An Experimental Operator Framework for the Riemann Hypothesis

Certified Zero Pipelines, Explicit Formula Residua, and Operator Scoring

Enter Yourname

December 20, 2025

Contents

1 Motivation and Context

1.1 The status of the Riemann Hypothesis

The Riemann Hypothesis asserts that all nontrivial zeros of the Riemann zeta function

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s}, \qquad \mathrm{Re}(s) > 1,$$

lie on the critical line $\mathrm{Re}(s) = \frac{1}{2}$ after analytic continuation. Its truth would imply optimal error bounds in the prime number theorem and profoundly constrain the fine-scale structure of prime number distributions.

Over more than a century, RH has accumulated strong indirect evidence: millions of zeros have been numerically verified on the critical line, deep analogies with random matrix theory have emerged, and multiple equivalent formulations have been established. Nevertheless, these results fall short of a proof, and no consensus exists on the correct conceptual pathway toward one.

1.2 The Hilbert–Pólya idea and its limitations

A long-standing heuristic proposes that the imaginary parts of the nontrivial zeros of $\zeta(s)$ arise as eigenvalues of a self-adjoint operator. Such an operator, if explicitly constructed, would imply RH immediately through the reality of its spectrum.

Despite its appeal, the Hilbert–Pólya idea faces several obstacles. Most proposed operators are either formally defined, lack rigorous spectral control, or fail to demonstrate compatibility with the arithmetic structure of primes. In particular, reproducing the zero spectrum alone is insufficient: any viable operator must also encode the prime–zero duality captured by the explicit formulas of analytic number theory.

1.3 Why an experimental operator framework is needed

Numerical experiments play an essential role in modern research on RH, yet they are often ad hoc, non-reproducible, or disconnected from operator-theoretic hypotheses. Lists of computed zeros, statistical tests of level spacings, and heuristic comparisons with random matrix ensembles are frequently treated as isolated evidence.

The absence of a unified experimental framework leads to two recurring problems. First, numerical results are difficult to compare across different studies. Second, operator proposals cannot be systematically falsified, as failure modes are rarely formalized.

The motivation of this work is to address these issues by defining a structured experimental environment in which operator-based approaches to RH can be evaluated under common constraints. Rather than searching directly for a proof, we focus on constructing a pipeline that enforces consistency between zeros, primes, and operator spectra.

1.4 Scope and philosophy of the present work

The framework developed in this paper is intentionally conservative. We make no claims beyond what can be numerically certified or logically constrained. Experimental agreement is not interpreted as proof, but as evidence of structural compatibility.

Our guiding principle is the following: a candidate operator relevant to the Riemann Hypothesis must survive simultaneous tests imposed by certified zero computations, explicit prime–zero relations, and stability under refinement of numerical scales.

By formalizing these requirements, we aim to transform operator-based research on RH from a collection of isolated ideas into a cumulative, test-driven scientific program.

2 Certified Computation of Zeta Zeros

2.1 Motivation for certification

Numerical verification of the Riemann Hypothesis has progressed to extraordinary heights, with trillions of zeros confirmed on the critical line. However, raw numerical verification alone is insufficient for operator-based or trace-formula investigations. Any experimental framework intended to constrain candidate operators must rely on zero data whose completeness and correctness are themselves certified within controlled error bounds.

In particular, the absence of certification leads to two critical risks: missing zeros in a given interval and spurious numerical artifacts falsely identified as zeros. Both failure modes can invalidate subsequent spectral or arithmetic analyses.

For these reasons, we adopt a certified zero pipeline, in which each computed zero is accompanied by a verification procedure that guarantees both existence and uniqueness within a prescribed interval.

2.2 The Riemann–Siegel Z-function

Let

$$Z(t) = e^{i\theta(t)}\zeta\left(\tfrac{1}{2} + it\right),$$

where $\theta(t)$ is the Riemann–Siegel theta function. For real t, the function $Z(t)$ is real-valued, and its real zeros correspond exactly to the nontrivial zeros of $\zeta(s)$ on the critical line.

The numerical computation of $Z(t)$ is significantly more stable than direct evaluation of $\zeta(s)$, and it provides a natural foundation for detecting zeros via sign changes.

2.3 Bracketing and root refinement

The first stage of the pipeline consists of scanning $Z(t)$ over an interval $[0, T_{\max}]$ using an adaptive step size Δt. Whenever a sign change

$$Z(t_k) \cdot Z(t_{k+1}) < 0$$

is detected, an interval $[a, b] = [t_k, t_{k+1}]$ is recorded as a zero bracket.

Within each bracket, a root refinement procedure is applied. Any standard method may be used, such as bisection, Newton iteration, or hybrid schemes, provided that convergence is monotone and error bounds are explicitly tracked.

The output of this stage is a list of candidate zeros $\{\gamma_n\}$, each associated with a certified interval of width at most ε.

2.4 Counting zeros and checkpoint validation

Bracketing and refinement alone do not guarantee that all zeros in a given interval have been found. To certify completeness, we employ counting formulas derived from analytic number theory.

Let $N(T)$ denote the number of nontrivial zeros with imaginary part in $(0, T]$. The Riemann–von Mangoldt formula states

$$N(T) = \frac{T}{2\pi}\log\left(\frac{T}{2\pi}\right) - \frac{T}{2\pi} + O(\log T).$$

In practice, we use this formula together with explicit error bounds to define checkpoint heights $0 < T_1 < T_2 < \cdots < T_k$. At each checkpoint T_j, the number of detected zeros below T_j is compared against the predicted value $N(T_j)$. Agreement within certified bounds serves as evidence that no zeros have been missed.

