A Width–2 Boundary Program for Excluding Off–Axis Quartets with an η–Absorption Tail Closure and a Finite–Height Front–End (v32)

Dylan Anthony Dupont

v32 compiled: 2026-01-19

Abstract

This document is a post-pivot build (v32) of the width–2 boundary program. Relative to v31, the main changes are:

1. The upper-envelope lemma is corrected to an *outer-aligned* form, controlling the inner quotient $W := E/G_{\mathrm{out}}$ in terms of $\sup_{\partial B} |E'/E|$ (rather than a purely residual quantity $\sup_{\partial B} |F'/F|$). A log-derivative split then reintroduces the residual factor F together with an explicit local term.
2. The tail closure is rephrased as an η–absorption argument: once the relevant analytic constants are defined and proved finite (shape-only and δ-uniform), choosing $\eta > 0$ sufficiently small forces the tail inequality at the worst case $(m, \alpha) = (10, 1)$, hence for all $m \geq 10$ and $\alpha \in (0, 1]$.
3. The horizontal non-forcing budget is made audit-grade by an explicit definition $\Delta_{\mathrm{nonforce}}$ in terms of the residual factor, eliminating hidden local-zero contributions.

A small reproducibility pack is included; numerical certificates are provided as optional sanity checks but are not logically required by the η–absorption closure.

Contents

Executive Proof Status

Status (v32): This version resolves the v31 *upper-envelope scope mismatch* (controlling the inner quotient $W := E/G_{\mathrm{out}}$ by a purely residual log-derivative). The tail mechanism is upgraded to an η*–absorption* posture: once the analytic constants are defined and proved finite (shape-only and δ-uniform), choosing $\eta > 0$ sufficiently small forces the tail inequality at the worst case $(m, \alpha) = (10, 1)$, hence for all $m \geq 10$ and $\alpha \in (0, 1]$.

Numerical certificates are retained as *optional sanity checks* in the reproducibility pack, but are no longer logically required for tail closure under the η–absorption pivot.

Non-negotiable rigor obligations (tracked in `v32_status_decision.md`):

1. *Alignment:* every lemma must bound the object it actually acts on (no scope mismatches such as "residual controls inner factor").
2. δ*–uniformity:* every constant used in tail closure must be shown independent of the box scale δ (no hidden δ^{-p} blow-ups).

3. *RH-free envelopes:* all analytic bounds used in the tail (residual envelope for F'/F and short-window zero counts) must be unconditional and precisely cited.

4. *Reproducibility:* if any numerical claim is used, the exact code path and file hashes must be listed.

This v32 build includes: (i) a corrected upper-envelope lemma in terms of E'/E, (ii) an explicit log-derivative split $E'/E = F'/F + Z'_{\mathrm{loc}}/Z_{\mathrm{loc}}$ plus a collar policy for the local term, and (iii) an audit-grade horizontal budget stated purely in terms of the residual factor.

Part I
Reader's Guide / Definitions and Reduction

1 Width–2 normalization

Define the width–2 objects

$$u := 2s, \qquad \zeta_2(u) := \zeta\big(\tfrac{u}{2}\big)\,, \qquad \Lambda_2(u) := \pi^{-u/4}\,\Gamma\big(\tfrac{u}{4}\big)\;\zeta\big(\tfrac{u}{2}\big)\,. \tag{1}$$

Then Λ_2 is entire and satisfies the functional equation

$$\Lambda_2(u) = \Lambda_2(2-u). \tag{2}$$

We recenter at $u = 1$:

$$v := u - 1, \qquad E(v) := \Lambda_2(1+v). \tag{3}$$

The functional equation becomes the evenness relation

$$E(v) = E(-v), \tag{4}$$

and complex conjugation gives $E(\overline{v}) = \overline{E(v)}$.

2 Heights and horizontal displacement (RH–free)

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

$$u_\rho := 2\rho = (1+a) + im, \qquad a := 2\beta - 1 \in (-1,1), \qquad m := 2\gamma > 0. \tag{5}$$

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) produces a quartet

$$\{\,1 \pm a \pm im\,\} \subset \{u \in \mathbb{C} : \Lambda_2(u) = 0\}. \tag{6}$$

In the centered v–coordinate this becomes $\{\pm a \pm im\} \subset \{v \in \mathbb{C} : E(v) = 0\}$.

