2.871e-08

Table 13: 1+1D worked example: induced lapse $N(i) = \kappa_0/\kappa(i)$ vs. the weak-field target $N_{\mathrm{Schw}}(r_i)$, with finite-size error metrics evaluated on interior sites.

fit range (orders)	slope for RMS vs. h	R^2
6–11	1.9642	0.99993

Table 14: Finite-size scaling fit for the 1+1D worked example. The midpoint construction yields second-order convergence in h up to the integer-quantization floor.

H Reproducible experiment code

H.1 Experiment A: discrete Hilbert addressing locality check

```
# -*- coding: utf-8 -*-
"""
Discrete 2D Hilbert curve addressing: index -> (x,y), and locality verification.

This script checks the one-step Manhattan locality:
  ||H_n(t+1) - H_n(t)||_1 = 1
for orders n = 1..8, and writes a small LaTeX row file into sections/generated/.
"""

from __future__ import annotations

from pathlib import Path


def _rot(s: int, x: int, y: int, rx: int, ry: int) -> tuple[int, int]:
    """Rotate/flip a quadrant (standard Hilbert helper)."""
    if ry == 0:
        if rx == 1:
            x = s - 1 - x
            y = s - 1 - y
        x, y = y, x
    return x, y


def hilbert_d2xy(order: int, d: int) -> tuple[int, int]:
    """
    2D Hilbert mapping: d in [0, 2^(2*order)-1] -> (x,y) in [0, 2^order-1]^2.
    """
    n = 1 << order
    x = 0
    y = 0
    t = int(d)
    s = 1
```

r	$N_{\mathrm{Schw}}(r)$	$N(r)$ (measured)	ΔN
1.0137	0.9493952556	0.9493926892	-2.566e-06
1.64384	0.9691061173	0.9691055461	-5.713e-07
2.28767	0.9778994964	0.9778992893	-2.072e-07
2.91781	0.9827144547	0.9827143562	-9.856e-08
3.54795	0.9858066097	0.9858065554	-5.438e-08
4.17808	0.9879603098	0.9879602764	-3.339e-08
4.82192	0.9895763556	0.9895763343	-2.134e-08
5.45205	0.9907867033	0.9907866885	-1.486e-08
6.08219	0.9917452085	0.9917451979	-1.055e-08
6.71233	0.9925230682	0.9925230598	-8.374e-09
7.35616	0.9931797214	0.9931797157	-5.705e-09
7.9863	0.9937195576	0.9937195533	-4.308e-09

Table 15: Representative lapse curve samples from the 1+1D worked example at a fixed order (script default).

```
    while s < n:
        rx = 1 & (t // 2)
        ry = 1 & (t ^ rx)
        x, y = _rot(s, x, y, rx, ry)
        x += s * rx
        y += s * ry
        t //= 4
        s *= 2
    return x, y


def manhattan(p: tuple[int, int], q: tuple[int, int]) -> int:
    return abs(p[0] - q[0]) + abs(p[1] - q[1])


def check_order(order: int) -> tuple[int, int]:
    n = 1 << order
    total = n * n
    pts = [hilbert_d2xy(order, d) for d in range(total)]
    dists = [manhattan(pts[i], pts[i + 1]) for i in range(total - 1)]
    return min(dists), max(dists)


def write_rows(rows: list[tuple[int, int, int]]) -> None:
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)
    out_path = out_dir / "hilbert_locality_rows.tex"

    lines = []
    for order, mn, mx in rows:
        lines.append(f"{order} & {mn} & {mx} \\\\")
    out_path.write_text("\n".join(lines) + "\n", encoding="utf-8")


def main() -> None:
    rows: list[tuple[int, int, int]] = []
    for order in range(1, 9):
        mn, mx = check_order(order)
```

```
        rows.append((order, mn, mx))
        print(f"order={order}: min_L1={mn}, max_L1={mx}")
        if mn != 1 or mx != 1:
            raise AssertionError("Hilbert one-step locality violated.")
    write_rows(rows)
    print("Wrote sections/generated/hilbert_locality_rows.tex")


if __name__ == "__main__":
    main()
```

H.2 Experiment B: golden-branch star discrepancy bound check

```
# -*- coding: utf-8 -*-
"""
Golden-branch Kronecker scan star discrepancy and an explicit logarithmic bound.

Computes the exact 1D star discrepancy D*_N for P_N = {x0 + t*alpha mod 1}_{t=0}^{N-1}
with alpha = 1/phi, and compares against:
  D*_N <= 2(2 + log_phi N)/N.

Writes a LaTeX row file into sections/generated/golden_discrepancy_rows.tex.
"""

from __future__ import annotations

import math
from pathlib import Path


def kronecker_points(alpha: float, N: int, x0: float) -> list[float]:
    return [((x0 + t * alpha) % 1.0) for t in range(N)]


def star_discrepancy_1d(points: list[float]) -> float:
    xs = sorted(points)
    N = len(xs)
    # For intervals [0,u): D* = max(max_i ((i+1)/N - x_i), max_i (x_i - i/N))
    d_plus = max((i + 1) / N - xs[i] for i in range(N))
    d_minus = max(xs[i] - i / N for i in range(N))
    return max(d_plus, d_minus)


def golden_bound(N: int) -> float:
    phi = (1.0 + 5.0**0.5) / 2.0
    return 2.0 * (2.0 + (math.log(N) / math.log(phi))) / N


def write_rows(rows: list[tuple[int, float, float, float]]) -> None:
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)
    out_path = out_dir / "golden_discrepancy_rows.tex"

    lines = []
    for N, D, bound, ratio in rows:
```

```
        lines.append(f"{N} & {D:.6g} & {bound:.6g} & {ratio:.3f} \\\\")
    out_path.write_text("\n".join(lines) + "\n", encoding="utf-8")


def main() -> None:
    phi = (1.0 + 5.0**0.5) / 2.0
    alpha = 1.0 / phi
    x0 = 0.1

    Ns = [100, 300, 1000, 3000, 10000, 30000]
    rows: list[tuple[int, float, float, float]] = []

    for N in Ns:
        pts = kronecker_points(alpha, N, x0=x0)
        D = star_discrepancy_1d(pts)
        bound = golden_bound(N)
        ratio = D / bound if bound > 0 else float("nan")
        rows.append((N, D, bound, ratio))
        print(f"N={N:6d}  D*={D:.6g}  bound={bound:.6g}  ratio={ratio:.3f}")

    write_rows(rows)
    print("Wrote sections/generated/golden_discrepancy_rows.tex")


if __name__ == "__main__":
    main()
```

H.3 Experiment C: Abel pole-barrier toy model

```
# -*- coding: utf-8 -*-
"""
Toy Abel pole-barrier / threshold experiment.

We consider the mode:
  u_t = exp((beta - 1/2) t) * cos(gamma t),
and its Abel-weighted partial sum:
  S(r;T) = sum_{t=0}^{T-1} r^t u_t.

When beta > 1/2, the effective growth factor is r * exp(beta - 1/2).
The threshold radius is:
  r_c = exp(-(beta - 1/2)).

This script evaluates |S(r;T_max)| across r values around r_c and writes
sections/generated/abel_barrier_rows.tex for the paper table.
"""

from __future__ import annotations

import math
from pathlib import Path


def abel_mode_partial_sum(r: float, beta: float, gamma: float, T_max: int) -> float:
    lam = beta - 0.5
    s = 0.0
    for t in range(T_max):
```

```
        s += (r**t) * math.exp(lam * t) * math.cos(gamma * t)
    return s


def critical_radius(beta: float) -> float:
    return math.exp(-(beta - 0.5))


def write_rows(rows: list[tuple[float, float, float]]) -> None:
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)
    out_path = out_dir / "abel_barrier_rows.tex"

    lines = []
    for beta, r, val in rows:
        lines.append(f"{beta:.3f} & {r:.4f} & {val:.6e} \\\\")
    out_path.write_text("\n".join(lines) + "\n", encoding="utf-8")


def main() -> None:
    gamma = 1.0
    T_max = 5000

    betas = [0.5, 0.6]
    rows: list[tuple[float, float, float]] = []

    for beta in betas:
        rc = critical_radius(beta)
        print(f"beta={beta:.3f}  critical radius r_c={rc:.6f}")

        # Choose r values around the threshold.
        r_values = [0.80, 0.88, 0.90, 0.905, 0.92, 0.95, 0.98]
        for r in r_values:
            s = abel_mode_partial_sum(r, beta=beta, gamma=gamma, T_max=T_max)
            mag = abs(s)
            rows.append((beta, r, mag))
            print(f"  r={r:.4f}  |S|={mag:.6e}")

    write_rows(rows)
    print("Wrote sections/generated/abel_barrier_rows.tex")


if __name__ == "__main__":
    main()
```