More stringent validation can be achieved via Turing-type methods, which compare integrals of $\arg \zeta(s)$ with zero counts. The framework is compatible with such enhancements, although they are not strictly required for moderate ranges of T.

2.5 Definition of a certified zero set

We formalize the output of the pipeline as follows.

Definition 2.1 (Certified zero set). A finite set

$$\mathcal{Z}(T) = \{\gamma_1, \gamma_2, \ldots, \gamma_N\}$$

is called a *certified zero set up to height* T if the following conditions hold:

1. Each γ_n is contained in a verified bracket on which $Z(t)$ changes sign.
2. Each γ_n is refined to an interval of width at most ε.
3. The total number of elements of $\mathcal{Z}(T)$ agrees with the predicted zero count $N(T)$ within certified error bounds.

Only certified zero sets are admitted as input to subsequent stages of the framework.

2.6 Role within the operator framework

Certified zero sets serve as the spectral reference data against which candidate operators are evaluated. Any operator proposed as a realization of the Hilbert–Pólya idea must reproduce these zeros as eigenvalues within comparable accuracy.

Equally important, certified zero sets enable controlled truncation of explicit formulas, as the contribution of omitted zeros can be bounded quantitatively. This property is essential for distinguishing genuine arithmetic structure from numerical artifacts.

By isolating zero computation as a rigorously constrained preprocessing step, the framework ensures that all later conclusions rest on a stable and verifiable foundation.

3 The Explicit Formula as a Diagnostic Tool

3.1 Prime–zero duality

One of the most profound links between prime numbers and the zeros of the Riemann zeta function is provided by the explicit formulas of analytic number theory. These formulas express weighted sums over primes in terms of sums over nontrivial zeros, revealing a deep spectral duality.

For experimental operator-based investigations, the explicit formula plays a central role: it provides a consistency constraint that cannot be satisfied by reproducing the zero spectrum alone. Any viable Hilbert–Pólya type operator must not only generate the correct zeros, but must do so in a manner compatible with the arithmetic structure of primes.

3.2 The Chebyshev function

Let $\Lambda(n)$ denote the von Mangoldt function, and define the Chebyshev function

$$\psi(x) = \sum_{n \le x} \Lambda(n).$$

The function $\psi(x)$ encodes prime powers and admits an explicit representation involving the nontrivial zeros of $\zeta(s)$.

In its classical form, the explicit formula reads

$$\psi(x) = x - \sum_{\rho} \frac{x^{\rho}}{\rho} - \log(2\pi) - \frac{1}{2}\log\bigl(1 - x^{-2}\bigr),$$

where the sum runs over nontrivial zeros ρ of $\zeta(s)$. For experimental purposes, the precise treatment of lower-order terms is less important than the controlled interplay between the prime sum and the zero sum.

3.3 Truncated zero sums

In practice, only finitely many zeros are available. Let $\mathcal{Z}(T)$ denote a certified zero set up to height T, as defined in Section 2. We define the truncated explicit formula

$$\psi_T(x) = x - \sum_{\rho \in \mathcal{Z}(T)} \frac{x^{\rho}}{\rho},$$

with the understanding that omitted terms contribute a truncation error that depends explicitly on T and x.

The use of certified zero sets ensures that truncation errors can be bounded quantitatively, rather than treated heuristically. This property is essential for interpreting discrepancies between arithmetic data and spectral reconstructions.

3.4 Smoothing and test functions

Direct evaluation of $\psi(x)$ and $\psi_T(x)$ is sensitive to sharp cutoffs and numerical noise. To mitigate these effects, we introduce smoothing via test functions.

Let f be a compactly supported or rapidly decaying test function. We consider smoothed quantities of the form

$$\Psi_f(X) = \sum_{n=1}^{\infty} \Lambda(n)\, f(\log n - \log X),$$

and define the corresponding zero-based approximation

$$\Psi_{f,T}(X) = X\widehat{f}(0) - \sum_{\rho \in \mathcal{Z}(T)} X^{\rho}\, \widehat{f}(\rho),$$

where $\widehat{f}$ denotes the Mellin transform of f.

Smoothing improves numerical stability and isolates structural discrepancies from high-frequency artifacts. The framework is compatible with a wide class of test functions, allowing sensitivity analysis across scales.

3.5 Residual functions

The central diagnostic quantity introduced in this work is the residual function

$$R_T(X) = \Psi_f(X) - \Psi_{f,T}(X).$$

Under ideal circumstances, the residual should consist only of:

- truncation effects due to missing zeros above T,
- controllable smoothing artifacts,
- lower-order deterministic terms.

Persistent or structured deviations in $R_T(X)$, especially those stable under refinement of T and variation of the test function f, signal incompatibility between the zero data and the underlying arithmetic structure.

3.6 Interpretation as a diagnostic criterion

The residual function provides a powerful filter for candidate operator models. An operator that reproduces the zero spectrum but fails to produce residuals consistent with prime data can be systematically rejected.

Conversely, operators whose induced spectra yield small and stable residuals across multiple scales demonstrate a higher degree of arithmetic compatibility.