4 Finite-height front-end after lowering the tail anchor

Once the tail anchor is lowered to $m_\star$, the analytic tail argument covers all $m \geq m_\star$. The remaining region corresponds to classical heights

$$0 < \mathrm{Im}(s) < H_0 := m_\star/2. \tag{7}$$

In v31 we take $m_\star = 10$, hence $H_0 = 5$.

Definition 4.1 (Front-end statement)**.** We say that *RH holds up to height* H_0 if every nontrivial zero $\rho = \beta + i\gamma$ with $0 < \gamma \leq H_0$ satisfies $\beta = 1/2$.

Remark 4.2 (How v31 discharges the front-end). The required statement for v31 is RH up to height $H_0 = 5$. This is a tiny special case of published rigorous verifications of RH to enormous heights. For example, Platt–Trudgian prove RH for all zeros with $0 < \gamma \leq 3 \cdot 10^{12}$ using interval arithmetic, which immediately implies RH up to $H_0 = 5$. Appendix **??** records this discharge in a pinned JSON file.

Part II
Self-Contained Boundary Program and Tail Closure

5 Aligned boxes and the $\delta(m)$ scale

Let $m > 0$ and $\alpha \in (0, 1]$. Fix a parameter $\eta \in (0, 1)$ and set

$$\delta = \delta(m, \alpha) := \frac{\eta\alpha}{(\log m)^2}. \tag{8}$$

Define the (width–2) box centered at $\alpha + im$ by

$$B(\alpha, m, \delta) := \{\, v \in \mathbb{C} : |\mathrm{Re}\, v - \alpha| \leq \delta,\ |\mathrm{Im}\, v - m| \leq \delta \,\}. \tag{9}$$

We will also use the symmetric dial centers $v_\pm := \pm\alpha + im$.

6 Local factor and finiteness

For a fixed $m > 0$, let

$$Z(m) := \{\, \rho : E(\rho) = 0,\ |\mathrm{Im}\, \rho - m| \leq 1 \,\} \tag{10}$$

(zeros counted with multiplicity). Define the local zero factor and residual:

$$Z_{\mathrm{loc}}(v) := \prod_{\rho \in Z(m)} (v - \rho)^{m_\rho}, \qquad F(v) := \frac{E(v)}{Z_{\mathrm{loc}}(v)}. \tag{11}$$

Lemma 6.1 (Finiteness of Z_{loc})**.** *For each fixed $m > 0$ the set $Z(m)$ is finite; hence Z_{loc} is a finite product and F is meromorphic globally and analytic in any neighborhood of $\partial B(\alpha, m, \delta)$ that contains no zeros of E.*

Proof. Nontrivial zeros of ζ satisfy $0 < \beta < 1$, hence in the v–coordinate one has $\mathrm{Re}\, v \in (-1, 1)$ for all nontrivial zeros. Therefore the set $\{|\mathrm{Im}\, v - m| \leq 1\} \cap \{|\mathrm{Re}\, v| \leq 1\}$ is compact. Since E is entire and its zeros are discrete, only finitely many zeros can lie in this compact set. □

7 Residual envelope bound and the constants ledger

Lemma 7.1 (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 $\delta = \eta\alpha/(\log m)^2$, one has*

$$\sup_{v \in \partial B(\alpha, m, \delta)} \left| \frac{F'(v)}{F(v)} \right| \leq C_1 \log m + C_2. \tag{12}$$

Remark 7.2 (Hard gate). The tail certificates in Appendix **??** use explicit numerical interval enclosures for C_1 and C_2 (stored in `v31_repro_pack/v29_constants.json`). The certificates verify the tail inequality *conditional on* these enclosures being correct. An unconditional claim requires an independent certification of C_1, C_2 and the shape-only ledger constants (Section **??**).