H.4 Experiment D: Poisson phase potential via FFT

```
# -*- coding: utf-8 -*-
"""
Poisson solver in 3D (periodic box) for a point source.

We solve:
  -Delta Phi = 4*pi rho
on a periodic cube using either:
  (A) an FFT-based solver (if numpy is available), or
```

```
  (B) a pure-Python iterative Jacobi/SOR fallback (if numpy is unavailable).

We set the zero mode to 0 (zero-mean gauge) by subtracting the mean of rho.

The script outputs radial shell averages <Phi>(r) and writes a small LaTeX row
file into sections/generated/poisson_rows.tex.

It also reports simple finite-size and source-width robustness statistics for the
approximate 1/r window by checking the flatness of r*<Phi>(r) over an intermediate
radius band, and writes sections/generated/poisson_scaling_rows.tex.
"""

from __future__ import annotations

import math
from pathlib import Path

try:
    import numpy as np  # type: ignore
except ModuleNotFoundError:
    np = None  # type: ignore


def poisson_solve_fft(rho: "np.ndarray", L: float = 1.0) -> "np.ndarray":
    """
    Solve -Delta Phi = 4*pi rho on a periodic cube [0,L)^3 using FFT.
    The k=0 mode is set to 0 (zero-mean gauge).
    """
    if np is None:
        raise RuntimeError("numpy is required for the FFT solver.")

    N = int(rho.shape[0])
    assert rho.shape == (N, N, N)

    # Ensure solvability in the periodic setting by removing the mean (zero mode).
    rho = rho - float(rho.mean())
    rho_k = np.fft.fftn(rho)

    k = 2 * np.pi * np.fft.fftfreq(N, d=L / N)
    kx, ky, kz = np.meshgrid(k, k, k, indexing="ij")
    k2 = kx * kx + ky * ky + kz * kz

    phi_k = np.zeros_like(rho_k, dtype=np.complex128)
    mask = k2 != 0.0
    phi_k[mask] = (4 * np.pi) * rho_k[mask] / k2[mask]
    phi_k[~mask] = 0.0

    phi = np.fft.ifftn(phi_k).real
    return phi


def radial_shell_average_numpy(phi: "np.ndarray", r_max: int = 12):
    """
    Return shell averages for integer radii r=1..r_max (in lattice steps).
    """
    if np is None:
        raise RuntimeError("numpy is required for the numpy radial averaging.")
```

```
    N = int(phi.shape[0])
    assert phi.shape == (N, N, N)

    center = np.array([N // 2, N // 2, N // 2], dtype=float)
    coords = np.indices((N, N, N), dtype=float).reshape(3, -1).T
    r = np.linalg.norm(coords - center, axis=1)
    phi_flat = phi.reshape(-1)

    out = []
    for rad in range(1, r_max + 1):
        shell = (r >= rad - 0.5) & (r < rad + 0.5)
        if int(shell.sum()) == 0:
            continue
        mean = float(phi_flat[shell].mean())
        out.append((rad, mean))
    return out


def poisson_solve_jacobi_periodic(
    N: int,
    rho: list[float],
    max_iter: int = 4000,
    omega: float = 0.9,
) -> list[float]:
    """
    Pure-Python periodic Poisson solver for -Delta phi = 4*pi rho using damped
    ↪  Jacobi/SOR.

    The grid spacing is 1. rho is a flat list of length N^3.
    """
    n3 = N * N * N
    if len(rho) != n3:
        raise ValueError("rho must have length N^3.")

    # Enforce solvability by subtracting mean (zero mode).
    mean_rho = sum(rho) / n3
    rho = [v - mean_rho for v in rho]

    phi = [0.0] * n3
    phi_new = [0.0] * n3

    for _it in range(max_iter):
        for i in range(N):
            ip = (i + 1) % N
            im = (i - 1) % N
            for j in range(N):
                jp = (j + 1) % N
                jm = (j - 1) % N
                base_ip = ip * N * N
                base_im = im * N * N
                base_i = i * N * N
                base_jp = jp * N
                base_jm = jm * N
                base_j = j * N
                for k in range(N):
                    kp = (k + 1) % N
                    km = (k - 1) % N
                    p = base_i + base_j + k
```

```
                    neigh = (
                        phi[base_ip + base_j + k]
                        + phi[base_im + base_j + k]
                        + phi[base_i + base_jp + k]
                        + phi[base_i + base_jm + k]
                        + phi[base_i + base_j + kp]
                        + phi[base_i + base_j + km]
                    )

                    # Discrete stencil for -Delta phi = 6 phi - sum(neigh) = 4*pi rho
                    candidate = (neigh + 4.0 * math.pi * rho[p]) / 6.0
                    phi_new[p] = (1.0 - omega) * phi[p] + omega * candidate

        phi, phi_new = phi_new, phi

    # Gauge-fix to zero mean.
    mean_phi = sum(phi) / n3
    phi = [v - mean_phi for v in phi]
    return phi


def poisson_solve_jacobi_dirichlet(
    N: int,
    rho: list[float],
    max_iter: int = 12000,
    omega: float = 0.9,
) -> list[float]:
    """
    Pure-Python Dirichlet Poisson solver for -Delta phi = 4*pi rho on a cube with
    boundary condition phi=0 on the boundary.

    The grid spacing is 1. rho is a flat list of length N^3.
    """
    n3 = N * N * N
    if len(rho) != n3:
        raise ValueError("rho must have length N^3.")

    def idx(i: int, j: int, k: int) -> int:
        return (i * N + j) * N + k

    phi = [0.0] * n3
    phi_new = [0.0] * n3

    # Keep boundaries pinned to 0; update only interior sites (weighted Jacobi).
    for _it in range(max_iter):
        for i in range(1, N - 1):
            for j in range(1, N - 1):
                for k in range(1, N - 1):
                    p = idx(i, j, k)
                    neigh = (
                        phi[idx(i + 1, j, k)]
                        + phi[idx(i - 1, j, k)]
                        + phi[idx(i, j + 1, k)]
                        + phi[idx(i, j - 1, k)]
                        + phi[idx(i, j, k + 1)]
                        + phi[idx(i, j, k - 1)]
                    )
```

```
                    candidate = (neigh + 4.0 * math.pi * rho[p]) / 6.0
                    phi_new[p] = (1.0 - omega) * phi[p] + omega * candidate

        # Boundaries stay 0.
        phi, phi_new = phi_new, phi

    return phi


def radial_shell_average_pure(phi: list[float], N: int, r_max: int = 12) ->
↪  list[tuple[int, float]]:
    """
    Pure-Python shell averages for integer radii r=1..r_max (in lattice steps).
    """
    n3 = N * N * N
    if len(phi) != n3:
        raise ValueError("phi must have length N^3.")

    cx = N // 2
    cy = N // 2
    cz = N // 2

    sums = [0.0] * (r_max + 1)
    counts = [0] * (r_max + 1)

    for i in range(N):
        dx = i - cx
        for j in range(N):
            dy = j - cy
            for k in range(N):
                dz = k - cz
                r = (dx * dx + dy * dy + dz * dz) ** 0.5
                rad = int(round(r))