Importantly, residual minimization is not interpreted as evidence of a proof. Rather, it serves as a falsification mechanism: operators that fail the explicit formula test cannot represent the arithmetic dynamics encoded by the zeta function.

3.7 Role within the experimental framework

Within the overall framework, the explicit formula occupies a mediating position between spectral data and arithmetic data. Certified zero sets enter as controlled inputs, while prime sums provide independent constraints.

This dual perspective transforms the Riemann Hypothesis from a purely spectral question into a coupled spectral–arithmetic consistency problem. The explicit formula thus functions as the primary diagnostic bridge linking operators, zeros, and primes within a single experimental architecture.

4 Operator Targets and Hilbert–Pólya Constraints

4.1 From spectral heuristics to constrained operators

The Hilbert–Pólya idea suggests that the imaginary parts of the nontrivial zeros of the Riemann zeta function arise as eigenvalues of a self-adjoint operator. While this heuristic has guided a large body of work, it does not by itself specify the mathematical or physical nature of such an operator.

In an experimental setting, the operator should not be viewed as an abstract existence statement, but as a concrete object subject to testable constraints. The purpose of this section is to articulate the minimal structural requirements that any candidate operator must satisfy in order to be meaningfully evaluated within the present framework.

4.2 Self-adjointness and spectral reality

Let $\mathcal{H}$ be a Hilbert space and H a densely defined linear operator on $\mathcal{H}$. A necessary condition for compatibility with the Hilbert–Pólya program is that H be self-adjoint, or at least essentially self-adjoint on a natural core domain.

Self-adjointness ensures that the spectrum of H is real and that its spectral measure is well-defined. In the present framework, we do not require a full proof of self-adjointness at the outset; however, the operator must admit a well-controlled symmetric form and demonstrate numerical stability consistent with an underlying self-adjoint structure.

Operators that are formally symmetric but exhibit unstable or non-real spectra under refinement are excluded at this stage.

4.3 Spectral correspondence with zeta zeros

The central spectral constraint is the following: the spectrum of the operator H must reproduce the imaginary parts of the nontrivial zeros of $\zeta(s)$.

Formally, if $\{\gamma_n\}$ denotes a certified zero set, then the operator should admit a sequence of eigenvalues $\{\lambda_n\}$ such that

$$\lambda_n \approx \gamma_n,$$

up to controlled numerical error.

This correspondence is understood in an asymptotic sense. Low-lying discrepancies may be tolerated, provided that the agreement improves with increasing spectral resolution and refinement of the operator approximation.

The framework explicitly avoids overfitting: an operator whose spectrum is artificially tuned to match a finite zero list without structural justification is considered invalid.

4.4 Trace regularization and test functions

Most operators capable of producing an unbounded sequence of eigenvalues are not trace class. As a result, expressions of the form $\mathrm{Tr}(f(H))$ require regularization.

We therefore require that candidate operators admit a well-defined regularized trace for a suitable class of test functions f. Typical examples include:

- heat traces $\mathrm{Tr}(e^{-tH})$,
- spectral zeta functions $\mathrm{Tr}(H^{-s})$,
- smoothed spectral sums $\sum_n f(\lambda_n)$.

The regularized trace must be compatible with the explicit formula diagnostics introduced in Section 3. In particular, trace expressions derived from H should reproduce prime-weighted terms up to controlled residuals.

4.5 Arithmetic compatibility

Reproducing the zero spectrum alone is insufficient. The operator must encode arithmetic information consistent with the prime–zero duality.

Within the present framework, this requirement is enforced indirectly via the explicit formula residuals. An operator whose spectrum matches the zeros but whose induced trace expressions fail to approximate prime sums is rejected as arithmetically incompatible.

This criterion reflects the guiding philosophy that the Riemann Hypothesis is not purely spectral, but fundamentally arithmetic in nature.

4.6 Stability under refinement

A key experimental constraint is stability under refinement. Refinement may include:

- increasing the number of basis states,
- extending the spectral window,
- tightening numerical tolerances,
- modifying smoothing parameters.

A viable operator model must exhibit convergence of its spectral and trace properties under such refinements. Operators whose apparent agreement deteriorates with improved resolution are considered artifacts of approximation.

4.7 Minimal admissibility criteria

We summarize the admissibility requirements for candidate operators as follows.

Criterion 4.1 (Admissible operator). An operator H is admissible within the framework if it satisfies:

1. formal symmetry and numerical evidence of (essential) self-adjointness,
2. asymptotic spectral correspondence with certified zeta zeros,
3. existence of compatible regularized traces,
4. consistency with explicit formula diagnostics,
5. stability under systematic refinement.

Only operators meeting these criteria are admitted to the scoring and comparison stage introduced in subsequent sections.

4.8 Position within the overall program

This section establishes the conceptual boundary between speculative operator proposals and experimentally constrained models. By fixing explicit admissibility criteria, the framework transforms the Hilbert–Pólya idea from a heuristic vision into a test-driven research program.

The next section applies these constraints to arithmetic transfer operators and related constructions, illustrating how concrete models can be embedded within the framework without presupposing their correctness.

5 Embedding the SMRK Hamiltonian into the Framework

5.1 From arithmetic transfers to a Hamiltonian

The SMRK Hamiltonian arises from the idea that arithmetic structure can be encoded via transfer operations acting on discrete states. Rather than postulating an abstract spectral operator, the construction begins with explicit arithmetic transitions, most notably multiplicative shifts indexed by prime numbers.