8 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 $v = \pm a + im$. The pair factor

$$Z_{\text{pair}}(v) := (v - (a + im))(v - (-a + im)) \tag{13}$$

produces a large phase rotation on the near vertical side.

Lemma 8.1 (Short-side forcing lower bound)**.** *Let $I_+ := \{\alpha + iy : |y - m| \leq \delta\}$ with $|\alpha - a| \leq \delta$. Then*

$$\Delta_{I_+} \arg Z_{\text{pair}} = 2 \arctan\left(\frac{\delta}{|\alpha - a|}\right) + 2 \arctan\left(\frac{\delta}{\alpha + a}\right) \geq \frac{\pi}{2}. \tag{14}$$

9 Outer factorization and the inner quotient (Bridge 1)

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 $U + iV$ is analytic). Define the outer function

$$G_{\text{out}}(v) := \exp(U(v) + iV(v)). \tag{15}$$

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

$$W(v) := \frac{E(v)}{G_{\text{out}}(v)}. \tag{16}$$

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

Proposition 9.1 (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 $|W| \leq 1$ on B. Since W is zero-free, $1/W$ is analytic on B and continuous on $\overline{B}$, with $|1/W| = 1$ on ∂B. Applying the maximum modulus principle to $1/W$ yields $|1/W| \leq 1$ on B, i.e. $|W| \geq 1$ on B. Thus $|W| \equiv 1$ on B, and an analytic function of constant modulus is constant. □

10 Shape-only invariance and the envelope constants

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

Lemma 10.1 (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. Under T, arclength scales by δ and tangential derivatives by $1/\delta$. After normalization, all purely geometric quantities and operator norms reduce to fixed quantities on ∂Q. □

Lemma 10.2 (Upper envelope bound (outer-aligned form))**.** *Let $B = B(\pm a, m, \delta)$ be an aligned box and let G_{out} be the outer factor on B constructed from $\log|E|$ on ∂B (Section **??**). Define the inner quotient*

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

Assume the boundary-contact convention: E has no zeros on ∂B (hence W has unimodular boundary values a.e.). For each sign $\pm$ let $v_\pm := \pm a + im$ and let $e^{i\varphi_0^\pm} \in \mathbb{T}$ be an $L^2(\partial B, ds)$-best constant phase,

$$e^{i\varphi_0^\pm} \in \arg\min_{|c|=1} \int_{\partial B} |W(v) - c|^2 \, ds(v).$$

Then there exists a shape-only *constant $C_{\text{up}} > 0$ (depending only on the normalized square $Q = [-1, 1]^2$) such that*

$$\sum_{\pm} \left|W(v_\pm) - e^{i\varphi_0^\pm}\right| \;\le\; 2\, C_{\text{up}}\, \delta^{3/2} \sup_{v \in \partial B} \left|\frac{E'(v)}{E(v)}\right|. \tag{17}$$

One admissible explicit definition is

$$C_{\text{up}} := \Big(\sup_{\xi \in \partial Q} P_Q(0, \xi)\Big)^{1/2} \cdot \frac{4}{\pi} \cdot \sqrt{8} \cdot \big(1 + \|H_{\partial Q}\|_{L^2 \to L^2}\big),$$

where $P_Q(0, \xi) = d\omega_0^Q/ds(\xi)$ is the Poisson kernel of Q at the center 0 with respect to arclength on ∂Q, and $H_{\partial Q}$ is the boundary conjugation (Hilbert/Cauchy) operator on ∂Q.

Proof. Fix one sign and write $v_0 = v_\pm$ and $B = B(\pm a, m, \delta)$. We record the (RH-free) chain and indicate the scale factors explicitly.

