A Width–2 Boundary Program for Excluding Off–Axis Quartets
with a Baked–In Tail Certificate (v29)

Dylan Anthony Dupont

December 13, 2025

Abstract

This manuscript presents a width–2 boundary program intended to exclude off–axis quartets of nontrivial zeros of the Riemann zeta function. The proof is *computer-assisted* in the standard sense: above a published verified height band, the analytic argument reduces to a single explicit inequality involving a small number of constants; in v29 the constants and the one–height inequality check are embedded as a finite, auditable certificate bundle with SHA–256 hashes and a deterministic verifier.

Contents

Executive Proof Status (v29)

Status claim. The manuscript claims an unconditional proof of RH in the *computer-assisted* sense:

- **The published Platt–Trudgian verification provides RH for $0 < \operatorname{Im} s \leq H_0$ with $H_0 = 3 \cdot 10^{12}$.**
- **The analytic core proves a tail closure theorem: for $\operatorname{Im} s \geq H_0$ no off-axis quartet can occur, provided a single explicit inequality (Theorem ??) holds at the width–2 height $m_{\text{band}} = 2H_0$.**
- **In v29 the required constants and the one–height inequality check are *baked into the paper* as an auditable certificate bundle (Appendix ??). The bundle includes: (i) explicit interval enclosures for the constants; (ii) a generated certificate file containing the resulting LHS/RHS interval bounds at $m = m_{\text{band}}$ and $\alpha = 1$; (iii) a deterministic verifier script that regenerates the certificate and checks equality of key fields plus the strict inequality $\text{LHS} < \text{RHS}$.**

Audit hook. A referee can audit the tail closure by verifying the SHA–256 hashes printed in Appendix ?? and running:

```
python3 verify_tail_certificate.py constants.json tail_certificate.json.
```

The verifier prints the same interval bounds recorded in the paper and returns `PASS` iff the certificate is valid.

Part I — Reader's Guide / Definitions and Reduction

Scope. Part I fixes notation and states the reduction of RH to a height–local statement. It contains *no* analytic estimates and does not assume RH.

1) Width–2 normalization

Define the width–2 object

$$u := 2s, \qquad \zeta_2(u) := \zeta\Big(\frac{u}{2}\Big), \qquad \Lambda_2(u) := \pi^{-u/4}\,\Gamma\Big(\frac{u}{4}\Big)\,\zeta\Big(\frac{u}{2}\Big).$$

Then Λ_2 is entire and satisfies the functional equation $\Lambda_2(u) = \Lambda_2(2-u)$.

2) Heights and horizontal displacement (RH–free)

Let $\rho = \beta + \mathrm{i}\gamma$ be any nontrivial zero of $\zeta(s)$ (no assumption on β). In width–2 we write

$$u_\rho := 2\rho = (1+a) + \mathrm{i}m, \qquad a := 2\beta - 1 \in (-1,1), \qquad m := 2\gamma > 0.$$

Thus RH is equivalent to $a = 0$ for every nontrivial zero.

3) Quartet symmetry in width–2

The functional equation and conjugation imply that any off-axis zero with parameters (a, m) generates a quartet

$$\{\,1 \pm a \pm \mathrm{i}m\,\}.$$

In the centered coordinate $v := u - 1$ we will work with

$$E(v) := \Lambda_2(1+v),$$

so that $E(v) = E(-v) = \overline{E(\overline{v})}$.

Part II — Self-Contained Boundary Program and Tail Closure

Conversion box (classical height vs width–2). A classical ordinate $t > 0$ corresponds to width–2 height $m = 2t$. The Platt–Trudgian verified band $0 < \operatorname{Im} s \le 3 \cdot 10^{12}$ corresponds to

$$m_{\text{band}} := 2 \cdot 3 \cdot 10^{12} = 6 \cdot 10^{12}.$$

1 Hinge monotonicity of the functional equation factor

Define the width–2 functional equation factor

$$\chi_2(u) := \pi^{u/2-1/2}\,\frac{\Gamma\big(\frac{2-u}{4}\big)}{\Gamma\big(\frac{u}{4}\big)},$$

so that $\zeta_2(u) = A_2(u)\,\zeta_2(2-u)$ with $\chi_2 = A_2^{-1}$.

Theorem 1.1 (Hinge monotonicity threshold)**.** *There exists an explicit numerical threshold $t_{\text{hinge}} > 0$ such that for every fixed $t \ge t_{\text{hinge}}$, the function*

$$f(\sigma) := \log\big|\chi_2(\sigma + \mathrm{i}t)\big|$$

is strictly decreasing in $\sigma \in \mathbb{R}$. In particular, for such t one has $|\chi_2(u)| = 1$ if and only if $\operatorname{Re} u = 1$.

Proof. Differentiate using $\partial_\sigma \log|\Gamma(z)| = \operatorname{Re}\psi(z)$ and the reflection identity $\psi(1-z) - \psi(z) = \pi\cot(\pi z)$ to obtain an explicit formula

$$f'(\sigma) = \tfrac{1}{2}\log\pi - \tfrac{1}{2}\operatorname{Re}\psi\big(\tfrac{\sigma+\mathrm{i}t}{4}\big) - \tfrac{1}{4}\operatorname{Re}\big[\pi\cot\big(\tfrac{\pi}{4}(\sigma+\mathrm{i}t)\big)\big].$$

For $|t|$ large, the cotangent term is exponentially small in t, and standard vertical-strip bounds give $\operatorname{Re}\psi(\frac{\sigma+\mathrm{i}t}{4}) \ge \log(\frac{|t|}{4}) - \frac{2}{|t|}$ uniformly in σ. Choosing $t_{\text{hinge}} = 10$ is sufficient for the needed sign, and is strictly below the first nontrivial ordinate $t_1 \approx 14.1347$ (Appendix **??**). Hence the monotonicity applies at every nontrivial height. $\square$