Formally, the SMRK Hamiltonian is conceived as a linear operator acting on a Hilbert space of arithmetic states, such as $\ell^2(\mathbb{N})$ or a related weighted space. Its defining feature is

that transitions between states are governed by prime-indexed transfer rules, reflecting the multiplicative structure of the integers.

In this sense, the SMRK Hamiltonian is not introduced as a solution to the Riemann Hypothesis, but as a concrete arithmetic candidate that can be subjected to the admissibility criteria defined in Section 4.

5.2 Structural form of the SMRK Hamiltonian

At an abstract level, the SMRK Hamiltonian may be written schematically as

$$(H_{\mathrm{SMRK}}\psi)(n) = \sum_{p\in\mathcal{P}} w_p(n)\,\psi(pn) + \sum_{p\in\mathcal{P}} v_p(n)\,\psi(n/p) + U(n)\,\psi(n),$$

where $\mathcal{P}$ denotes the set of prime numbers, $w_p(n)$ and $v_p(n)$ are prime-dependent weights, and $U(n)$ is a diagonal potential term.

The precise choice of weights and potential is not fixed at this stage. Different parameterizations correspond to different members of a family of operators, all of which may be evaluated within the same framework.

This general structure highlights the interpretation of H_{SMRK} as an arithmetic transfer Hamiltonian: states are shifted multiplicatively, rather than additively as in differential operators.

5.3 Relation to Hilbert–Pólya constraints

We now relate the SMRK Hamiltonian to the admissibility criteria of Section 4.

Self-adjointness. For suitable symmetric choices of the weights $w_p(n)$ and $v_p(n)$, the operator H_{SMRK} is formally symmetric on a dense domain. Establishing essential self-adjointness remains an open analytical task, but numerical stability under refinement serves as an experimental proxy within the present framework.

Spectral correspondence. Numerical truncations of H_{SMRK} yield finite-dimensional approximations whose spectra can be compared to certified zeta zero sets. Agreement is evaluated asymptotically and tested for stability under increasing truncation size.

Trace regularization. As an unbounded operator, H_{SMRK} is not trace class. However, smoothed spectral traces such as $\mathrm{Tr}(f(H_{\mathrm{SMRK}}))$ are well-defined for suitable test functions f, allowing direct comparison with explicit-formula diagnostics.

Arithmetic compatibility. By construction, the SMRK Hamiltonian incorporates prime-indexed structure. This makes it particularly well-suited for explicit formula tests, as prime-weighted trace expressions can be compared directly to Chebyshev-type sums.

Stability. The decisive criterion is stability under refinement: increasing the truncation dimension, extending the prime range, and tightening numerical tolerances. Only if spectral and trace properties converge under such refinements can the operator be considered viable.

5.4 Interpretational stance

It is essential to emphasize that the SMRK Hamiltonian is not assumed to be the operator postulated by the Hilbert–Pólya conjecture. Rather, it is treated as a test case: a concrete arithmetic Hamiltonian whose compatibility with RH-related constraints can be quantitatively evaluated.

Failure of the SMRK Hamiltonian to meet any admissibility criterion does not invalidate the framework. On the contrary, such failures provide valuable information about which structural features are incompatible with the arithmetic reality encoded by the zeta function.

5.5 Role within the experimental program

Within the broader program, the SMRK Hamiltonian serves three roles:

1. as a nontrivial arithmetic operator with explicit prime structure,
2. as a benchmark for testing the diagnostic power of the framework,
3. as a starting point for exploring broader classes of arithmetic Hamiltonians.

In the next section, we introduce a scoring protocol that quantifies how well the SMRK Hamiltonian and other candidate operators satisfy the admissibility criteria, enabling systematic comparison and iterative refinement.

6 Operator Scoring and Evaluation Protocol

6.1 Purpose of a scoring framework

The admissibility criteria defined in Section 4 establish qualitative requirements for candidate Hilbert–Pólya type operators. To support systematic comparison and refinement, these requirements must be translated into quantitative or semi-quantitative measures.

The purpose of the scoring protocol is not to declare a "winner", but to provide a structured method for ranking operators, detecting failure modes, and guiding iterative improvements. Scores are interpreted diagnostically, not as proofs.

6.2 Overview of evaluation dimensions

Each candidate operator H is evaluated along several independent dimensions:

1. spectral correspondence with certified zeta zeros,
2. statistical properties of level spacings,
3. compatibility with explicit formula diagnostics,
4. stability under refinement,
5. numerical robustness and convergence behavior.

Each dimension yields a partial score together with uncertainty indicators. The combined score is a vector, not a single scalar quantity.

6.3 Spectral matching score

Let $\{\gamma_n\}$ denote a certified zero set and $\{\lambda_n\}$ the computed eigenvalues of a finite-dimensional approximation of H, ordered increasingly.

We define a normalized spectral deviation

$$\Delta_{\text{spec}}(N) = \frac{1}{N} \sum_{n=1}^{N} \frac{|\lambda_n - \gamma_n|}{\gamma_n}.$$

A decreasing trend of $\Delta_{\text{spec}}(N)$ with increasing N and refinement of the operator approximation is taken as evidence of asymptotic spectral correspondence.