                if 1 <= rad <= r_max and abs(r - rad) < 0.5:
                    p = (i * N + j) * N + k
                    sums[rad] += phi[p]
                    counts[rad] += 1

    out: list[tuple[int, float]] = []
    for rad in range(1, r_max + 1):
        if counts[rad] == 0:
            continue
        out.append((rad, sums[rad] / counts[rad]))
    return out


def write_rows(rows: list[tuple[int, float]]) -> None:
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)
    out_path = out_dir / "poisson_rows.tex"

    lines = []
    for r, mean in rows:
        lines.append(f"{r} & {mean:+.6f} & {(r*mean):+.6f} \\\\")
    out_path.write_text("\n".join(lines) + "\n", encoding="utf-8")
```

```
def flatness_metric(stats: list[tuple[int, float]], r_min: int, r_max: int) ->
↪  tuple[float, float, int]:
    """
    Given shell averages (r, <Phi>(r)), compute mean and relative RMS of r*<Phi>(r)
    over the window r in [r_min, r_max]. Returns (mean, rel_rms, count).
    """
    vals: list[float] = []
    for r, mean in stats:
        if r_min <= r <= r_max:
            vals.append(float(r) * float(mean))
    if not vals:
        return 0.0, float("inf"), 0
    m = sum(vals) / float(len(vals))
    rms = (sum((v - m) ** 2 for v in vals) / float(len(vals))) ** 0.5
    rel = rms / abs(m) if m != 0.0 else float("inf")
    return m, rel, len(vals)


def write_scaling_rows(rows: list[str]) -> None:
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)
    out_path = out_dir / "poisson_scaling_rows.tex"
    out_path.write_text("\n".join(rows) + "\n", encoding="utf-8")


def build_rho_cube_pure(N: int, radius: int) -> list[float]:
    """
    A simple compact source: uniform mass on a (2*radius+1)^3 cube centered at N//2.
    """
    n3 = N * N * N
    rho = [0.0] * n3
    cx = cy = cz = N // 2
    total = 0
    for i in range(cx - radius, cx + radius + 1):
        for j in range(cy - radius, cy + radius + 1):
            for k in range(cz - radius, cz + radius + 1):
                if 0 <= i < N and 0 <= j < N and 0 <= k < N:
                    p = (i * N + j) * N + k
                    rho[p] = 1.0
                    total += 1
    if total > 0:
        rho = [v / float(total) for v in rho]
    return rho


def build_rho_cube_numpy(N: int, radius: int) -> "np.ndarray":
    if np is None:
        raise RuntimeError("numpy is required.")
    rho = np.zeros((N, N, N), dtype=float)
    c = N // 2
    for i in range(c - radius, c + radius + 1):
        for j in range(c - radius, c + radius + 1):
            for k in range(c - radius, c + radius + 1):
                if 0 <= i < N and 0 <= j < N and 0 <= k < N:
                    rho[i, j, k] = 1.0
    s = float(rho.sum())
    if s > 0.0:
```

```
        rho /= s
    return rho


def main() -> None:
    r_max = 12

    if np is not None:
        N = 64
        L = 1.0

        rho = build_rho_cube_numpy(N, radius=0)

        phi = poisson_solve_fft(rho, L=L)
        stats = radial_shell_average_numpy(phi, r_max=r_max)
    else:
        # Pure-Python fallback: smaller grid and iterative solver.
        N = 24
        rho = build_rho_cube_pure(N, radius=0)
        phi_flat = poisson_solve_jacobi_periodic(N=N, rho=rho, max_iter=6000,
        ↪  omega=0.9)
        stats = radial_shell_average_pure(phi_flat, N=N, r_max=r_max)

    print("r | <Phi> | r*<Phi>")
    for r, mean in stats[:12]:
        print(f"{r:2d} | {mean:+.6f} | {(r*mean):+.6f}")

    write_rows(stats[:10])
    print("Wrote sections/generated/poisson_rows.tex")

    # Robustness / finite-size scaling table for the 1/r window.
    scaling_lines: list[str] = []
    window_min = 3

    if np is not None:
        for N in (32, 48, 64):
            for radius in (0, 1):
                rho = build_rho_cube_numpy(N, radius=radius)
                phi = poisson_solve_fft(rho, L=1.0)
                statsN = radial_shell_average_numpy(phi, r_max=min(r_max, N // 3))
                window_max = min(10, N // 4)
                m, rel, cnt = flatness_metric(statsN, r_min=window_min,
                ↪  r_max=window_max)
                scaling_lines.append(
                    f"{N} & periodic-FFT & cube-{radius} & {window_min}--{window_max}
                    ↪  & {cnt} & {m:+.4e} & {rel:.3e} \\\\"
                )

        # Dirichlet comparison (pure-Python SOR) at a modest size.
        N = 32
        for radius in (0, 1):
            rho_pure = build_rho_cube_pure(N, radius=radius)
            phi_flat = poisson_solve_jacobi_dirichlet(N=N, rho=rho_pure,
            ↪  max_iter=12000, omega=0.9)
            statsD = radial_shell_average_pure(phi_flat, N=N, r_max=min(r_max, N //
            ↪  3))
            window_max = min(10, N // 4)
            m, rel, cnt = flatness_metric(statsD, r_min=window_min, r_max=window_max)
```

```
            scaling_lines.append(
                f"{N} & Dirichlet-Jacobi & cube-{radius} & {window_min}--{window_max}
                ↪  & {cnt} & {m:+.4e} & {rel:.3e} \\\\"
            )
    else:
        # Pure-Python only: keep the grid small for runtime.
        for N in (20, 24):
            for radius in (0, 1):
                rho_pure = build_rho_cube_pure(N, radius=radius)
                phi_flat = poisson_solve_jacobi_periodic(N=N, rho=rho_pure,
                ↪  max_iter=7000, omega=0.9)
                statsN = radial_shell_average_pure(phi_flat, N=N, r_max=min(r_max, N
                ↪  // 3))
                window_max = min(8, N // 4)
                m, rel, cnt = flatness_metric(statsN, r_min=window_min,
                ↪  r_max=window_max)
                scaling_lines.append(
                    f"{N} & periodic-Jacobi & cube-{radius} &
                    ↪  {window_min}--{window_max} & {cnt} & {m:+.4e} & {rel:.3e}
                    ↪  \\\\"
                )

        N = 20
        for radius in (0, 1):
            rho_pure = build_rho_cube_pure(N, radius=radius)
            phi_flat = poisson_solve_jacobi_dirichlet(N=N, rho=rho_pure,
            ↪  max_iter=10000, omega=0.9)
            statsD = radial_shell_average_pure(phi_flat, N=N, r_max=min(r_max, N //
            ↪  3))
            window_max = min(8, N // 4)
            m, rel, cnt = flatness_metric(statsD, r_min=window_min, r_max=window_max)
            scaling_lines.append(
                f"{N} & Dirichlet-Jacobi & cube-{radius} & {window_min}--{window_max}
                ↪  & {cnt} & {m:+.4e} & {rel:.3e} \\\\"
            )

    write_scaling_rows(scaling_lines)
    print("Wrote sections/generated/poisson_scaling_rows.tex")


if __name__ == "__main__":
    main()
```