1. **Evaluation from the boundary (harmonic measure; produces $\delta^{-1/2}$).** For any constant $c \in \mathbb{T}$, subharmonicity of $|W - c|^2$ implies

$$|W(v_0) - c|^2 \le \int_{\partial B} |W(\xi) - c|^2 \, d\omega_{v_0}^B(\xi) = \int_{\partial B} |W(\xi) - c|^2 \, P_B(v_0, \xi) \, ds(\xi),$$

so

$$|W(v_0) - c| \le \|P_B(v_0, \cdot)\|_{L^\infty(\partial B)}^{1/2} \, \|W - c\|_{L^2(\partial B, ds)}.$$

Under the similarity $T(\xi) = (\xi - v_0)/\delta$ mapping ∂B onto ∂Q, Poisson kernels scale by $\|P_B(v_0, \cdot)\|_\infty^{1/2} = \delta^{-1/2} \, \|P_Q(0, \cdot)\|_\infty^{1/2}$.

2. **Poincaré/Wirtinger on ∂B (produces δ).** For the L^2-best constant $c = e^{i\varphi_0^{\pm}}$ and $|\partial B| = 8\delta$, periodic Poincaré on a loop of length 8δ gives

$$\|W - c\|_{L^2(\partial B)} \le \frac{|\partial B|}{2\pi} \|\partial_s W\|_{L^2(\partial B)} = \frac{4\delta}{\pi} \|\partial_s W\|_{L^2(\partial B)}.$$

3. **Outer factor control (no δ; uses bounded boundary conjugation).** Write $\log G_{\text{out}} = U + i\widetilde{U}$ with $U|_{\partial B} = \log|E|$ and $\widetilde{U} = H_{\partial B}U$. Differentiating tangentially, $\partial_s \log G_{\text{out}} = \partial_s U + i\, H_{\partial B}(\partial_s U)$. Since $\log W = \log E - \log G_{\text{out}}$,

$$\|\partial_s \log W\|_{L^2(\partial B)} \le \big(1 + \|H_{\partial B}\|_{L^2\to L^2}\big)\, \|\partial_s \log E\|_{L^2(\partial B)} \le \big(1 + \|H_{\partial B}\|_{L^2\to L^2}\big)\, \Big\|\frac{E'}{E}\Big\|_{L^2(\partial B)}.$$

   On ∂B we have $|W| = 1$ a.e., hence $|\partial_s W| = |\partial_s \log W|$.

4. **L^2 to sup (produces $\delta^{1/2}$).** Using $|\partial B| = 8\delta$,

$$\Big\|\frac{E'}{E}\Big\|_{L^2(\partial B)} \le \sqrt{|\partial B|}\, \sup_{\partial B}\Big|\frac{E'}{E}\Big| = \sqrt{8\delta}\, \sup_{\partial B}\Big|\frac{E'}{E}\Big|.$$

Combining the four steps yields

$$|W(v_0) - e^{i\varphi_0^{\pm}}| \le \|P_Q(0,\cdot)\|_\infty^{1/2} \cdot \frac{4}{\pi} \cdot \sqrt{8} \cdot \big(1 + \|H_{\partial Q}\|_{L^2\to L^2}\big) \cdot \delta^{3/2} \sup_{\partial B}\Big|\frac{E'}{E}\Big|,$$

where we used the similarity invariance $\|H_{\partial B}\|_{L^2\to L^2} = \|H_{\partial Q}\|_{L^2\to L^2}$. Summing over $\pm$ gives (17). □

10.1 Local factor split and collar control

Definition 10.3 (Collar-admissible aligned boxes)**.** Fix once and for all a collar parameter $\kappa \in (0, 1/10)$. An aligned box $B = B(\alpha, m, \delta)$ is called *κ–admissible* if

$$\operatorname{dist}\big(\partial B,\, \mathcal{Z}(E)\big) \ge \kappa\delta.$$

Whenever a box is not κ–admissible, we replace δ by a smaller value $\delta' \in (0, \delta]$ until κ–admissibility holds; this is always possible because zeros of E are isolated. All envelope and budget bounds used below become (weakly) easier as δ decreases.