Operators exhibiting non-convergent or oscillatory behavior in this metric are penalized.

6.4 Level spacing statistics

Beyond pointwise spectral agreement, the local statistics of level spacings provide an important consistency check.

Let

$$s_n = \lambda_{n+1} - \lambda_n$$

denote nearest-neighbor spacings, and let $\tilde{s}_n$ be the unfolded spacings. We compare the empirical distribution of $\tilde{s}_n$ against random matrix universality classes, most notably the Gaussian Unitary Ensemble (GUE).

Deviation from the expected distribution is quantified using standard measures such as the Kolmogorov–Smirnov distance or integrated squared error. These statistics are interpreted cautiously: agreement supports compatibility, but does not constitute evidence of correctness.

6.5 Explicit formula residual score

Using the diagnostics of Section 3, we compute residual functions

$$R_T(X) = \Psi_f(X) - \Psi_{f,T}(X),$$

where $\Psi_{f,T}$ is constructed from the operator-induced spectrum.

A residual norm is defined, for example, as

$$\|R_T\|^2 = \int_{X_{\min}}^{X_{\max}} |R_T(X)|^2 \, dX.$$

Scores are assigned based on:

- magnitude of the residual norm,
- stability of the residual under changes in T,
- robustness under variation of the test function f.

Persistent structured residuals are interpreted as evidence of arithmetic incompatibility.

6.6 Stability and refinement score

Stability under refinement is assessed by repeating all scoring steps under systematic changes of resolution, including:

- increased truncation dimension,
- extended prime ranges,
- tighter numerical tolerances,
- denser sampling grids.

Let $S_i^{(k)}$ denote the score in dimension i at refinement level k. Stability is quantified by convergence or bounded variation of $S_i^{(k)}$ as k increases.

Operators whose apparent performance degrades under refinement receive a low stability score.

6.7 Composite score representation

Rather than collapsing results into a single scalar value, we represent the evaluation outcome as a score vector

$$\mathbf{S}(H) = (S_{\text{spec}}, S_{\text{spacing}}, S_{\text{explicit}}, S_{\text{stability}}, S_{\text{robustness}}).$$

This representation reflects the multi-faceted nature of the problem and avoids masking deficiencies in one dimension by strong performance in another.

6.8 Interpretation and falsification

The scoring protocol is explicitly falsification-oriented. An operator that fails one or more dimensions is not "almost correct"; it is incompatible with the imposed constraints at the tested resolution.

Conversely, a high score across all dimensions does not imply correctness or a proof of RH. It indicates only that, within the tested regime, the operator remains consistent with known spectral and arithmetic data.

6.9 Role within the experimental program

The scoring protocol transforms operator-based approaches to the Riemann Hypothesis into a cumulative experimental program. Results obtained for one operator can be directly compared to those of another, and improvements can be tracked quantitatively.

In this sense, the protocol provides a missing methodological layer between speculative operator constructions and rigorous mathematical analysis, clarifying which candidates merit deeper theoretical investigation.

7 Failure Modes and Limitations

7.1 Numerical experiments versus mathematical proof

The framework presented in this work is explicitly experimental. While it imposes strong consistency constraints on candidate operators, it does not constitute a proof of the Riemann Hypothesis.

Numerical agreement, even across large spectral ranges and multiple diagnostic tests, remains inherently finite and approximate. Consequently, all conclusions drawn within this framework must be interpreted as conditional and provisional.

This distinction is essential: the framework is designed to *exclude* incompatible operator models, not to certify correctness in the logical sense required for a proof.

7.2 Finite truncation effects

All numerical implementations operate on finite data: a finite number of zeros, a finite operator truncation, and finite numerical precision. Truncation effects can produce apparent agreement that does not persist under extension.

In particular, an operator may be tuned to reproduce a finite initial segment of zeta zeros while failing asymptotically. Such overfitting is explicitly discouraged by the stability and refinement tests, but cannot be ruled out a priori.

7.3 Sensitivity to smoothing and test functions

Explicit formula diagnostics rely on smoothing via test functions. Different choices of test functions probe different scales and aspects of prime–zero interactions.

Apparent agreement for a specific choice of smoothing may not generalize. For this reason, robustness under variation of test functions is treated as a core stability criterion. Operators whose performance depends critically on a narrow class of test functions are considered fragile.

7.4 Statistical ambiguity in level spacings

Level spacing statistics, including comparisons with random matrix ensembles, are inherently statistical. Finite samples may exhibit fluctuations that mimic or obscure universal behavior.

Agreement with Gaussian Unitary Ensemble (GUE) statistics supports compatibility with known phenomenology, but does not uniquely characterize the zeta zeros. Conversely, deviations from GUE do not necessarily invalidate an operator model, especially at low spectral heights.

These tests must therefore be interpreted as supportive diagnostics, not decisive criteria.

7.5 Operator domain and self-adjointness issues

Formal symmetry of an operator does not guarantee self-adjointness. Subtle domain issues may arise, particularly for unbounded operators defined via arithmetic transfers.

Numerical stability alone cannot substitute for rigorous analysis of essential self-adjointness. An operator may appear well-behaved numerically while lacking a mathematically well-defined spectral theory.

The framework highlights such issues, but resolving them ultimately requires analytic techniques beyond its scope.

7.6 Dependence on certified zero data

The reliability of all subsequent diagnostics depends on the correctness and completeness of the certified zero sets. Errors or omissions at this stage can propagate through the entire pipeline.