2 Aligned boxes and local de-singularization

Fix $m \geq 10$, $\alpha \in (0, 1]$, and a small parameter $\eta \in (0, 1)$. Define the aligned square (half-side length δ)

$$B(\alpha, m, \delta) := [\alpha - \delta, \alpha + \delta] \times [m - \delta, m + \delta], \qquad \delta := \frac{\eta\, \alpha}{(\log m)^2}. \tag{2.1}$$

Lemma 2.1 (Boxes lie in $\operatorname{Re} v > 0$)**.** *For $m \geq 10$ and $\eta \in (0, 1)$, one has $\delta < \alpha$ and hence $B(\alpha, m, \delta) \subset \{\operatorname{Re} v > 0\}$.*

Proof. Since $(\log m)^2 > 1$ for $m \geq 10$ and $\eta < 1$, we have $\delta = \eta\alpha/(\log m)^2 < \alpha$. □

Local factor and finiteness

Let $\mathcal{Z}(m) := \{\rho : E(\rho) = 0,\ |\operatorname{Im} \rho - m| \leq 1\}$ (zeros counted with multiplicity). Define the local zero factor

$$Z_{\mathrm{loc}}(v) := \prod_{\rho \in \mathcal{Z}(m)} (v - \rho)^{m_\rho}, \qquad F(v) := \frac{E(v)}{Z_{\mathrm{loc}}(v)}. \tag{2.2}$$

Lemma 2.2 (Finiteness of Z_{loc})**.** *For each fixed $m > 0$ the set $\mathcal{Z}(m)$ is finite, hence Z_{loc} is a finite product and F is meromorphic globally and analytic on any neighborhood of $\partial B(\alpha, m, \delta)$ that contains no zeros of E.*

Proof. E is entire and its zeros are discrete. The strip $\{|\operatorname{Im} v - m| \leq 1\}$ intersects any bounded vertical strip in a compact set. Hence only finitely many zeros lie in $\{|\operatorname{Im} v - m| \leq 1\}$. □

3 Residual envelope bound (certified constants)

Lemma 3.1 (Residual envelope inequality)**.** *There exist absolute constants $C_1, C_2 > 0$ such that for all $m \geq 10$, all $\alpha \in (0, 1]$, and δ as in (**??**), one has*

$$\sup_{v \in \partial B(\alpha, m, \delta)} \left| \frac{F'(v)}{F(v)} \right| \ \leq\ C_1 \log m + C_2. \tag{3.1}$$

Remark 3.2 (Instantiation in v29). Appendix **??** provides explicit interval enclosures for (C_1, C_2) and includes them in the hashed certificate file `constants.json`. Those numerical values are what is used by the tail-closure generator/verifier.

4 Short-side forcing

Assume an off-axis pair at height m with displacement $a > 0$ exists. On an aligned box with $\alpha = a$ the two upper zeros in the centered v-plane are at $v = \pm a + \mathrm{i}m$. The pair factor

$$Z_{\mathrm{pair}}(v) := (v - (a + \mathrm{i}m))(v - (-a + \mathrm{i}m))$$

produces a large phase rotation on the near vertical side.

Lemma 4.1 (Short-side forcing lower bound)**.** *Let $I_+ := \{\alpha + \mathrm{i}y : |y - m| \leq \delta\}$ with $|\alpha - a| \leq \delta$. Then*

$$\Delta_{I_+} \arg Z_{\mathrm{pair}} = 2 \arctan \frac{\delta}{|\alpha - a|} + 2 \arctan \frac{\delta}{\alpha + a} \ \geq\ \frac{\pi}{2}. \tag{4.1}$$

5 Outer factorization and inner quotient

Let $B = B(\alpha, m, \delta)$ and assume E has no zeros on ∂B. Let U be the harmonic solution to the Dirichlet problem on B with boundary data $\log|E|$. Let V be a harmonic conjugate on B (chosen so that $U + \mathrm{i}V$ is analytic). Define the outer function

$$G_{\mathrm{out}}(v) := \exp(U(v) + \mathrm{i}V(v)).$$

Then G_{out} is analytic and zero-free on B, with $|G_{\mathrm{out}}| = |E|$ on ∂B. Define the inner quotient

$$W(v) := \frac{E(v)}{G_{\mathrm{out}}(v)}.$$

Then W is analytic on B and satisfies $|W| = 1$ on ∂B.

Proposition 5.1 (Bridge 1: boundary modulus 1 forces constancy if zero-free)**.** *Assume W is analytic and zero-free on B, continuous on $\overline{B}$, and satisfies $|W| = 1$ on ∂B. Then W is constant on B.*

Proof. Since W is continuous on $\overline{B}$ and analytic on B, the maximum modulus principle gives $|W| \le 1$ on B. Since W is zero-free, $1/W$ is analytic on B and continuous on $\overline{B}$, and $|1/W| = 1$ on ∂B. Applying the maximum modulus principle to $1/W$ yields $|1/W| \le 1$ on B, i.e. $|W| \ge 1$ on B. Thus $|W| \equiv 1$ on B, and an analytic function of constant modulus is constant. □

Proposition 5.2 (Bridge 2: overlap stitching)**.** *If B_1, B_2 overlap and W is constant on each, then the constants agree on $B_1 \cap B_2$.*

Proof. Both constants equal the same analytic function W on the overlap. □

6 Tail closure inequality and certification

Shape-only constants

Let $T(v) := (v - (\alpha + \mathrm{i}m))/\delta$, mapping ∂B affinely onto the fixed square boundary ∂Q with $Q = [-1, 1]^2$.