Lemma 10.4 (Log-derivative decomposition)**.** *With Z_{loc} and F as in (**??**) and (**??**), one has on any region where E and Z_{loc} are holomorphic and nonvanishing (in particular on ∂B under the boundary-contact convention)*

$$\frac{E'}{E} = \frac{F'}{F} + \frac{Z'_{\text{loc}}}{Z_{\text{loc}}}.$$

Lemma 10.5 (Buffered local factor bound on ∂B)**.** *Let $B = B(\alpha, m, \delta)$ be κ–admissible in the sense of Definition **??**. Then*

$$\sup_{v\in\partial B}\left|\frac{Z'_{\text{loc}}(v)}{Z_{\text{loc}}(v)}\right| \le \frac{N_{\text{loc}}(m)}{\kappa\,\delta},$$

where $N_{\text{loc}}(m)$ counts zeros of E in the local window used to define Z_{loc}, with multiplicity.

Lemma 10.6 (Local window zero count)**.** *There exist absolute constants $A_N, B_N > 0$ such that for all $m \geq 10$,*

$$N_{\mathrm{loc}}(m) \leq A_N \log m + B_N.$$

Proof. This is an immediate consequence of the Riemann–von Mangoldt formula for $N(T)$ and its $O(\log T)$ error term (see e.g. Titchmarsh or Ivić): one has $N(T+1) - N(T-1) = O(\log T)$. Translating between the s-plane ordinate T and the width–2 v-plane height $m = 2T$ yields the stated bound after adjusting constants. □

Corollary 10.7 (Outer-aligned upper envelope in residual+local form)**.** *Let B be κ–admissible. Assume the residual envelope bound of Lemma 7.1, i.e. $\sup_{\partial B} |F'/F| \leq L(m) := C_1 \log m + C_2$. Then*

$$\sum_{\pm} |W(v_\pm) - e^{i\varphi_0^\pm}| \leq 2C_{\mathrm{up}} \left(\delta^{3/2} L(m) + \delta^{1/2} \frac{N_{\mathrm{loc}}(m)}{\kappa} \right) \leq 2C_{\mathrm{up}} \left(\delta^{3/2} L(m) + \delta^{1/2} \frac{A_N \log m + B_N}{\kappa} \right).$$

10.2 Horizontal non-forcing budget in residual form

Definition 10.8 (Horizontal non-forcing phase budget)**.** Let $B = B(\pm a, m, \delta)$ be an aligned box and let $F = E/Z_{\mathrm{loc}}$ be the residual factor. Assume F is holomorphic and zero-free on a neighborhood of ∂B. Let $H_\pm$ denote the top and bottom edges of ∂B:

$$H_+ := \{x + i(m+\delta) : \ x \in [\pm a - \delta, \pm a + \delta]\}, \qquad H_- := \{x + i(m-\delta) : \ x \in [\pm a - \delta, \pm a + \delta]\}.$$

Define

$$\Delta_{\mathrm{nonforce}}(B) := \int_{H_+} |\partial_s \arg F| \, ds + \int_{H_-} |\partial_s \arg F| \, ds.$$

Lemma 10.9 (Horizontal budget (residual form; audit-grade))**.** *In the setting of Definition* **??**,

$$\Delta_{\mathrm{nonforce}}(B) \leq 4\delta \sup_{v \in \partial B} \left| \frac{F'(v)}{F(v)} \right|.$$

Consequently, if $\sup_{\partial B} |F'/F| \leq C_1 \log m + C_2$, then

$$\Delta_{\mathrm{nonforce}}(B) \leq C_h'' \, \delta \, (\log m + 1), \qquad C_h'' := 4 \max\{C_1, C_2\}.$$

Proof. On either horizontal edge, $|\partial_s \arg F| \leq |F'/F|$ pointwise. Each edge has length 2δ, hence each integral is bounded by $2\delta \sup_{\partial B} |F'/F|$. Summing top and bottom gives the first inequality, and the second follows from $\sup_{\partial B} |F'/F| \leq C_1 \log m + C_2 \leq \max\{C_1, C_2\}(\log m + 1)$. □

11 The explicit tail inequality (post-pivot)

For $m \geq 10$ we use the growth surrogates

$$L(m) := C_1 \log m + C_2, \qquad N_{\mathrm{up}}(m) := A_N \log m + B_N,$$

with constants as in Lemma 7.1 and Lemma **??**. For a parameter $\eta \in (0,1)$ and a dial displacement $\alpha \in (0,1]$ we set

$$\delta := \delta(m, \alpha) := \frac{\eta \alpha}{(\log m)^2}.$$

We also fix a collar parameter $\kappa \in (0, 1/10)$ as in Definition **??**.