While checkpoint validation and optional Turing-type methods reduce this risk, they do not eliminate it entirely. Future extensions of the framework may incorporate stronger certification techniques as they become available.

7.7 Computational limitations

Large-scale experiments require substantial computational resources. As spectral windows and truncation dimensions increase, memory and runtime constraints may limit achievable resolutions.

These practical limitations necessitate trade-offs between depth and breadth of exploration. The framework is therefore best viewed as an exploratory tool rather than an exhaustive verifier.

7.8 Scope of conclusions

The conclusions supported by this framework are necessarily limited in scope. A high-scoring operator is best interpreted as *consistent with current experimental constraints*, not as confirmed or unique.

Likewise, failure of a particular operator, including the SMRK Hamiltonian, does not imply that the Hilbert–Pólya program itself is misguided. Rather, it refines our understanding of which structural features are incompatible with known arithmetic data.

7.9 Summary

By explicitly acknowledging its limitations, the framework avoids overinterpretation of numerical evidence. Its strength lies not in claiming resolution, but in providing a disciplined environment for eliminating incorrect ideas and focusing analytical effort on the most promising candidates.

8 Outlook and Future Directions

8.1 From experimental consistency to analytic structure

The framework developed in this work establishes a disciplined experimental environment for operator-based investigations of the Riemann Hypothesis. By enforcing simultaneous consistency with certified zero data, explicit prime–zero relations, and stability under refinement, it narrows the space of viable operator candidates.

A natural next step is to translate experimentally successful structures into objects amenable to rigorous analysis. In particular, operators that score highly across all dimensions warrant detailed investigation of their domains, closures, and self-adjoint extensions. Such analysis lies beyond the scope of the present work, but the framework provides a principled filter for selecting promising targets.

8.2 Toward rigor in operator analysis

The ultimate goal of the Hilbert–Pólya program is not numerical agreement, but a mathematically precise correspondence between arithmetic structure and spectral theory. For candidate operators identified by the framework, future work may address:

- proofs of essential self-adjointness,
- characterization of spectral measures,

- derivation of trace formulas from first principles,
- control of error terms and asymptotics.

The experimental framework can inform these efforts by highlighting which features appear structurally necessary and which are likely artifacts of approximation.

8.3 Extension to families of L-functions

Although this work focuses on the Riemann zeta function, the framework is not intrinsically limited to it. Many components, including certified zero pipelines and explicit formula diagnostics, extend naturally to families of L-functions.

Exploring operator models that simultaneously encode multiple L-functions may reveal deeper structural principles governing arithmetic spectra. Such extensions offer a pathway from isolated conjectures toward a unified operator-theoretic view of analytic number theory.

8.4 Computational scaling and distributed architectures

As numerical resolution increases, computational demands become significant. Future implementations may benefit from distributed or parallel architectures, allowing larger spectral windows and higher truncation dimensions to be explored.

The framework is compatible with such extensions, provided reproducibility and data integrity are preserved. Distributed computation may thus serve as an enabling technology, not as a conceptual dependency.

8.5 Interaction with broader operator frameworks

The experimental methodology outlined here is not restricted to a single operator construction. Arithmetic transfer Hamiltonians, differential operators, and hybrid models can all be evaluated within the same structure.

This flexibility encourages cross-fertilization between different approaches to the Riemann Hypothesis and reduces fragmentation in operator-based research. Negative results are as informative as positive ones, clarifying which conceptual directions deserve further attention.

8.6 Final perspective

The Riemann Hypothesis occupies a unique position at the intersection of arithmetic, analysis, and spectral theory. While a proof remains elusive, progress need not be confined to isolated theoretical advances.

By treating operator proposals as experimentally testable objects, the framework presented here offers a path toward cumulative understanding. Its value lies not in resolving the conjecture directly, but in structuring inquiry, eliminating incompatible ideas, and guiding analytical effort toward the most coherent and constrained models.

In this sense, the framework aims to serve as an intermediate layer between numerical exploration and rigorous mathematics, supporting steady progress on one of the deepest problems in mathematics.

A Numerical Protocols (Pseudocode)

This appendix records the core numerical protocols referenced in Sections 2–3. The purpose is not to prescribe a single implementation, but to fix a reproducible logical structure: *zero discovery → root refinement → counting / checkpoints → certified zero set.*

A.1 Configuration parameters

We assume a configuration object `cfg` containing numerical parameters. Typical fields include:

```
cfg.T_max              # maximum height
cfg.dt_init            # initial scan step
cfg.dt_min             # minimal step for adaptive scan
cfg.dt_max             # maximal step for adaptive scan
cfg.eps_root           # target bracket width for refined roots
cfg.max_refine_iter    # max iterations in root refinement
cfg.checkpoints        # list of checkpoint heights [T1, T2, ..., Tk]
cfg.count_tolerance    # allowed discrepancy in checkpoint counts
cfg.smoothing_params   # parameters for smoothed explicit-formula tests
```

The framework does not depend on specific values, but requires that all values are logged and version-controlled to ensure reproducibility.