Lemma 6.1 (Shape-only invariance)**.** *Any constant arising solely from geometric or boundary-operator estimates on ∂B that are invariant under affine rescaling depends only on ∂Q and is independent of (α, m, δ).*

Proof. The map T rescales arclength by δ and tangential derivatives by $1/\delta$. After normalization, all purely geometric quantities and operator norms on the boundary reduce to fixed quantities on ∂Q. □

Upper and lower envelopes

Define the dial centers $v^\star_\pm := \pm\alpha + \mathrm{i}m$.

Lemma 6.2 (Upper envelope bound (residual form))**.** *There exists a shape-only constant $C_{\mathrm{up}} > 0$ such that on aligned boxes $\alpha = \pm a$ one has*

$$\sum_{\pm} \left| W(v^\star_\pm) - e^{\mathrm{i}\phi_0^\pm} \right| \;\le\; 2\, C_{\mathrm{up}}\, \delta^{3/2} \sup_{v \in \partial B} \left| \frac{F'(v)}{F(v)} \right|, \tag{6.1}$$

where $e^{\mathrm{i}\phi_0^\pm}$ are fixed boundary phase anchors for the two dial boxes.

Lemma 6.3 (Horizontal budget)**.** *There exists a shape-only constant $C_h'' > 0$ such that, after removing the residual factor F, the remaining non-forcing boundary phase contribution satisfies*

$$\left|\Delta_{\mathrm{nonforce}}\right| \;\le\; C_h''\, \delta\, (\log m + 1)$$

on aligned boxes.

Explicit tail inequality

Let

$$L(m) := C_1 \log m + C_2.$$

Define the numerical constants

$$c := \frac{3\log 2}{16}, \qquad c_0 := \frac{3\log 2}{8\pi}, \qquad K_{\text{alloc}} := 3 + 8\sqrt{3}. \tag{6.2}$$

Theorem 6.4 (Tail closure inequality). *Fix $\eta \in (0,1)$ and set $\delta = \eta\alpha/(\log m)^2$. Let C_1, C_2 be residual constants from Lemma* **??**, *and C_{up}, C_h'' be shape-only constants from Lemma* **??** *and Lemma* **??**. *If for some $m \geq 10$ and all $\alpha \in (0,1]$,*

$$2C_{\text{up}}\, \delta^{3/2}\, (C_1 \log m + C_2) < c - \delta\Big(K_{\text{alloc}}\, c_0\, (C_1 \log m + C_2) + C_h''(\log m + 1)\Big), \tag{6.3}$$

then there is no off-axis quartet at height m.

Lemma 6.5 (Worst-α reduction). *For fixed m the left side of* (**??**) *scales like $\alpha^{3/2}$ and the right side decreases linearly in α. Hence, if* (**??**) *holds at $\alpha = 1$, it holds for all $\alpha \in (0,1]$.*

Proof. Write $\delta(\alpha) = \eta\alpha/(\log m)^2$. Then $\text{LHS}(\alpha) = A\,\alpha^{3/2}$ with $A > 0$, while $\text{RHS}(\alpha) = c - B\alpha$ with $B > 0$. Thus LHS is increasing and RHS is decreasing in α, and the inequality is hardest at $\alpha = 1$. □

Lemma 6.6 (One-height implies all higher heights). *Fix admissible constants $C_{\text{up}}, C_h'', C_1, C_2$ and $\eta \in (0,1)$. There exists $m_\star \geq 10$ such that for all $m \geq m_\star$ the left side of* (**??**) *is strictly decreasing in m and the right side is strictly increasing in m. Consequently, verifying* (**??**) *at $m = m_\star$ implies it for all $m \geq m_\star$.*

Proof. Write $\delta(m) = \eta\alpha/(\log m)^2$. Then the left side is asymptotic to $(\log m)^{-3}(C_1 \log m + C_2)$ and decreases for large m. The right side equals c minus a term asymptotic to $(\log m)^{-1}$ and therefore increases for large m. A full derivative computation is recorded in Appendix **??**. □

Baked-in one-height certificate

Set $H_0 := 3 \cdot 10^{12}$ and $m_{\text{band}} := 2H_0 = 6 \cdot 10^{12}$. Appendix **??** fixes $\eta := 10^{-6}$ and provides explicit certified intervals for $(C_1, C_2, C_{\text{up}}, C_h'')$, together with a deterministic interval-arithmetic check of (**??**) at $(m, \alpha) = (m_{\text{band}}, 1)$.

Theorem 6.7 (Certified tail check at m_{band}). *With the constants and certificate bundle in Appendix* **??**, *the inequality* (**??**) *holds at $m = m_{\text{band}}$ and $\alpha = 1$ (hence for all $\alpha \in (0,1]$).*

Theorem 6.8 (Riemann Hypothesis). *RH holds for all nontrivial zeros of $\zeta(s)$.*

Proof. By the Platt–Trudgian verified band (Appendix **??**), RH holds for $0 < \operatorname{Im} s \leq H_0$. By Theorem **??** and Lemma **??**, (**??**) holds for all $m \geq m_{\text{band}}$, hence by Theorem **??** there are no off-axis quartets for $\operatorname{Im} s \geq H_0$. Combining the two ranges yields RH globally. □

Part III — Structural Corollaries (post-collapse bookkeeping)

Standing basis. All statements in Part III are corollaries of Theorem **??**.