H.5 Experiment E: Wigner–Smith time delay interface

```
# -*- coding: utf-8 -*-
"""
Wigner--Smith time-delay matrix and an overhead proxy (standard-library only).

Given a unitary scattering matrix S(E), define:
  Q(E) = -i S(E)^dagger dS/dE
  tau_WS(E) = Tr Q(E)
  kappa_WS(E) = tau_WS(E) / tau0.

This script provides a finite-difference approximation and a small toy example.
Users can replace the toy S(E) with a model or experimental S(E).
```

```
"""

from __future__ import annotations

import argparse
import cmath
import math
from pathlib import Path
from typing import Callable


Matrix = list[list[complex]]
Scattering = complex | Matrix


def _is_matrix(S: Scattering) -> bool:
    return isinstance(S, list)


def conj_transpose(A: Matrix) -> Matrix:
    n = len(A)
    if n == 0:
        raise ValueError("Empty matrix.")
    m = len(A[0])
    if any(len(row) != m for row in A):
        raise ValueError("Ragged matrix.")
    return [[A[i][j].conjugate() for i in range(n)] for j in range(m)]


def matmul(A: Matrix, B: Matrix) -> Matrix:
    n = len(A)
    if n == 0:
        raise ValueError("Empty matrix.")
    k = len(A[0])
    if any(len(row) != k for row in A):
        raise ValueError("Ragged matrix A.")
    if len(B) == 0:
        raise ValueError("Empty matrix B.")
    m = len(B[0])
    if any(len(row) != m for row in B):
        raise ValueError("Ragged matrix B.")
    if len(B) != k:
        raise ValueError("Incompatible shapes for matmul.")

    out: Matrix = [[0j for _ in range(m)] for _ in range(n)]
    for i in range(n):
        for j in range(m):
            s = 0j
            for t in range(k):
                s += A[i][t] * B[t][j]
            out[i][j] = s
    return out


def trace(A: Matrix) -> complex:
    n = len(A)
    if n == 0:
        raise ValueError("Empty matrix.")
```

```
    if any(len(row) != len(A[0]) for row in A):
        raise ValueError("Ragged matrix.")
    m = len(A[0])
    if n != m:
        raise ValueError("Trace requires a square matrix.")
    return sum(A[i][i] for i in range(n))


def wigner_smith_Q(Sm: Scattering, Sp: Scattering, dE: float) -> Scattering:
    """
    Central finite-difference approximation of Q(E) using S(E-dE) and S(E+dE).
    """
    if dE == 0:
        raise ValueError("dE must be nonzero.")

    if not _is_matrix(Sm) and not _is_matrix(Sp):
        dS = (Sp - Sm) / (2.0 * dE)
        S_mid = 0.5 * (Sp + Sm)
        return -1j * (S_mid.conjugate() * dS)

    if not _is_matrix(Sm) or not _is_matrix(Sp):
        raise ValueError("Sm and Sp must have the same representation (both scalar or
        ↪  both matrix).")

    dS: Matrix = [[(Sp[i][j] - Sm[i][j]) / (2.0 * dE) for j in range(len(Sp[0]))] for
    ↪  i in range(len(Sp))]
    S_mid: Matrix = [[0.5 * (Sp[i][j] + Sm[i][j]) for j in range(len(Sp[0]))] for i in
    ↪  range(len(Sp))]
    return [[(-1j) * z for z in row] for row in matmul(conj_transpose(S_mid), dS)]


def tau_ws_from_S(Sm: Scattering, Sp: Scattering, dE: float) -> float:
    Q = wigner_smith_Q(Sm, Sp, dE)
    if _is_matrix(Q):
        return float(trace(Q).real)
    return float(Q.real)


def toy_S_breit_wigner(E: float, *, E0: float = 1.0, gamma: float = 0.2) -> complex:
    """
    A 1-channel unitary Breit--Wigner resonance model:
      S(E) = (E - E0 - i gamma/2) / (E - E0 + i gamma/2).
    """
    z = (E - E0) + 0.5j * gamma
    return ((E - E0) - 0.5j * gamma) / z


def linspace(a: float, b: float, n: int) -> list[float]:
    if n < 2:
        raise ValueError("n must be >= 2.")
    step = (b - a) / float(n - 1)
    return [a + i * step for i in range(n)]


def sample_tau_ws(S: Callable[[float], Scattering], energies: list[float]) ->
↪  list[float | None]:
    """
    Compute tau_WS(E) for interior points using central differences.
```

```
    Endpoints are returned as None.
    """
    if len(energies) < 3:
        raise ValueError("Need at least 3 energy points.")

    S_list = [S(float(E)) for E in energies]
    taus: list[float | None] = [None for _ in energies]

    # Scalar 1-channel case: compute tau_WS as a phase derivative with unwrapping to
    # avoid branch-cut artifacts at resonant phase jumps.
    if not _is_matrix(S_list[0]):
        phases = [cmath.phase(z) for z in S_list]  # principal values in (-pi, pi]
        unwrapped: list[float] = [phases[0]]
        for i in range(1, len(phases)):
            p = phases[i]
            prev = unwrapped[-1]
            delta = p - prev
            while delta > math.pi:
                p -= 2.0 * math.pi
                delta = p - prev
            while delta < -math.pi:
                p += 2.0 * math.pi
                delta = p - prev
            unwrapped.append(p)

        for i in range(1, len(energies) - 1):
            dE = energies[i + 1] - energies[i - 1]
            taus[i] = (unwrapped[i + 1] - unwrapped[i - 1]) / dE
        return taus

    for i in range(1, len(energies) - 1):
        dE = energies[i + 1] - energies[i - 1]
        taus[i] = tau_ws_from_S(S_list[i - 1], S_list[i + 1], dE)
    return taus


def moving_average(values: list[float], window: int) -> list[float]:
    if window <= 1:
        return list(values)
    if window % 2 == 0:
        raise ValueError("Smoothing window must be odd.")
    n = len(values)
    half = window // 2
    out: list[float] = []
    for i in range(n):
        a = max(0, i - half)
        b = min(n, i + half + 1)
        out.append(sum(values[a:b]) / float(b - a))
    return out


def sample_tau_ws_from_scalar_samples(
    energies: list[float],
    S_list: list[complex],
    *,
    smooth_window: int = 1,
) -> list[float | None]:
    """
```

```
    Compute tau_WS(E) from sampled 1-channel complex S(E) values.
    Uses phase unwrapping and central differences; optional smoothing is applied
    to the unwrapped phase before differentiating.
    """
    if len(energies) != len(S_list):
        raise ValueError("energies and S_list must have the same length.")
    if len(energies) < 3:
        raise ValueError("Need at least 3 sample points.")

    phases = [cmath.phase(z) for z in S_list]  # principal values in (-pi, pi]
    unwrapped: list[float] = [phases[0]]
    for i in range(1, len(phases)):
        p = phases[i]
        prev = unwrapped[-1]
        delta = p - prev
        while delta > math.pi:
            p -= 2.0 * math.pi
            delta = p - prev
        while delta < -math.pi:
            p += 2.0 * math.pi
            delta = p - prev
        unwrapped.append(p)

    unwrapped = moving_average(unwrapped, window=smooth_window)

    taus: list[float | None] = [None for _ in energies]
    for i in range(1, len(energies) - 1):
        dE = energies[i + 1] - energies[i - 1]
        if dE == 0.0:
            raise ValueError("Repeated energy value encountered.")
        taus[i] = (unwrapped[i + 1] - unwrapped[i - 1]) / dE
    return taus


def load_scalar_S_data(path: str) -> tuple[list[float], list[complex]]:
    """
    Load sampled 1-channel scattering data from a text file.

    Supported formats (whitespace or comma separated, comments start with '#'):
      - E  phase_radians
      - E  Re(S)  Im(S)
    """
    energies: list[float] = []
    values: list[complex] = []
    with open(path, "r", encoding="utf-8") as f:
        for line in f:
            line = line.strip()
            if not line or line.startswith("#"):
                continue
            parts = line.replace(",", " ").split()
            if len(parts) == 2:
                E = float(parts[0])
                phase = float(parts[1])
                S = cmath.exp(1j * phase)
            elif len(parts) == 3:
                E = float(parts[0])
                re = float(parts[1])
                im = float(parts[2])
```

```
                S = complex(re, im)
            else:
                raise ValueError("Expected 2 or 3 columns per row.")
            energies.append(E)
            values.append(S)

    if len(energies) < 3:
        raise ValueError("Need at least 3 sample points.")

    # Sort by energy.
    pairs = sorted(zip(energies, values), key=lambda t: t[0])
    energies_sorted = [p[0] for p in pairs]
    values_sorted = [p[1] for p in pairs]
    return energies_sorted, values_sorted


def write_latex_rows(energies: list[float], taus: list[float | None], tau0: float,
↪  out_path: Path) -> None:
    kappas: list[float | None] = [None if t is None else (t / tau0) for t in taus]
    rows: list[str] = []
    for E, tau, kappa in zip(energies, taus, kappas):
        if tau is None or kappa is None:
            continue
        rows.append(f"{E:.6g} & {tau:.6g} & {kappa:.6g} \\\\")
    out_path.write_text("\n".join(rows) + "\n", encoding="utf-8")


def main() -> None:
    ap = argparse.ArgumentParser(description="Compute Wigner-Smith time delay and
    ↪  kappa_WS(E).")
    ap.add_argument("--input", type=str, default="", help="Optional input data file
    ↪  for 1-channel S(E).")
    ap.add_argument("--tau0", type=float, default=1.0, help="Reference tick duration
    ↪  tau0 for kappa_WS.")
    ap.add_argument("--smooth", type=int, default=1, help="Odd moving-average window
    ↪  for phase smoothing (1=off).")
    ap.add_argument("--E0", type=float, default=1.0, help="Toy model resonance center
    ↪  E0.")
    ap.add_argument("--gamma", type=float, default=0.2, help="Toy model linewidth
    ↪  gamma.")
    ap.add_argument("--Emin", type=float, default=0.0, help="Toy model: minimum
    ↪  energy.")
    ap.add_argument("--Emax", type=float, default=2.0, help="Toy model: maximum
    ↪  energy.")
    ap.add_argument("--n", type=int, default=17, help="Toy model: number of energy
    ↪  points.")
    ap.add_argument(
        "--output",
        type=str,
        default="",
        help="Optional output path for LaTeX rows (default:
        ↪  sections/generated/wigner_smith_rows.tex).",
    )
    args = ap.parse_args()

    tau0 = float(args.tau0)
    if tau0 <= 0.0:
        raise ValueError("tau0 must be positive.")
```

```
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)

    out_path = Path(args.output) if args.output else (out_dir /
    ↪  "wigner_smith_rows.tex")

    if args.input:
        energies, S_list = load_scalar_S_data(args.input)
        # Basic unitarity diagnostic for the 1-channel case.
        mags = [abs(z) for z in S_list]
        mean_mag = sum(mags) / float(len(mags))
        max_dev = max(abs(m - 1.0) for m in mags)
        print(f"Loaded {len(energies)} samples. mean|S|={mean_mag:.6f},
        ↪  max||S|-1|={max_dev:.6f}")

        taus = sample_tau_ws_from_scalar_samples(energies, S_list,
        ↪  smooth_window=int(args.smooth))
    else:
        energies = linspace(float(args.Emin), float(args.Emax), int(args.n))

        def S(E: float) -> complex:
            return toy_S_breit_wigner(E, E0=float(args.E0), gamma=float(args.gamma))

        taus = sample_tau_ws(S, energies)

    write_latex_rows(energies, taus, tau0=tau0, out_path=out_path)

    print("E\ttau_WS(E)\tkappa_WS(E)")
    for E, tau in zip(energies, taus):
        if tau is None:
            continue
        kappa = tau / tau0
        print(f"{E:.6g}\t{tau:.6g}\t{kappa:.6g}")
    print(f"Wrote {out_path}")


if __name__ == "__main__":
    main()
```