A.2 Protocol P0: Certified zero computation

P0.1: Scan and bracket zeros via sign changes

We use the Riemann–Siegel Z-function $Z(t)$ on the critical line. The scan step may be adaptive.

```
function scan_brackets(T_max, cfg):
    t = 0
    dt = cfg.dt_init
    brackets = []

    Z_prev = Z(t)

    while t < T_max:
        t_next = min(t + dt, T_max)
        Z_next = Z(t_next)

        if Z_prev == 0:
            # Rare in practice; treat as degenerate bracket
            brackets.append([t, t])
        else if Z_prev * Z_next < 0:
            brackets.append([t, t_next])

        # Optional adaptive stepping heuristic:
        # if |Z_next| is small or oscillatory, reduce dt; otherwise increase.
        dt = adapt_step(dt, Z_prev, Z_next, cfg)
```

```
        t = t_next
        Z_prev = Z_next

    return brackets
```

P0.2: Refine each bracket to a certified root interval

Any monotone method is acceptable (bisection / hybrid). We record both the refined interval and the approximate root.

```
function refine_root_interval(a, b, cfg):
    Za = Z(a)
    Zb = Z(b)

    if a == b:
        return (a, b)   # degenerate bracket case

    assert Za * Zb <= 0

    for iter in 1..cfg.max_refine_iter:
        if (b - a) <= cfg.eps_root:
            break

        m = 0.5 * (a + b)
        Zm = Z(m)

        if Zm == 0:
            a = m
            b = m
            break
        else if Za * Zm < 0:
            b = m
            Zb = Zm
        else:
            a = m
            Za = Zm

    return (a, b)
```

P0.3: Build the certified zero set up to each checkpoint

We maintain an ordered list of refined root intervals. We then enforce checkpoint validation against a theoretical count model.

```
function compute_certified_zeros(cfg):
    brackets = scan_brackets(cfg.T_max, cfg)

    refined = []
    for [a,b] in brackets:
        (ra, rb) = refine_root_interval(a, b, cfg)
```

```
        refined.append([ra, rb])

    # Sort by interval center
    refined.sort_by_center()

    # Checkpoints: validate completeness
    for T in cfg.checkpoints:
        count_found = number_of_intervals_with_center_leq(refined, T)
        count_pred  = predicted_zero_count_N(T)

        if abs(count_found - count_pred) > cfg.count_tolerance:
            raise Error("Checkpoint validation failed at T = " + str(T))

    # Output as certified zero set:
    zeros = []
    for [ra, rb] in refined:
        gamma = 0.5*(ra + rb)    # representative value
        zeros.append(gamma)

    return zeros, refined
```

A.3 Protocol P1: Data products and logging

To prevent silent drift between experiments, the following artifacts should be logged:

```
- cfg used for the run (serialized)
- list of refined root intervals [a_n, b_n]
- representative zeros gamma_n
- checkpoint table: (T_j, count_found, count_pred, delta)
- hash / checksum of the produced zero set file
```

A minimal table format for checkpoint output is:

```
T_j | found | predicted | delta
----+-------+-----------+------
... |  ...  |    ...    |  ...
```

A.4 Protocol P2: Explicit formula residuals (high-level)

Section 3 introduces smoothed prime–zero diagnostics. We record the minimal interface here.

P2.1: Prime-side evaluation

```
function compute_Psi_prime_side(X_grid, f, cfg):
    # Psi_f(X) = sum_{n>=1} Lambda(n) * f(log n - log X)
    # In practice truncate n up to N_max(X) determined by f and cfg.
    Psi = []
    for X in X_grid:
        val = 0
        for n in relevant_n(X, f, cfg):
```

```
            val += Lambda(n) * f(log(n) - log(X))
        Psi.append(val)
    return Psi
```

P2.2: Zero-side evaluation (truncated)

```
function compute_Psi_zero_side(X_grid, f_hat, zeros, cfg):
    # Psi_{f,T}(X) = X * f_hat(0) - sum_{rho in Z(T)} X^{rho} * f_hat(rho)
    # Here rho = 1/2 + i*gamma for critical-line zeros.
    PsiT = []
    for X in X_grid:
        val = X * f_hat(0)
        for gamma in zeros:
            rho = 0.5 + i*gamma
            val -= X**rho * f_hat(rho)
        PsiT.append(real(val))  # after pairing +/-gamma, result is real
    return PsiT
```

P2.3: Residual construction and stability checks

```
function compute_residuals(X_grid, Psi, PsiT):
    R = []
    for k in 1..len(X_grid):
        R.append(Psi[k] - PsiT[k])
    return R

function stability_sweep(cfg_list):
    # Vary T, smoothing, grid density, etc.
    # Track whether residual structure persists under refinement.
    results = []
    for cfg in cfg_list:
        zeros, intervals = compute_certified_zeros(cfg)
        Psi  = compute_Psi_prime_side(cfg.X_grid, cfg.f, cfg)
        PsiT = compute_Psi_zero_side(cfg.X_grid, cfg.f_hat, zeros, cfg)
        R    = compute_residuals(cfg.X_grid, Psi, PsiT)
        results.append(summary_statistics(R, cfg))
    return results
```

A.5 Notes on certification strength

The above protocols define the *logical* structure of certification. For increased rigor, checkpoint validation may be strengthened by Turing-type methods and explicit bounds on $\arg \zeta(s)$ integrals. The framework is compatible with such enhancements; the minimal requirement is that the resulting zero set is treated as certified input for subsequent spectral and explicit-formula computations.