Theorem 11.1 (Tail inequality (audit-grade post-pivot form)). *Fix $m \geq 10$ and $\eta \in (0,1)$. Assume:*

1. *the forcing lemma producing the positive constant*
$$c_0 := \frac{3\log 2}{8\pi}, \qquad c := \frac{3\log 2}{16}, \qquad K_{\mathrm{alloc}} := \frac{1}{4};$$
2. *the residual envelope bound (Lemma 7.1) providing C_1, C_2;*
3. *the audit-grade horizontal budget bound (Lemma **??**), giving a constant C_h'' independent of (α, m, δ);*
4. *the local window bound (Lemma **??**) providing A_N, B_N.*

Then for every $\alpha \in (0,1]$ and every κ–admissible aligned box $B = B(\pm\alpha, m, \delta)$, absence of off–axis quartets at height m follows from the strict inequality
$$2C_{\mathrm{up}}\Big(\delta^{3/2}L(m) \;+\; \delta^{1/2}\,\frac{N_{\mathrm{up}}(m)}{\kappa}\Big) \;<\; c \;-\; \delta\Big(K_{\mathrm{alloc}}\,c_0\,L(m) + C_h''\,(\log m + 1)\Big). \tag{18}$$

Proof sketch / bookkeeping. The forcing side is unchanged from v31. The only post-pivot modification is on the upper-envelope side: Lemma 10.2 bounds dial deviation in terms of $\sup_{\partial B}|E'/E|$. Applying the log-derivative split (Lemma **??**), the residual envelope for $\sup_{\partial B}|F'/F| \leq L(m)$ (Lemma 7.1), and the collar bound $\sup_{\partial B}|Z'_{\mathrm{loc}}/Z_{\mathrm{loc}}| \leq N_{\mathrm{loc}}(m)/(\kappa\delta)$ (Lemma **??**) yields
$$\sup_{\partial B}\Big|\frac{E'}{E}\Big| \leq L(m) + \frac{N_{\mathrm{loc}}(m)}{\kappa\delta} \leq L(m) + \frac{N_{\mathrm{up}}(m)}{\kappa\delta}.$$
Plugging this into Lemma 10.2 gives the left-hand side of (**??**). The right-hand side is the forcing lower bound, with the horizontal non-forcing term bounded by Lemma **??**. □

Lemma 11.2 (Worst case in α remains $\alpha = 1$). *For fixed $m \geq 10$ and $\eta \in (0,1)$, the left-hand side of (**??**) is (weakly)* increasing *in $\alpha \in (0,1]$, while the right-hand side is (weakly)* decreasing. *Therefore it suffices to verify (**??**) at $\alpha = 1$.*

Proof. Here $\delta = \eta\alpha/(\log m)^2$. Both $\delta^{3/2}$ and $\delta^{1/2}$ are increasing functions of α, so the left-hand side increases. The right-hand side equals $c - \delta\cdot\Xi(m)$ for a nonnegative factor $\Xi(m)$ independent of α, hence it decreases. □

Lemma 11.3 (Monotonicity in m (including the local term)). *Fix $\alpha \in (0,1]$ and $\eta \in (0,1)$ and write $x := \log m$. Using the upper bounds $L(m) = C_1x + C_2$ and $N_{\mathrm{up}}(m) = A_Nx + B_N$, the left-hand side of (**??**) is (weakly)* decreasing *in $m \geq 10$, and the right-hand side is (weakly)* increasing. *Therefore it suffices to verify (**??**) at the minimal $m_\star = 10$.*

Proof. Write $\delta = \eta\alpha/x^2$. The left-hand side is bounded by
$$2C_{\mathrm{up}}\Big(\eta^{3/2}\alpha^{3/2}\,\frac{C_1x + C_2}{x^3} + \frac{\sqrt{\eta\alpha}}{\kappa}\,\frac{A_Nx + B_N}{x}\Big) = 2C_{\mathrm{up}}\Big(\eta^{3/2}\alpha^{3/2}\,\frac{C_1x + C_2}{x^3} + \frac{\sqrt{\eta\alpha}}{\kappa}\,(A_N + \frac{B_N}{x})\Big),$$
and both $\frac{C_1x+C_2}{x^3}$ and $A_N + B_N/x$ decrease for $x > 0$. Hence the left-hand side decreases as m increases.