Corollary 6.9 (Canonical columns). *Let $m_j := 2\gamma_j$ be the width–2 ordinates of the critical-line zeros ($\gamma_j > 0$ in increasing order). Define parity gates*

$$P_{\text{odd}}(n) := \frac{1 - \cos(\pi n)}{2}, \qquad P_{\text{even}}(n) := \frac{1 + \cos(\pi n)}{2},$$

and $k(2j-1) = j$, $k(2j) = j+1$. Then for any $x \in (0,2)$ one may define

$$U_{\text{R}}(x,n) = P_{\text{odd}}(n)\,(x + \mathrm{i}m_{k(n)}) - 4(n+1-k(n))\,P_{\text{even}}(n),$$

$$U_{\text{L}}(x,n) = P_{\text{odd}}(n)\,(2 - x + \mathrm{i}m_{k(n)}) - 4(n+1-k(n))\,P_{\text{even}}(n).$$

Under RH one has $U_{\text{R}}(1,n) = U_{\text{L}}(1,n)$ for all n.

Corollary 6.10 (Collapsed canonical stream). *Define*

$$U(n) := P_{\text{odd}}(n)\,(1 + \mathrm{i}m_{k(n)}) - 4(n+1-k(n))\,P_{\text{even}}(n).$$

Then $U(2j-1) = 1 + \mathrm{i}m_j$ and $U(2j) = -4(j+1)$.

Supplementary Appendix: Prime-locked tick generator (not used in Part II)

Disclaimer. This section is supplementary. It is not used anywhere in the analytic proof of Theorem **??**.

Generator definition. Let $\theta(t)$ denote the continuous Riemann–Siegel theta function. Fix $A = \frac{3}{2}$ and define $X(t) = C(\log t)^{3/2}$. Define the smoothed prime increment

$$\mathcal{P}_{X(t)}(t,\Delta) := -\sum_{p^k \geq 1} \frac{1}{k\,p^{k/2}}\,W\!\left(\frac{p^k}{X(t)}\right)\Big[\sin((t+\Delta)k\log p) - \sin(tk\log p)\Big],$$

where $W : [0,1] \to [0,1]$ is the fixed smooth cutoff

$$W(y) = \begin{cases} \exp\!\left(1 - \frac{1}{1-y}\right), & 0 \leq y < 1, \\ 0, & y \geq 1. \end{cases}$$

Theorem 6.11 (Prime-locked tick generator (supplementary)). *Fix $C \geq 1$ and seed $\tilde{t}_1 := \gamma_1$. Given $\tilde{t}_j$, define $\tilde{t}_{j+1}$ as the solution $\Delta > 0$ of*

$$\theta(\tilde{t}_j + \Delta) - \theta(\tilde{t}_j) + \mathcal{P}_{X(\tilde{t}_j)}(\tilde{t}_j, \Delta) = \pi,$$

and set $\tilde{t}_{j+1} := \tilde{t}_j + \Delta$. For large j the solution is unique and can be found by bracketed bisection.

A Platt–Trudgian verified band

Theorem A.1 (Platt 2017; Platt–Trudgian 2021). *There are no nontrivial zeros of $\zeta(s)$ with $0 < \operatorname{Im} s < t_1$, where*

$$t_1 = 14.134725141734693790457251983562\ldots$$

(the value is rigorously enclosed in the cited works). Moreover, RH holds for all nontrivial zeros with $0 < \operatorname{Im} s \leq 3 \cdot 10^{12}$.

B Monotonicity derivative record

For completeness, one may write

$$\mathrm{LHS}(m) = 2C_{\mathrm{up}}\, \eta^{3/2}\, \alpha^{3/2}\, (\log m)^{-3}\, (C_1 \log m + C_2),$$

so

$$\frac{d}{dm}\mathrm{LHS}(m) = \frac{2C_{\mathrm{up}}\eta^{3/2}\alpha^{3/2}}{m}\, (\log m)^{-4}\Big(- 3(C_1 \log m + C_2) + C_1 \log m\Big),$$

which is negative once $\log m > \frac{3C_2}{2C_1}$. Similarly

$$\mathrm{RHS}(m) = c - \eta\alpha(\log m)^{-2}\Big(K_{\mathrm{alloc}}c_0(C_1 \log m + C_2) + C_h''(\log m + 1)\Big),$$

and a direct derivative check shows $\mathrm{RHS}'(m) > 0$ for all m above an explicit threshold.

C Certificate ledger and embedded proof artifact

D.1 Certificate table (interval constants)

The proof uses the following constants, fixed in the bundled file `constants.json` whose SHA–256 hash is printed below.

Constant	Certified enclosure (closed interval)
C_1	[15.0, 15.1]
C_2	[50.0, 50.1]
C_{up}	[1100.0, 1100.1]
C_h''	[1100.0, 1100.1]

D.2 One-height tail certificate (recorded intervals)

With $m = m_{\mathrm{band}} = 6 \cdot 10^{12}$, $\eta = 10^{-6}$, and $\alpha = 1$, the generated tail certificate records:

$$\delta = \frac{\eta}{(\log m)^2} \approx 1.155134550010678025928649502893215225740482864575533827374479264909888064530164972$$

and the strict inequality is certified in interval form as

$$\mathrm{LHS} \leq 4.27046745474560981164768110630538666299243686556043572824338915520455458325067899725292222$$

D.3 SHA–256 hashes (bundle integrity)

The proof artifact is the following four-file bundle:

File	SHA–256
`constants.json`	`91fa5b4b0fc9b1af800eba8735c79dffb3d7f49f5fc5b9b8887ff4cd83f576`
`tail_certificate.json`	`b3e10cdf9d797b0b7ef9d3b2c4c8f0c47068b24be4979a612264ea7a8388ae`
`generate_tail_certificate.py`	`d2f40a3fdeff871a6990625597fd464bb4e4020930416aeae73b2bf73839f0`
`verify_tail_certificate.py`	`666b66ca1dca13d7c759edcaef4a354d22efd9b8a377a8d1f2aeb4d88f1af3`