B Reproducibility and Data Integrity

This appendix specifies the minimal reproducibility standard for the experimental framework introduced in Sections 2–3. The goal is to ensure that numerical claims (e.g. "certified zeros

up to height T" or "residual norms across scales") can be independently re-run, verified, and compared.

B.1 Principles

We adopt the following principles:

1. **Determinism where possible.** If an experiment uses randomness (e.g. randomized grids, Monte-Carlo probes), the random seed must be fixed and recorded.
2. **Full provenance.** Every produced dataset must carry enough metadata to uniquely identify the code version, configuration, and environment.
3. **Immutable artifacts.** Primary outputs (zero lists, checkpoint tables, residual arrays) are treated as immutable once published. Any change creates a new artifact with a new hash.
4. **Comparability.** Artifacts must be stored in stable formats with documented schemas, so results can be compared across versions and implementations.

B.2 Version control requirements

All code used to generate numerical artifacts must be version controlled. A minimal record for each run includes:

```
- repository URL / identifier
- git commit hash (or equivalent immutable revision id)
- dirty flag (whether local changes existed)
- branch / tag name (optional)
```

If the experiment is performed outside a version-controlled repository, a snapshot archive of the code must be produced and hashed alongside the outputs.

B.3 Environment capture

The computational environment can affect numerical results. For each run, record at minimum:

```
- OS name and version
- CPU architecture (and optional instruction set flags)
- language/runtime version (Python/Julia/C++ compiler etc.)
- numerical library versions (mpmath, Arb, MPFR, FFT libs, etc.)
- floating-point precision settings (binary64, mp.dps, Arb prec, etc.)
```

If containerization is used (Docker/Podman), the container image hash should be recorded and preferred as a canonical environment reference.

B.4 Configuration logging

All experiment parameters must be serialized and stored as part of the run. We recommend using a plain text format such as JSON or YAML. A minimal configuration must include:

```
- T_max
- scan parameters: dt_init, dt_min, dt_max
- root refinement parameters: eps_root, max_refine_iter
- checkpoint heights and count tolerances
- smoothing parameters for explicit-formula tests
- grids used: X_grid specification, spacing, bounds
```

The configuration file itself must be hashed and referenced by that hash in every output file.

B.5 Artifact formats and schemas

To support long-term comparability, we recommend the following stable formats:

- **Zeros:** a line-oriented text file (CSV/TSV) or JSONL containing one row per zero.
- **Root intervals:** CSV/TSV with columns (a_n, b_n, γ_n) where $\gamma_n = \frac{1}{2}(a_n + b_n)$.
- **Checkpoint table:** CSV/TSV with columns $(T_j, \texttt{found}, \texttt{predicted}, \Delta)$.
- **Residual arrays:** CSV/TSV or a binary format (Parquet/NPY) with explicit column names and units.

If binary formats are used, a text manifest must accompany them to document the schema and the software required to read the file.

B.6 Cryptographic hashing and manifests

Every primary artifact produced by the framework must be hashed. Let $H(\cdot)$ denote a cryptographic hash function (e.g. SHA-256). For each run, produce a manifest file containing:

```
- artifact filename
- artifact byte size
- SHA-256 hash
- creation timestamp (UTC)
- config hash reference
- code revision reference
```

A recommended minimal manifest structure is:

```
run_id: <string>
created_utc: <timestamp>
code_revision: <commit-hash>
config_sha256: <hash>
artifacts:
  - name: zeros_Tmax_<...>.tsv
    bytes: <...>
    sha256: <...>
  - name: checkpoints_<...>.tsv
    bytes: <...>
    sha256: <...>
  - name: residuals_<...>.tsv
    bytes: <...>
    sha256: <...>
```

B.7 Run identifiers and naming conventions

We recommend a deterministic run identifier:

$$\texttt{run\_id} := H(\texttt{code\_revision} \,\|\, \texttt{config\_hash} \,\|\, \texttt{environment\_summary}),$$

where $\|$ denotes concatenation of canonicalized strings.

Artifact filenames should include at least:

- the run identifier (or a prefix of it),
- the maximum height $T_{\max}$ for zero computations,
- the smoothing/test-function label for explicit-formula runs.

Example:

```
zeros_Tmax_1e6_run_ab12cd34.tsv
checkpoints_Tmax_1e6_run_ab12cd34.tsv
residuals_gauss_sigma0p2_Tmax_1e6_run_ab12cd34.tsv
manifest_run_ab12cd34.yaml
```

B.8 Minimal comparability tests

To confirm that two runs are comparable, the following must match:

1. identical code revision (or documented differences),
2. identical configuration hash,
3. compatible numerical precision settings,
4. identical artifact schema versions.

If any of these differ, results may still be scientifically comparable, but should not be treated as byte-identical reproductions. Such cases must be documented explicitly.

B.9 Integrity notes on timestamping

If third-party timestamping is used (e.g. OpenTimestamps), it should be applied to the manifest file rather than to individual artifacts. This anchors the full run state (code revision, configuration, and all output hashes) under a single immutable timestamp commitment.

Timestamping strengthens provenance claims, but does not replace reproducibility requirements: the experiment must remain runnable from the recorded code and configuration.

B.10 Summary

The reproducibility protocol above defines the minimum standard for treating numerical results in this work as verifiable scientific artifacts. The intent is to support cumulative progress: future operator models and refinements can be compared against prior results without ambiguity about computational provenance or data integrity.