H.6 Experiment A′: address-family sensitivity (Hilbert vs. Morton vs. shuffled)

```
# -*- coding: utf-8 -*-
"""
Address-family sensitivity on a 2D screen lattice.

For a finite-resolution address map A_n : {0..2^(2n)-1} -> {0..2^n-1}^2,
define the neighbor separation on screen edges:
  Delta_A(x,y) = |A_n^{-1}(x) - A_n^{-1}(y)|
and the local scan-chain overhead proxy:
  kappa_tilde(x) = max_{y~x} Delta_A(x,y).

This script compares Hilbert vs Morton (Z-order) plus a shuffled baseline for
orders n=1..8 (2D), and writes LaTeX table rows into sections/generated/.
```

```
It also reports:
  (i) sensitivity to the neighborhood model (Manhattan vs Chebyshev neighbors),
  (ii) a simple finite-size trend fit for the growth of high quantiles.

As an additional robustness check, it also computes the same proxy for a 3D
screen using Morton vs shuffled baselines (orders n=1..5).
"""

from __future__ import annotations

from dataclasses import dataclass
import math
from pathlib import Path
import random
from typing import Callable


def _rot(s: int, x: int, y: int, rx: int, ry: int) -> tuple[int, int]:
    """Rotate/flip a quadrant (standard Hilbert helper)."""
    if ry == 0:
        if rx == 1:
            x = s - 1 - x
            y = s - 1 - y
        x, y = y, x
    return x, y


def hilbert_d2xy(order: int, d: int) -> tuple[int, int]:
    """
    2D Hilbert mapping: d in [0, 2^(2*order)-1] -> (x,y) in [0, 2^order-1]^2.
    """
    n = 1 << order
    x = 0
    y = 0
    t = int(d)
    s = 1
    while s < n:
        rx = 1 & (t // 2)
        ry = 1 & (t ^ rx)
        x, y = _rot(s, x, y, rx, ry)
        x += s * rx
        y += s * ry
        t //= 4
        s *= 2
    return x, y


def morton_d2xy(order: int, d: int) -> tuple[int, int]:
    """
    2D Morton / Z-order mapping via bit de-interleaving.
    """
    x = 0
    y = 0
    for i in range(order):
        x |= ((d >> (2 * i)) & 1) << i
        y |= ((d >> (2 * i + 1)) & 1) << i
    return x, y
```

```
def morton_d2xyz(order: int, d: int) -> tuple[int, int, int]:
    """
    3D Morton / Z-order mapping via bit de-interleaving.
    """
    x = 0
    y = 0
    z = 0
    for i in range(order):
        x |= ((d >> (3 * i)) & 1) << i
        y |= ((d >> (3 * i + 1)) & 1) << i
        z |= ((d >> (3 * i + 2)) & 1) << i
    return x, y, z


def build_index_map(order: int, d2xy) -> list[list[int]]:
    m = 1 << order
    total = m * m
    idx = [[0] * m for _ in range(m)]
    for d in range(total):
        x, y = d2xy(order, d)
        idx[x][y] = d
    return idx


def build_index_map_shuffled(order: int, seed: int) -> list[list[int]]:
    """
    A deterministic shuffled baseline: assign indices to uniformly shuffled
    ↪  coordinates.
    """
    m = 1 << order
    total = m * m
    coords = [(x, y) for x in range(m) for y in range(m)]
    rng = random.Random(seed)
    rng.shuffle(coords)
    idx = [[0] * m for _ in range(m)]
    for d in range(total):
        x, y = coords[d]
        idx[x][y] = d
    return idx


def build_index_map_3d(order: int, d2xyz) -> list[int]:
    m = 1 << order
    total = m * m * m
    idx = [0] * total
    for d in range(total):
        x, y, z = d2xyz(order, d)
        idx[(x * m + y) * m + z] = d
    return idx


def build_index_map_shuffled_3d(order: int, seed: int) -> list[int]:
    m = 1 << order
    total = m * m * m
    coords = [(x, y, z) for x in range(m) for y in range(m) for z in range(m)]
    rng = random.Random(seed)
```

```
    rng.shuffle(coords)
    idx = [0] * total
    for d in range(total):
        x, y, z = coords[d]
        idx[(x * m + y) * m + z] = d
    return idx


def percentile(sorted_values: list[int], p: float) -> int:
    if not sorted_values:
        raise ValueError("Empty sample.")
    if p <= 0.0:
        return sorted_values[0]
    if p >= 1.0:
        return sorted_values[-1]
    i = int(p * (len(sorted_values) - 1))
    return sorted_values[i]


@dataclass(frozen=True)
class Stats:
    mean: float
    p50: int
    p90: int
    p99: int
    mx: int


def stats_int(values: list[int]) -> Stats:
    if not values:
        raise ValueError("Empty sample.")
    values_sorted = sorted(values)
    n = len(values_sorted)
    mean = sum(values_sorted) / float(n)
    return Stats(
        mean=mean,
        p50=percentile(values_sorted, 0.50),
        p90=percentile(values_sorted, 0.90),
        p99=percentile(values_sorted, 0.99),
        mx=values_sorted[-1],
    )


def neighbor_separations(idx: list[list[int]]) -> list[int]:
    m = len(idx)
    deltas: list[int] = []
    for x in range(m):
        for y in range(m):
            if x + 1 < m:
                deltas.append(abs(idx[x][y] - idx[x + 1][y]))
            if y + 1 < m:
                deltas.append(abs(idx[x][y] - idx[x][y + 1]))
    return deltas


@dataclass(frozen=True)
class NeighborModel:
    name: str
```

```
    offsets: tuple[tuple[int, int], ...]


MANHATTAN_1 = NeighborModel(name="Manhattan", offsets=((1, 0), (-1, 0), (0, 1), (0,
↪  -1)))
CHEBYSHEV_1 = NeighborModel(
    name="Chebyshev",
    offsets=(
        (-1, -1),
        (-1, 0),
        (-1, 1),
        (0, -1),
        (0, 1),
        (1, -1),
        (1, 0),
        (1, 1),
    ),
)


def local_kappa_proxy(idx: list[list[int]], neighbors: NeighborModel) -> list[int]:
    m = len(idx)
    prox: list[int] = []
    for x in range(m):
        for y in range(m):
            best = 0
            for dx, dy in neighbors.offsets:
                xx = x + dx
                yy = y + dy
                if 0 <= xx < m and 0 <= yy < m:
                    best = max(best, abs(idx[x][y] - idx[xx][yy]))
            prox.append(best)
    return prox


def local_kappa_proxy_3d(idx: list[int], m: int) -> list[int]:
    """
    3D Manhattan (6-neighbor) local proxy field on an m x m x m grid.
    """
    prox: list[int] = []
    for x in range(m):
        for y in range(m):
            for z in range(m):
                p = (x * m + y) * m + z
                base = idx[p]
                best = 0
                if x > 0:
                    best = max(best, abs(base - idx[((x - 1) * m + y) * m + z]))
                if x + 1 < m:
                    best = max(best, abs(base - idx[((x + 1) * m + y) * m + z]))
                if y > 0:
                    best = max(best, abs(base - idx[(x * m + (y - 1)) * m + z]))
                if y + 1 < m:
                    best = max(best, abs(base - idx[(x * m + (y + 1)) * m + z]))
                if z > 0:
                    best = max(best, abs(base - idx[(x * m + y) * m + (z - 1)]))
                if z + 1 < m:
                    best = max(best, abs(base - idx[(x * m + y) * m + (z + 1)]))
```

```
                prox.append(best)
    return prox


def linear_fit(xs: list[float], ys: list[float]) -> tuple[float, float, float]:
    """
    Ordinary least squares fit for y = a + b x. Returns (b, a, R^2).
    """
    if len(xs) != len(ys) or len(xs) < 2:
        raise ValueError("Need at least 2 points for a fit.")
    n = float(len(xs))
    x_bar = sum(xs) / n
    y_bar = sum(ys) / n
    sxx = sum((x - x_bar) ** 2 for x in xs)
    if sxx == 0.0:
        raise ValueError("Degenerate x values.")
    sxy = sum((x - x_bar) * (y - y_bar) for x, y in zip(xs, ys))
    b = sxy / sxx
    a = y_bar - b * x_bar
    ss_tot = sum((y - y_bar) ** 2 for y in ys)
    ss_res = sum((y - (a + b * x)) ** 2 for x, y in zip(xs, ys))
    r2 = 1.0 - (ss_res / ss_tot) if ss_tot > 0.0 else 1.0
    return b, a, r2


@dataclass(frozen=True)
class AddressMap:
    name: str
    build_idx: Callable[[int], list[list[int]]]


def write_rows(rows: list[str], filename: str) -> None:
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)
    (out_dir / filename).write_text("\n".join(rows) + "\n", encoding="utf-8")


def main() -> None:
    maps = [
        AddressMap(name="Hilbert", build_idx=lambda order: build_index_map(order,
        ↪  hilbert_d2xy)),
        AddressMap(name="Z-order", build_idx=lambda order: build_index_map(order,
        ↪  morton_d2xy)),
        AddressMap(name="Shuffled", build_idx=lambda order:
        ↪  build_index_map_shuffled(order, seed=123456 + order)),
    ]

    edge_rows: list[str] = []
    proxy_rows: list[str] = []
    fit_rows: list[str] = []
    neighbor_model_rows: list[str] = []

    # Accumulate per-map proxy stats across orders for a simple scaling fit.
    by_map: dict[str, list[tuple[int, Stats]]] = {m.name: [] for m in maps}

    for order in range(1, 9):
        for amap in maps:
```

```
        idx = amap.build_idx(order)

        deltas = neighbor_separations(idx)
        s_edge = stats_int(deltas)
        edge_rows.append(
            f"{order} & {amap.name} & {s_edge.mean:.2f} & {s_edge.p50} &
            ↪  {s_edge.p90} & {s_edge.p99} & {s_edge.mx} \\\\"
        )

        prox = local_kappa_proxy(idx, MANHATTAN_1)
        s_prox = stats_int(prox)
        by_map[amap.name].append((order, s_prox))
        proxy_rows.append(
            f"{order} & {amap.name} & {s_prox.mean:.2f} & {s_prox.p50} &
            ↪  {s_prox.p90} & {s_prox.p99} & {s_prox.mx} \\\\"
        )

    print(f"order={order}: ok")

# Neighborhood-model sensitivity at a fixed resolution.
fixed_order = 8
for amap in maps:
    idx = amap.build_idx(fixed_order)
    s_m = stats_int(local_kappa_proxy(idx, MANHATTAN_1))
    s_c = stats_int(local_kappa_proxy(idx, CHEBYSHEV_1))
    ratio_p99 = (float(s_c.p99) / float(s_m.p99)) if s_m.p99 > 0 else float("inf")
    ratio_max = (float(s_c.mx) / float(s_m.mx)) if s_m.mx > 0 else float("inf")
    neighbor_model_rows.append(
        f"{amap.name} & {s_m.p99} & {s_c.p99} & {ratio_p99:.3f} & {s_m.mx} &
        ↪  {s_c.mx} & {ratio_max:.3f} \\\\"
    )

# Simple finite-size trend fits for high quantiles of the Manhattan proxy.
for amap in maps:
    data = by_map[amap.name]
    xs = [float(order) for order, _ in data]
    ys_p99 = [math.log2(float(s.p99)) for _, s in data]
    ys_max = [math.log2(float(s.mx)) for _, s in data]
    b_p99, _a_p99, r2_p99 = linear_fit(xs, ys_p99)
    b_max, _a_max, r2_max = linear_fit(xs, ys_max)
    fit_rows.append(f"{amap.name} & {b_p99:.3f} & {r2_p99:.3f} & {b_max:.3f} &
    ↪  {r2_max:.3f} \\\\")

write_rows(edge_rows, "address_neighbor_separation_rows.tex")
write_rows(proxy_rows, "address_kappa_proxy_rows.tex")
write_rows(neighbor_model_rows, "address_neighbor_model_sensitivity_rows.tex")
write_rows(fit_rows, "address_kappa_proxy_fit_rows.tex")
# 3D robustness rows.
proxy_3d_rows: list[str] = []
for order in range(1, 6):
    m = 1 << order
    for name, idx3 in [
        ("Z-order", build_index_map_3d(order, morton_d2xyz)),
        ("Shuffled", build_index_map_shuffled_3d(order, seed=654321 + order)),
    ]:
        prox3 = local_kappa_proxy_3d(idx3, m=m)
        s3 = stats_int(prox3)
        proxy_3d_rows.append(
```

```
                f"{order} & {name} & {s3.mean:.2f} & {s3.p50} & {s3.p90} & {s3.p99} &
                ↪  {s3.mx} \\\\"
            )
        print(f"order={order} (3D): ok")
    write_rows(proxy_3d_rows, "address_kappa_proxy_3d_rows.tex")
    print("Wrote sections/generated/address_neighbor_separation_rows.tex")
    print("Wrote sections/generated/address_kappa_proxy_rows.tex")
    print("Wrote sections/generated/address_neighbor_model_sensitivity_rows.tex")
    print("Wrote sections/generated/address_kappa_proxy_fit_rows.tex")
    print("Wrote sections/generated/address_kappa_proxy_3d_rows.tex")


if __name__ == "__main__":
    main()
```