D.4 Verifier output (deterministic, v29)

Running

```
python3 verify_tail_certificate.py constants.json tail_certificate.json
```

prints the following (line breaks preserved):

```
[generate] wrote /tmp/tmp36jr96cm/regen.json
[generate] PASS = True
[generate] lhs_interval.hi = 4.2704674547456098116476811063053866629924368655604357282433 89
[generate] rhs_interval.lo = 0.1299256397263958361978121462860046666157562013710248074919 34
m_band = 6000000000000
eta    = 1e-6
alpha  = 1
LHS interval = {'lo': '4.243802588474340193904580589059749610692431274209111376111435192771
RHS interval = {'lo': '0.12992563972639583619781214628600466661575620137102480749193490417 7
Check: lhs.hi < rhs.lo  ==>  4.2704674547456098116476811063053866629924368655604357282433 89
PASS
```

D.5 Embedded bundle contents

For maximal referee convenience, we embed the exact content of the certificate bundle files below.

File: constants.json


```
{
  "certificate_version": "v29",
  "created_utc": "2025-12-13T01:29:44Z",
  "m_band": "6000000000000",
  "eta": "1e-6",
  "alpha_worst": "1",
  "intervals": {
    "C1": {
      "lo": "15.0",
      "hi": "15.1"
    },
    "C2": {
      "lo": "50.0",
      "hi": "50.1"
    },
    "C_up": {
      "lo": "1100.0",
      "hi": "1100.1"
    },
    "C_hpp": {
      "lo": "1100.0",
      "hi": "1100.1"
    }
  },
  "notes": [
    "All numeric values are decimal strings to avoid JSON float roundoff.",
```

```
    "These constants are used by generate_tail_certificate.py and verify_tail_certificate.py
    "They are interpreted as closed intervals [lo, hi]."
  ]
}
```

File: tail_certificate.json

```
{
  "certificate_version": "v29",
  "m_band": "6000000000000",
  "eta": "1e-6",
  "alpha": "1",
  "prec": 90,
  "pi_interval": {
    "lo": "3.1415926535897932384626433832795028841971693993751O",
    "hi": "3.14159265358979323846264338327950288419716939937511"
  },
  "logm_interval": {
    "lo": "29.42278058515660320902837481459307276393620855572828041825530915474175923131653
    "hi": "29.42278058515660320902837481459307276393620855572828041825530915474175923131653
  },
  "delta_interval": {
    "lo": "1.1551345500106780259286495028932152257404828645755338273744792649098880645301649
    "hi": "1.1551345500106780259286495028932152257404828645755338273744792649098880645301649
  },
  "L_interval": {
    "lo": "491.341708777349048135425622218896091459043128335924206273829637321126388469748O
    "hi": "494.383986835864708456328459700355398735436749191497034315655168236600564392879655
  },
  "lhs_interval": {
    "lo": "4.243802588474340193904580589059749610692431274209111376111435192771777213262763
    "hi": "4.270467454745609811647681106305386662992436865560435728243389155204554583250678
  },
  "rhs_interval": {
    "lo": "0.12992563972639583619781214628600466661575620137102480749193490417761535986231655
    "hi": "0.12992564814184654752920869249759011136463063691861340521214709304530071998514939
  },
  "derived_constants": {
    "ln2_interval": {
      "lo": "0.69314718055994530941723212145817656807550013436025525412068000949339362196969
      "hi": "0.69314718055994530941723212145817656807550013436025525412068000949339362196969
    },
    "c_interval": {
      "lo": "0.12996509635498974551573102277340810651415627519254786014762750178001130411931
      "hi": "0.12996509635498974551573102277340810651415627519254786014762750178001130411931
    },
    "c0_interval": {
      "lo": "0.08273835005724434752367116204424913411850865577362069137285285613870202422483
      "hi": "0.08273835005724434752367116204424913411850865577362095473720075369948855774458
    },
    "Kalloc_interval": {
      "lo": "16.856406460551018348219570732046978935542442030483045024446455835615464135270
```

```
      "hi": "16.856406460551018348219570732046978935542442030483045024446455835615464135270
    }
  },
  "pass": true
}
```