H.7 Experiment F: scan-chain redshift toy from a computed $\kappa(x)$ landscape

```
# -*- coding: utf-8 -*-
"""
Toy redshift from a computed overhead landscape on a scan chain.

We build a position-dependent overhead proxy kappa(x) on the 2D screen lattice
using a fixed address map A_n and a scan-chain placement pi(x)=A_n^{-1}(x):

  kappa(x) := max_{y~x} |A_n^{-1}(x) - A_n^{-1}(y)|

Interpreting one local "clock cycle" at x as taking kappa(x) global ticks,
the relational time scaling predicts:
  d tau_loc = (kappa0 / kappa(x)) dt
and the redshift ratio between x1 and x2 is kappa(x2)/kappa(x1).

This script demonstrates the ratio numerically by counting completed cycles
over a long horizon t_max and writes LaTeX rows into sections/generated/.
"""

from __future__ import annotations

from dataclasses import dataclass
from pathlib import Path


def _rot(s: int, x: int, y: int, rx: int, ry: int) -> tuple[int, int]:
    if ry == 0:
        if rx == 1:
            x = s - 1 - x
            y = s - 1 - y
        x, y = y, x
    return x, y


def hilbert_d2xy(order: int, d: int) -> tuple[int, int]:
    n = 1 << order
    x = 0
    y = 0
    t = int(d)
    s = 1
```

```
    while s < n:
        rx = 1 & (t // 2)
        ry = 1 & (t ^ rx)
        x, y = _rot(s, x, y, rx, ry)
        x += s * rx
        y += s * ry
        t //= 4
        s *= 2
    return x, y


def morton_d2xy(order: int, d: int) -> tuple[int, int]:
    x = 0
    y = 0
    for i in range(order):
        x |= ((d >> (2 * i)) & 1) << i
        y |= ((d >> (2 * i + 1)) & 1) << i
    return x, y


def build_index_map(order: int, d2xy) -> list[list[int]]:
    m = 1 << order
    total = m * m
    idx = [[0] * m for _ in range(m)]
    for d in range(total):
        x, y = d2xy(order, d)
        idx[x][y] = d
    return idx


def kappa_proxy_field(idx: list[list[int]]) -> tuple[list[int], tuple[int, int], int,
↪  tuple[int, int], int]:
    m = len(idx)
    vals: list[int] = []
    min_xy = (0, 0)
    max_xy = (0, 0)
    min_k = 10**18
    max_k = -1

    for x in range(m):
        for y in range(m):
            best = 0
            if x > 0:
                best = max(best, abs(idx[x][y] - idx[x - 1][y]))
            if x + 1 < m:
                best = max(best, abs(idx[x][y] - idx[x + 1][y]))
            if y > 0:
                best = max(best, abs(idx[x][y] - idx[x][y - 1]))
            if y + 1 < m:
                best = max(best, abs(idx[x][y] - idx[x][y + 1]))

            vals.append(best)
            if best < min_k:
                min_k = best
                min_xy = (x, y)
            if best > max_k:
                max_k = best
                max_xy = (x, y)
```

```
    return vals, min_xy, int(min_k), max_xy, int(max_k)


def find_any_coordinate_with_kappa(idx: list[list[int]], target_k: int) -> tuple[int,
↪  int] | None:
    m = len(idx)
    for x in range(m):
        for y in range(m):
            best = 0
            if x > 0:
                best = max(best, abs(idx[x][y] - idx[x - 1][y]))
            if x + 1 < m:
                best = max(best, abs(idx[x][y] - idx[x + 1][y]))
            if y > 0:
                best = max(best, abs(idx[x][y] - idx[x][y - 1]))
            if y + 1 < m:
                best = max(best, abs(idx[x][y] - idx[x][y + 1]))
            if best == target_k:
                return (x, y)
    return None


def percentile(sorted_values: list[int], p: float) -> int:
    if not sorted_values:
        raise ValueError("Empty sample.")
    if p <= 0.0:
        return sorted_values[0]
    if p >= 1.0:
        return sorted_values[-1]
    i = int(p * (len(sorted_values) - 1))
    return sorted_values[i]


@dataclass(frozen=True)
class PairRow:
    x1: int
    y1: int
    k1: int
    x2: int
    y2: int
    k2: int
    pred: float
    meas: float


def cycles(t_max: int, kappa: int) -> int:
    if kappa <= 0:
        raise ValueError("kappa must be positive.")
    return t_max // kappa


def write_rows(rows: list[str], filename: str) -> None:
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)
    (out_dir / filename).write_text("\n".join(rows) + "\n", encoding="utf-8")
```

```
def main() -> None:
    order = 8
    t_max = 200_000_000

    maps = [
        ("Hilbert", hilbert_d2xy),
        ("Z-order", morton_d2xy),
    ]

    rows: list[str] = []

    for name, d2xy in maps:
        idx = build_index_map(order, d2xy)
        vals, min_xy, min_k, max_xy, max_k = kappa_proxy_field(idx)

        vals_sorted = sorted(vals)
        med_k = percentile(vals_sorted, 0.50)
        med_xy = find_any_coordinate_with_kappa(idx, med_k)
        if med_xy is None:
            med_xy = (0, 0)

        points = [
            ("min", min_xy, min_k),
            ("median", med_xy, med_k),
            ("max", max_xy, max_k),
        ]

        pairs: list[PairRow] = []
        for (tag1, (x1, y1), k1), (tag2, (x2, y2), k2) in [
            (points[0], points[1]),
            (points[0], points[2]),
            (points[1], points[2]),
        ]:
            pred = float(k2) / float(k1)
            c1 = cycles(t_max, k1)
            c2 = cycles(t_max, k2)
            meas = float(c1) / float(c2) if c2 != 0 else float("inf")
            pairs.append(PairRow(x1=x1, y1=y1, k1=k1, x2=x2, y2=y2, k2=k2, pred=pred,
            ↪  meas=meas))

        for p in pairs:
            rows.append(
                f"{order} & {name} & ({p.x1},{p.y1}) & {p.k1} & ({p.x2},{p.y2}) &
                ↪  {p.k2} & {p.pred:.6f} & {p.meas:.6f} \\\\"
            )

        print(f"{name}: min_k={min_k} at {min_xy}, median_k={med_k} at {med_xy},
        ↪  max_k={max_k} at {max_xy}")

    write_rows(rows, "kappa_redshift_toy_rows.tex")
    print("Wrote sections/generated/kappa_redshift_toy_rows.tex")


if __name__ == "__main__":
    main()
```

H.8 Appendix worked example: 1+1D weighted scan-chain lapse matching

```
# -*- coding: utf-8 -*-
"""
1+1D worked example: weighted scan-chain compilation and weak-field lapse matching.

We specify:
  - a 1D screen lattice Sigma_n = {0..L-1}, L = 2^n,
  - a weighted path hardware graph with edge weights w_{i+1/2} (tick costs),
  - a local clock task at site i that executes a fixed two-body primitive on
    edges (i-1,i) and (i,i+1) sequentially (shared vertex constraint).

Then the exact compilation depth for interior sites is:
  kappa(i) = w_{i-1/2} + w_{i+1/2}
and the induced lapse is:
  N(i) = kappa0 / kappa(i).

We choose the edge weights by midpoint sampling so that N(i) matches a standard
weak-field target along a radial line: the Schwarzschild lapse
  N_Schw(r) = sqrt(1 - 2M/r)  (units c=G=1),
and we report finite-size errors and a log-log scaling fit.

This script writes LaTeX table rows into:
  sections/generated/1p1d_error_rows.tex
  sections/generated/1p1d_scaling_fit_rows.tex
  sections/generated/1p1d_curve_rows.tex
"""

from __future__ import annotations

from dataclasses import dataclass
import math
from pathlib import Path


def schw_lapse(r: float, *, M: float) -> float:
    if r <= 2.0 * M:
        raise ValueError("Need r > 2M for a real Schwarzschild lapse.")
    return math.sqrt(1.0 - (2.0 * M) / r)


@dataclass(frozen=True)
class Metrics:
    n: int
    L: int
    h: float
    max_abs: float
    rms: float
    rel_rms: float


def _round_half_up(x: float) -> int:
    return int(math.floor(x + 0.5))


def build_profile(
    *,
    n: int,
    r_min: float,
```

```
    r_max: float,
    M: float,
    kappa0: int,
) -> tuple[list[float], list[float], list[int], list[int | None], list[float | None]]:
    """
    Returns:
      r_sites: r_i at sites
      N_target: N_Schw(r_i)
      w_edges: w_{i+1/2} for i=0..L-2 (edge weights, integers)
      kappa_sites: kappa(i) for i=0..L-1 (None at endpoints)
      N_meas: induced N(i)=kappa0/kappa(i) (None at endpoints)
    """
    L = 1 << n
    if L < 4:
        raise ValueError("Need L >= 4 to have interior sites.")
    if r_max <= r_min:
        raise ValueError("Need r_max > r_min.")

    h = (r_max - r_min) / float(L - 1)
    r_sites = [r_min + i * h for i in range(L)]
    N_target = [schw_lapse(r, M=M) for r in r_sites]

    # Midpoint construction of edge weights:
    #   w_{i+1/2} ~ (1/2) * kappa_target(r_{i+1/2}), rounded to integer ticks.
    w_edges: list[int] = []
    for i in range(L - 1):
        r_mid = r_sites[i] + 0.5 * h
        N_mid = schw_lapse(r_mid, M=M)
        kappa_mid = float(kappa0) / N_mid
        w_edges.append(_round_half_up(0.5 * kappa_mid))

    # Exact compilation depth for the 2-edge local task on interior sites.
    kappa_sites: list[int | None] = [None for _ in range(L)]
    N_meas: list[float | None] = [None for _ in range(L)]
    for i in range(1, L - 1):
        k = w_edges[i - 1] + w_edges[i]
        kappa_sites[i] = k
        N_meas[i] = float(kappa0) / float(k)

    return r_sites, N_target, w_edges, kappa_sites, N_meas


def metrics_for_n(
    *,
    n: int,
    r_min: float,
    r_max: float,
    M: float,
    kappa0: int,
) -> Metrics:
    r_sites, N_target, _w_edges, _kappa_sites, N_meas = build_profile(
        n=n, r_min=r_min, r_max=r_max, M=M, kappa0=kappa0
    )
    L = len(r_sites)
    h = (r_max - r_min) / float(L - 1)

    errs: list[float] = []
    rel_errs: list[float] = []
```

```
    for i in range(1, L - 1):
        if N_meas[i] is None:
            continue
        e = abs(N_meas[i] - N_target[i])
        errs.append(e)
        rel_errs.append(e / N_target[i])

    if not errs:
        raise ValueError("No interior errors computed.")

    max_abs = max(errs)
    rms = math.sqrt(sum(e * e for e in errs) / float(len(errs)))
    rel_rms = math.sqrt(sum(e * e for e in rel_errs) / float(len(rel_errs)))
    return Metrics(n=n, L=L, h=h, max_abs=max_abs, rms=rms, rel_rms=rel_rms)


def linear_fit(xs: list[float], ys: list[float]) -> tuple[float, float, float]:
    """
    Ordinary least squares fit for y = a + b x.
    Returns (b, a, R^2).
    """
    if len(xs) != len(ys) or len(xs) < 2:
        raise ValueError("Need at least 2 points for a fit.")
    n = float(len(xs))
    x_bar = sum(xs) / n
    y_bar = sum(ys) / n
    sxx = sum((x - x_bar) ** 2 for x in xs)
    if sxx == 0.0:
        raise ValueError("Degenerate x values.")
    sxy = sum((x - x_bar) * (y - y_bar) for x, y in zip(xs, ys))
    b = sxy / sxx
    a = y_bar - b * x_bar
    ss_tot = sum((y - y_bar) ** 2 for y in ys)
    ss_res = sum((y - (a + b * x)) ** 2 for x, y in zip(xs, ys))
    r2 = 1.0 - (ss_res / ss_tot if ss_tot != 0.0 else 0.0)
    return b, a, r2


def write_rows(lines: list[str], filename: str) -> None:
    root = Path(__file__).resolve().parent.parent
    out_dir = root / "sections" / "generated"
    out_dir.mkdir(parents=True, exist_ok=True)
    (out_dir / filename).write_text("\n".join(lines) + "\n", encoding="utf-8")


def main() -> None:
    # Weak-field target parameters (units c=G=1).
    M = 0.05
    r_min = 1.0
    r_max = 8.0

    # Tick calibration: larger kappa0 reduces integer-quantization effects.
    kappa0 = 1_000_000_000

    # Finite-size scan (orders).
    n_min = 6
    n_max = 11
```

```
    # Representative curve sample order.
    n_curve = 9
    curve_points = 12

    metrics: list[Metrics] = []
    for n in range(n_min, n_max + 1):
        m = metrics_for_n(n=n, r_min=r_min, r_max=r_max, M=M, kappa0=kappa0)
        metrics.append(m)
        print(
            f"n={m.n:2d}  L={m.L:4d}  h={m.h:.6g}  max|dN|={m.max_abs:.3e}
            ↪  rms|dN|={m.rms:.3e}  rel_rms={m.rel_rms:.3e}"
        )

    # LaTeX rows: errors per n.
    error_rows: list[str] = []
    for m in metrics:
        error_rows.append(
            f"{m.n} & {m.L} & {m.h:.6g} & {m.max_abs:.3e} & {m.rms:.3e} &
            ↪  {m.rel_rms:.3e} \\\\"
        )
    write_rows(error_rows, "1p1d_error_rows.tex")

    # Scaling fit: log(rms) vs log(h) (expect slope ~ 2 from midpoint averaging).
    xs = [math.log(mm.h) for mm in metrics]
    ys = [math.log(mm.rms) for mm in metrics]
    slope, _a, r2 = linear_fit(xs, ys)
    fit_rows = [f"{n_min}--{n_max} & {slope:.4f} & {r2:.5f} \\\\"]
    write_rows(fit_rows, "1p1d_scaling_fit_rows.tex")
    print(f"fit: log(rms) ~ a + b log(h),  b={slope:.4f}, R^2={r2:.5f}")

    # Representative curve samples.
    r_sites, N_target, _w_edges, _kappa_sites, N_meas = build_profile(
        n=n_curve, r_min=r_min, r_max=r_max, M=M, kappa0=kappa0
    )
    L = len(r_sites)
    idxs = [
        int(round(1 + (L - 3) * k / float(curve_points - 1))) for k in
        ↪  range(curve_points)
    ]
    idxs = [min(max(i, 1), L - 2) for i in idxs]
    idxs = sorted(set(idxs))

    curve_rows: list[str] = []
    for i in idxs:
        Nm = float(N_meas[i]) if N_meas[i] is not None else float("nan")
        Nt = float(N_target[i])
        dn = Nm - Nt
        curve_rows.append(f"{r_sites[i]:.6g} & {Nt:.10f} & {Nm:.10f} & {dn:.3e} \\\\")
    write_rows(curve_rows, "1p1d_curve_rows.tex")

    print("Wrote sections/generated/1p1d_error_rows.tex")
    print("Wrote sections/generated/1p1d_scaling_fit_rows.tex")
    print("Wrote sections/generated/1p1d_curve_rows.tex")


if __name__ == "__main__":
    main()
```