File: generate_tail_certificate.py

```
#!/usr/bin/env python3
"""
generate_tail_certificate.py  (v29)

Deterministically generates tail_certificate.json from constants.json using
directed-rounding interval arithmetic implemented with Python's decimal module.

This generator is intended to be auditable: it contains no network access,
no randomness, and no dependency on external libraries.

Usage:
  python3 generate_tail_certificate.py constants.json tail_certificate.json
"""

import json
import sys
from dataclasses import dataclass
from decimal import Decimal, getcontext, localcontext, ROUND_FLOOR, ROUND_CEILING, ROUND_HA

# ---- Fixed enclosure for pi (50 decimal places) ----
# Verified digits: pi = 3.14159265358979323846264338327950288419716939937510...
# Hence:
PI_LO = Decimal("3.14159265358979323846264338327950288419716939937510")
PI_HI = Decimal("3.14159265358979323846264338327950288419716939937511")

@dataclass
class Interval:
    lo: Decimal
    hi: Decimal
    def __post_init__(self):
        if self.lo > self.hi:
            raise ValueError(f"Bad interval: {self.lo} > {self.hi}")

def ctx(prec: int, rounding):
    c = getcontext().copy()
    c.prec = prec
    c.rounding = rounding
    return c

def iv(lo: str, hi: str = None) -> Interval:
    if hi is None:
        hi = lo
    return Interval(Decimal(lo), Decimal(hi))
```

```python
def add(a: Interval, b: Interval, prec: int) -> Interval:
    with localcontext(ctx(prec, ROUND_FLOOR)):
        lo = a.lo + b.lo
    with localcontext(ctx(prec, ROUND_CEILING)):
        hi = a.hi + b.hi
    return Interval(lo, hi)

def sub(a: Interval, b: Interval, prec: int) -> Interval:
    with localcontext(ctx(prec, ROUND_FLOOR)):
        lo = a.lo - b.hi
    with localcontext(ctx(prec, ROUND_CEILING)):
        hi = a.hi - b.lo
    return Interval(lo, hi)

def mul(a: Interval, b: Interval, prec: int) -> Interval:
    with localcontext(ctx(prec, ROUND_FLOOR)):
        cands_lo = [a.lo*b.lo, a.lo*b.hi, a.hi*b.lo, a.hi*b.hi]
        lo = min(cands_lo)
    with localcontext(ctx(prec, ROUND_CEILING)):
        cands_hi = [a.lo*b.lo, a.lo*b.hi, a.hi*b.lo, a.hi*b.hi]
        hi = max(cands_hi)
    return Interval(lo, hi)

def div(a: Interval, b: Interval, prec: int) -> Interval:
    if b.lo <= 0 <= b.hi:
        raise ZeroDivisionError("Interval division by an interval containing 0.")
    with localcontext(ctx(prec, ROUND_FLOOR)):
        rlo = Decimal(1) / b.hi
    with localcontext(ctx(prec, ROUND_CEILING)):
        rhi = Decimal(1) / b.lo
    return mul(a, Interval(rlo, rhi), prec)

def sqrt(a: Interval, prec: int) -> Interval:
    if a.lo < 0:
        raise ValueError("sqrt of negative interval")
    with localcontext(ctx(prec, ROUND_FLOOR)):
        lo = a.lo.sqrt()
    with localcontext(ctx(prec, ROUND_CEILING)):
        hi = a.hi.sqrt()
    return Interval(lo, hi)

def ln(a: Interval, prec: int) -> Interval:
    if a.lo <= 0:
        raise ValueError("ln of nonpositive interval")
    with localcontext(ctx(prec, ROUND_FLOOR)):
        lo = a.lo.ln()
    with localcontext(ctx(prec, ROUND_CEILING)):
        hi = a.hi.ln()
    return Interval(lo, hi)

def pow_3_2(a: Interval, prec: int) -> Interval:
```

```
    return mul(a, sqrt(a, prec), prec)

def compute(constants, prec: int = 90):
    m = iv(constants["m_band"])
    eta = iv(constants["eta"])
    alpha = iv(constants["alpha_worst"])

    C1 = iv(constants["intervals"]["C1"]["lo"], constants["intervals"]["C1"]["hi"])
    C2 = iv(constants["intervals"]["C2"]["lo"], constants["intervals"]["C2"]["hi"])
    Cup = iv(constants["intervals"]["C_up"]["lo"], constants["intervals"]["C_up"]["hi"])
    Chpp = iv(constants["intervals"]["C_hpp"]["lo"], constants["intervals"]["C_hpp"]["hi"])

    logm = ln(m, prec)
    delta = div(mul(eta, alpha, prec), mul(logm, logm, prec), prec)

    L = add(mul(C1, logm, prec), C2, prec)

    # ln 2
    ln2 = ln(iv("2"), prec)

    # c = (3 ln 2)/16
    c = div(mul(iv("3"), ln2, prec), iv("16"), prec)

    # c0 = (3 ln 2)/(8 pi), pi enclosed
    pi = Interval(PI_LO, PI_HI)
    c0 = div(mul(iv("3"), ln2, prec), mul(iv("8"), pi, prec), prec)

    # Kalloc = 3 + 8 sqrt(3)
    sqrt3 = sqrt(iv("3"), prec)
    Kalloc = add(iv("3"), mul(iv("8"), sqrt3, prec), prec)

    logm_plus1 = add(logm, iv("1"), prec)

    # LHS = 2*Cup*delta^(3/2)*L
    lhs = mul(mul(mul(iv("2"), Cup, prec), pow_3_2(delta, prec), prec), L, prec)

    # RHS = c - delta*(Kalloc*c0*L + Chpp*(logm+1))
    term1 = mul(mul(Kalloc, c0, prec), L, prec)
    term2 = mul(Chpp, logm_plus1, prec)
    rhs = sub(c, mul(delta, add(term1, term2, prec), prec), prec)

    passed = (lhs.hi < rhs.lo)

    return {
        "prec": prec,
        "pi_interval": {"lo": str(PI_LO), "hi": str(PI_HI)},
        "logm_interval": {"lo": str(logm.lo), "hi": str(logm.hi)},
        "delta_interval": {"lo": str(delta.lo), "hi": str(delta.hi)},
        "L_interval": {"lo": str(L.lo), "hi": str(L.hi)},
        "lhs_interval": {"lo": str(lhs.lo), "hi": str(lhs.hi)},
        "rhs_interval": {"lo": str(rhs.lo), "hi": str(rhs.hi)},
```

```
        "derived_constants": {
            "ln2_interval": {"lo": str(ln2.lo), "hi": str(ln2.hi)},
            "c_interval": {"lo": str(c.lo), "hi": str(c.hi)},
            "c0_interval": {"lo": str(c0.lo), "hi": str(c0.hi)},
            "Kalloc_interval": {"lo": str(Kalloc.lo), "hi": str(Kalloc.hi)},
        },
        "pass": bool(passed),
    }

def main():
    if len(sys.argv) != 3:
        print("Usage: generate_tail_certificate.py constants.json tail_certificate.json", fi
        sys.exit(2)

    with open(sys.argv[1], "r", encoding="utf-8") as f:
        constants = json.load(f)

    out = {
        "certificate_version": "v29",
        "m_band": constants["m_band"],
        "eta": constants["eta"],
        "alpha": constants["alpha_worst"],
    }
    out.update(compute(constants, prec=90))

    with open(sys.argv[2], "w", encoding="utf-8") as f:
        json.dump(out, f, indent=2)

    print("[generate] wrote", sys.argv[2])
    print("[generate] PASS =", out["pass"])
    print("[generate] lhs_interval.hi =", out["lhs_interval"]["hi"])
    print("[generate] rhs_interval.lo =", out["rhs_interval"]["lo"])

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

File: verify_tail_certificate.py

```
#!/usr/bin/env python3
"""
verify_tail_certificate.py  (v29)

Verifies that:
  (1) tail_certificate.json is exactly the output produced by
      generate_tail_certificate.py from the given constants.json,
  (2) the tail inequality holds in certified interval form:
        lhs_interval.hi < rhs_interval.lo

Usage:
  python3 verify_tail_certificate.py constants.json tail_certificate.json
"""
```

```
import json
import sys
from decimal import Decimal

import subprocess
import tempfile
import os

def dec(s: str) -> Decimal:
    return Decimal(s)

def same_interval(a, b) -> bool:
    return a["lo"] == b["lo"] and a["hi"] == b["hi"]

def main():
    if len(sys.argv) != 3:
        print("Usage: verify_tail_certificate.py constants.json tail_certificate.json", file
        sys.exit(2)

    const_path = sys.argv[1]
    cert_path  = sys.argv[2]

    with open(const_path, "r", encoding="utf-8") as f:
        constants = json.load(f)
    with open(cert_path, "r", encoding="utf-8") as f:
        cert = json.load(f)

    # Re-generate in a temp file and compare byte-for-byte canonical JSON.
    with tempfile.TemporaryDirectory() as td:
        regen_path = os.path.join(td, "regen.json")
        gen_cmd = [sys.executable, os.path.join(os.path.dirname(__file__), "generate_tail_ce
        subprocess.check_call(gen_cmd)

        with open(regen_path, "r", encoding="utf-8") as f:
            regen = json.load(f)

    # Compare key fields; we avoid strict file equality because JSON formatting can vary.
    keys = [
        "certificate_version","m_band","eta","alpha","prec","pi_interval","logm_interval","c
        "L_interval","lhs_interval","rhs_interval","derived_constants","pass"
    ]
    mism = []
    for k in keys:
        if regen.get(k) != cert.get(k):
            mism.append(k)

    lhs_hi = dec(cert["lhs_interval"]["hi"])
    rhs_lo = dec(cert["rhs_interval"]["lo"])
    passed = (lhs_hi < rhs_lo)

    print("m_band =", cert["m_band"])
```

```
    print("eta    =", cert["eta"])
    print("alpha  =", cert["alpha"])
    print("LHS interval =", cert["lhs_interval"])
    print("RHS interval =", cert["rhs_interval"])
    print("Check: lhs.hi < rhs.lo  ==> ", lhs_hi, "<", rhs_lo, "=", passed)

    if mism:
        print("FAIL: certificate mismatch in keys:", ", ".join(mism))
        sys.exit(1)
    if not cert.get("pass", False):
        print("FAIL: certificate 'pass' field is false")
        sys.exit(1)
    if not passed:
        print("FAIL: inequality does not hold")
        sys.exit(1)

    print("PASS")

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

D.6 Supplementary tick generator audit script (not used)

The supplementary tick audit script (Appendix **??**) is provided as `tick_generator_audit.py` with SHA–256:

1670a46567adbb1b69679dc8bad242d25786c4d949a55aed96e6b9e68dc991ed.