References

[1] Gerard 't Hooft. Dimensional reduction in quantum gravity. *arXiv preprint gr-qc/9310026*, 1993.

[2] Leonard Susskind. The world as a hologram. *Journal of Mathematical Physics*, 36:6377–6396, 1995.

[3] Raphael Bousso. The holographic principle. *Reviews of Modern Physics*, 74:825–874, 2002.

[4] Ahmed Almheiri, Xi Dong, and Daniel Harlow. Bulk locality and quantum error correction in AdS/CFT. *Journal of High Energy Physics*, 2015(4):163, 2015.

[5] Fernando Pastawski, Beni Yoshida, Daniel Harlow, and John Preskill. Holographic quantum error-correcting codes: Toy models for the bulk/boundary correspondence. *Journal of High Energy Physics*, 2015(6):1–55, 2015.

[6] Daniel Harlow. The ryu–takayanagi formula from quantum error correction. *Communications in Mathematical Physics*, 354(3):865–912, 2017.

[7] G. Evenbly and G. Vidal. Algorithms for entanglement renormalization, 2009.

[8] Brian Swingle. Entanglement renormalization and holography, 2009.

[9] Leonard Susskind. Addendum to computational complexity and black hole horizons, 2014.

[10] Adam R. Brown, Daniel A. Roberts, Leonard Susskind, Brian Swingle, and Ying Zhao. Complexity equals action, 2016.

[11] Alexandre Belin, Robert C. Myers, Shan-Ming Ruan, Gábor Sárosi, and Antony J. Speranza. Does complexity equal anything? *Physical Review Letters*, 128(8):081602, 2022.

[12] Alexandre Belin, Robert C. Myers, Shan-Ming Ruan, Gábor Sárosi, and Antony J. Speranza. Complexity equals anything ii. *Journal of High Energy Physics*, 01:154, 2023.

[13] Robert C. Myers and Shan-Ming Ruan. Complexity equals (almost) anything. arXiv:2403.17475, 2024.

[14] Eivind Jørstad, Robert C. Myers, and Shan-Ming Ruan. Complexity=anything: Singularity probes. arXiv:2304.05453, 2023.

[15] Michal P. Heller, Jacopo Papalini, and Tim Schuhmann. Krylov spread complexity as holographic complexity beyond jt gravity. arXiv:2412.17785, 2024.

[16] Amin Faraji Astaneh. Quantum complexity of $t\bar{T}$-deformation and its implications. arXiv:2408.06055, 2024.

[17] Seth Lloyd. Ultimate physical limits to computation. *Nature*, 406:1047–1054, 2000.

[18] Luca Bombelli, Joohan Lee, David Meyer, and Rafael D. Sorkin. Space-time as a causal set. *Physical Review Letters*, 59(5):521–524, 1987.

[19] Hans Maassen and Jos B. M. Uffink. Generalized entropic uncertainty relations. *Physical Review Letters*, 60(12):1103–1106, 1988.

[20] Mario Berta, Matthias Christandl, Roger Colbeck, Joseph M. Renes, and Renato Renner. The uncertainty principle in the presence of quantum memory, 2011.

[21] René Schwonnek, David Reeb, and Reinhard F. Werner. Measurement uncertainty for finite quantum observables, 2016.

[22] Moisés Bermejo Morán and Felix Huber. Uncertainty relations from state polynomial optimization, 2024.

[23] Alain Connes. Trace formula in noncommutative geometry and the zeros of the riemann zeta function. *Selecta Mathematica (New Series)*, 5:29–106, 1999.

[24] M. V. Berry and J. P. Keating. The Riemann zeros and eigenvalue asymptotics. *SIAM Review*, 41(2):236–266, 1999.

[25] Alain Connes, Caterina Consani, and Henri Moscovici. Zeta zeros and prolate wave operators, 2024.

[26] L. Kuipers and Harald Niederreiter. *Uniform Distribution of Sequences*. Wiley, New York, 1974.

[27] Michael Drmota and Robert F. Tichy. *Sequences, Discrepancies and Applications*, volume 1651 of *Lecture Notes in Mathematics*. Springer, Berlin, 1997.

[28] Karl Kraus. *States, Effects, and Operations: Fundamental Notions of Quantum Theory*, volume 190 of *Lecture Notes in Physics*. Springer, Berlin, 1983.

[29] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.

[30] Paul Busch. Quantum states and generalized observables: A simple proof of gleason's theorem. *Physical Review Letters*, 91(12):120403, 2003.

[31] David Hilbert. über die stetige abbildung einer linie auf ein flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[32] Hans Sagan. *Space-Filling Curves*. Springer, New York, 1994.

[33] Bongki Moon, H. V. Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):124–141, 2001.

[34] Serge Massar and Philippe Spindel. Uncertainty relation for the discrete fourier transform. *Physical Review Letters*, 100(19):190401, 2008.

[35] Haobo Ma. Computational teleology in hpa–ω. Companion architecture manuscript, 2025.

[36] Charles W. Misner, Kip S. Thorne, and John Archibald Wheeler. *Gravitation*. W. H. Freeman, San Francisco, 1973.

[37] Robert M. Wald. *General Relativity*. University of Chicago Press, Chicago, 1984.

[38] Sean M. Carroll. *Spacetime and Geometry: An Introduction to General Relativity*. Addison-Wesley, 2004.

[39] F. T. Leighton, Bruce M. Maggs, and Satish B. Rao. Packet routing and job-shop scheduling in O(Congestion + Dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

[40] Haobo Ma. Computational action principle: Least-discrepancy dynamics and field unification in hpa–ω. Companion variational unification manuscript, 2025.

[41] Haobo Ma. Omega theory: Axiomatic foundations of holographic spacetime and interactive evolution. Companion physics manuscript, 2025.

[42] Eric Poisson. *A Relativist's Toolkit: The Mathematics of Black-Hole Mechanics*. Cambridge University Press, Cambridge, 2004.

[43] Neil Ashby. Relativity in the global positioning system. *Living Reviews in Relativity*, 6(1):1, 2003.

[44] Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, Providence, RI, 2 edition, 2010.

[45] Haobo Ma. Holographic polar dynamics: Topological inversion of the schwarzschild singularity and the phase origin of gravity. Companion dynamics manuscript, 2025.

[46] Arnaud Denjoy. Sur les courbes définies par les équations différentielles à la surface du tore. *Journal de Mathématiques Pures et Appliquées*, 11:333–375, 1932.

[47] J. F. Koksma. Ein mengentheoretischer satz über die gleichverteilung modulo eins. *Compositio Mathematica*, 2:250–258, 1935.

[48] Jiří Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Springer, 1999.

[49] Josef Dick and Friedrich Pillichshammer. *Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration*. Cambridge University Press, 2010.

[50] Stefan Heinrich, Henryk Woźniakowski, Grzegorz W. Wasilkowski, and Erich Novak. The inverse of the star-discrepancy depends linearly on the dimension. *Acta Arithmetica*, 96(3):279–302, 2001.

[51] Christoph Aistleitner and Gerhard Larcher. Additive energy and irregularities of distribution. arXiv:1608.06847, 2016.

[52] G. H. Hardy. *Divergent Series*. Oxford University Press, Oxford, 1949.

[53] Ricardo Estrada and Ram P. Kanwal. *A Distributional Approach to Asymptotics: Theory and Applications*. Birkhäuser, Boston, 2 edition, 2002.

[54] Stephen W. Hawking. Zeta function regularization of path integrals in curved spacetime. *Communications in Mathematical Physics*, 55:133–148, 1977.

[55] Emilio Elizalde. *Ten Physical Applications of Spectral Zeta Functions*. Springer, 1995.

[56] Klaus Kirsten. *Spectral Functions in Mathematics and Physics*. Chapman and Hall/CRC, 2001.

[57] arXiv:1104.4330, 2011.

[58] Henryk Iwaniec and Emmanuel Kowalski. *Analytic Number Theory*, volume 53 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 2004.

[59] E. C. Titchmarsh. *The Theory of the Riemann Zeta-Function*. Oxford University Press, Oxford, 2 edition, 1986. Revised by D. R. Heath-Brown.

[60] Haobo Ma. Riemann ground state: Holographic trace formulas, abel finite parts, and a protocol derivation of the riemann hypothesis in hpa–omega. Companion trace-formula manuscript, 2025.

[61] Michael Rosen. *Number Theory in Function Fields*, volume 210 of *Graduate Texts in Mathematics*. Springer, 2002.

[62] Eugene P. Wigner. Lower limit for the energy derivative of the scattering phase shift. *Physical Review*, 98(1):145–147, 1955.

[63] F. T. Smith. Lifetime matrix in collision theory. *Physical Review*, 118(1):349–356, 1960.