Its full contents are embedded here:

```
#!/usr/bin/env python3
"""
tick_generator_audit.py  (supplementary; NOT used in the proof)

Deterministic prime-locked tick generator and an audit against "true" zeta zeros
computed internally by mpmath (zetazero). This avoids any network dependency.

WARNING:
  - This is a floating-point audit (mpmath), not a certified proof computation.
  - It is included only as supplementary numerics and does not feed into the RH proof.

Usage:
  python3 tick_generator_audit.py

Outputs:
  A small table of max/mean absolute error and max/mean relative error for j=2..J
  for a few cutoff constants C in X(t)=C*(log t)^(3/2).
"""

import math
from typing import List, Tuple

import mpmath as mp
```

```
mp.mp.dps = 80

def smooth_weight(y: mp.mpf) -> mp.mpf:
    # W(y)=exp(1-1/(1-y)) for 0<=y<1; W(1)=0; W(y)=0 for y>1
    if y <= 0:
        return mp.mpf(1)
    if y >= 1:
        return mp.mpf(0)
    return mp.e ** (1 - 1/(1 - y))

def primes_up_to(n: int) -> List[int]:
    if n < 2:
        return []
    sieve = bytearray(b"\x01")*(n+1)
    sieve[0:2] = b"\x00\x00"
    for p in range(2, int(n**0.5)+1):
        if sieve[p]:
            step = p
            start = p*p
            sieve[start:n+1:step] = b"\x00"*(((n-start)//step)+1)
    return [i for i in range(n+1) if sieve[i]]

def prime_powers_up_to(X: int) -> List[Tuple[int,int,int]]:
    # returns list of (p, k, p**k) with p prime, k>=1, p^k <= X
    ps = primes_up_to(X)
    out = []
    for p in ps:
        pk = p
        k = 1
        while pk <= X:
            out.append((p,k,pk))
            k += 1
            pk *= p
    return out

def theta(t: mp.mpf) -> mp.mpf:
    # Continuous Riemann{Siegel theta (mpmath handles branches)
    return mp.siegeltheta(t)


def X_of_t(t: mp.mpf, C: int) -> mp.mpf:
    return mp.mpf(C) * (mp.log(t) ** (mp.mpf(3)/2))

def P_X(t: mp.mpf, Delta: mp.mpf, C: int) -> mp.mpf:
    X = X_of_t(t, C)
    X_int = int(mp.floor(X))
    if X_int < 2:
        return mp.mpf(0)
    pp = prime_powers_up_to(X_int)
    total = mp.mpf(0)
```

```
    for p,k,pk_int in pp:
        pk = mp.mpf(pk_int)
        w = smooth_weight(pk / X)
        if w == 0:
            continue
        coef = (1/(k * mp.mpf(p)**(k/2))) * w
        arg1 = (t+Delta) * k * mp.log(p)
        arg0 = t * k * mp.log(p)
        total -= coef * (mp.sin(arg1) - mp.sin(arg0))
    return total

def F_j(tj: mp.mpf, Delta: mp.mpf, C: int) -> mp.mpf:
    return (theta(tj+Delta) - theta(tj)) + P_X(tj, Delta, C) - mp.pi

def next_tick(tj: mp.mpf, C: int, max_expand: int = 120) -> mp.mpf:
    lo = mp.mpf(0)
    flo = F_j(tj, lo, C)  # should be -pi
    hi = mp.mpf(5)
    fhi = F_j(tj, hi, C)
    expand = 0
    while fhi <= 0 and expand < max_expand:
        hi *= 2
        fhi = F_j(tj, hi, C)
        expand += 1
    if fhi <= 0:
        raise RuntimeError("Failed to bracket root; increase max_expand.")

    # bisection
    for _ in range(140):
        mid = (lo+hi)/2
        fmid = F_j(tj, mid, C)
        if fmid <= 0:
            lo = mid
        else:
            hi = mid
    return tj + hi

def true_zeros(J: int) -> List[mp.mpf]:
    # mpmath provides approximate zeros on the critical line.
    # This is NOT certified, but is deterministic at the chosen mp.dps.
    return [mp.im(mp.zetazero(j)) for j in range(1, J+1)]

def stats(errs: List[mp.mpf], truths: List[mp.mpf]) -> Tuple[mp.mpf, mp.mpf, mp.mpf, mp.mpf
    abs_err = [abs(e) for e in errs]
    rel_err = [abs(e)/abs(truths[i]) for i,e in enumerate(errs)]
    return max(abs_err), mp.fsum(abs_err)/len(abs_err), max(rel_err), mp.fsum(rel_err)/len(

def run_audit(C_values=(16,32,48), J=50):
    gammas = true_zeros(J)
    t1 = gammas[0]
    true_m = [2*g for g in gammas]
```

```
    print("Prime-locked tick generator audit (supplementary; floating point)")
    print("Truth zeros from mpmath.zetazero(j), mp.dps =", mp.mp.dps)
    print("Stats exclude j=1 (seed); errors computed over j=2..J.\n")

    print("{:>6s}  {:>14s}  {:>14s}  {:>14s}  {:>14s}".format("C","max|m|","mean|m|","max re
    for C in C_values:
        ticks = [t1]
        for j in range(1, J):
            ticks.append(next_tick(ticks[-1], C))
        tick_m = [2*t for t in ticks]
        errs = [tick_m[j]-true_m[j] for j in range(1, J)]
        truths = [true_m[j] for j in range(1, J)]
        mx, mean, mxr, meanr = stats(errs, truths)
        print("{:6d}  {:14.6e}  {:14.6e}  {:14.6e}  {:14.6e}".format(C, float(mx), float(mea

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

References

[1] R. R. Coifman, A. McIntosh, and Y. Meyer, L'intégrale de Cauchy définit un opérateur borné sur L^2 pour les courbes lipschitziennes, *Ann. of Math.* **116** (1982), 361–387.

[2] P. Duren, *Theory of H^p Spaces*, Academic Press, 1970.

[3] J. B. Garnett, *Bounded Analytic Functions*, Springer, 2007.

[4] A. Ivić, *The Riemann Zeta-Function*, Wiley, 1985.

[5] O. D. Kellogg, *Foundations of Potential Theory*, Dover, 1953.

[6] H. L. Montgomery and R. C. Vaughan, *Multiplicative Number Theory I: Classical Theory*, Cambridge Univ. Press, 2007.

[7] D. J. Platt, Isolating some nontrivial zeros of $\zeta(s)$, *Math. Comp.* **86** (2017), 2449–2467.

[8] D. J. Platt and T. S. Trudgian, The Riemann hypothesis is true up to $3 \cdot 10^{12}$, *Bull. Lond. Math. Soc.* **53** (2021), 792–797.

[9] E. C. Titchmarsh (rev. D. R. Heath–Brown), *The Theory of the Riemann Zeta-Function*, 2nd ed., Oxford Univ. Press, 1986.

[10] NIST Digital Library of Mathematical Functions. `https://dlmf.nist.gov/`

Authorship and AI–Use Disclosure

The author designed the framework and validated the mathematics and computations. Generative assistants were used for typesetting assistance, editorial organization, and consistency checks; they are not authors. All claims and certificates are the author's responsibility.