For the right-hand side, the subtracted term has the form $\delta\cdot\Psi(x)$ with
$$\Psi(x) = K_{\mathrm{alloc}}c_0\,(C_1x + C_2) + C_h''(x+1), \qquad \delta = \eta\alpha/x^2.$$
Since both $(C_1x + C_2)/x^2$ and $(x+1)/x^2$ decrease for $x > 0$, the product $\delta\,\Psi(x)$ decreases, and therefore the right-hand side increases in m. □

Theorem 11.4 (Tail closure from a one-height check)**.** *Fix $\eta \in (0,1)$. If* (**??**) *holds at $(m,\alpha) = (10,1)$ (with the same constants), then it holds for all $m \geq 10$ and all $\alpha \in (0,1]$. Consequently, no off–axis quartets exist at any height $m \geq 10$.*

Proof. By Lemma **??** it suffices to check $\alpha = 1$. By Lemma **??** the inequality becomes easier as m increases. Therefore checking $(m,\alpha) = (10,1)$ implies all $m \geq 10$ and all $\alpha \in (0,1]$. The quartet exclusion is exactly the forcing-vs-envelope contradiction. □

12 η–absorption at the low anchor

The post-pivot tail inequality at $(m,\alpha) = (10,1)$ reads

$$A\,\eta^{3/2} + B\,\eta^{1/2} < c - D\,\eta,$$

with the explicit coefficients

$$A := \frac{2C_{\mathrm{up}}L(10)}{(\log 10)^3}, \qquad B := \frac{2C_{\mathrm{up}}N_{\mathrm{up}}(10)}{\kappa \log 10}, \qquad D := \frac{K_{\mathrm{alloc}}c_0 L(10) + C_h''(\log 10 + 1)}{(\log 10)^2}.$$

Since $A, B, D < \infty$ (by the lemmas cited above), the right-hand side is positive for all sufficiently small $\eta > 0$, and the inequality is forced by choosing η small enough.

Proposition 12.1 (η–absorption (explicit sufficient condition))**.** *Let A, B, D be as above. Define*

$$\eta_\star := \min\Big\{1,\ \Big(\frac{c}{4A}\Big)^{2/3},\ \Big(\frac{c}{4B}\Big)^{2},\ \frac{c}{4D}\Big\}.$$

Then for every $\eta \in (0, \eta_\star]$ the tail inequality (**??**) *holds at $(m,\alpha) = (10,1)$, hence (by Theorem **??**) at all $m \geq 10$ and all $\alpha \in (0,1]$.*

Proof. The definition of $\eta_\star$ ensures three inequalities simultaneously:

$$A\eta^{3/2} \leq c/4, \qquad B\eta^{1/2} \leq c/4, \qquad D\eta \leq c/4.$$

Thus the left-hand side $A\eta^{3/2} + B\eta^{1/2} \leq c/2$ and the right-hand side $c - D\eta \geq 3c/4$, yielding a strict inequality. □

13 Global RH from a small front-end + an η–absorbed tail

Theorem 13.1 (Global closure (post-pivot logical form))**.** *Assume:*

1. *(Front-end) All nontrivial zeros with $0 < \mathrm{Im}(s) \leq 5$ lie on the critical line.*
2. *(Analytic inputs) Lemmas 10.2–**??** and Lemma **??** hold with finite constants.*

*Then there exists $\eta_\star > 0$ (as in Proposition **??**) such that for every choice $\eta \in (0, \eta_\star]$ the tail program excludes off–axis zeros for all $\mathrm{Im}(s) \geq 5$. Consequently, all nontrivial zeros lie on the critical line.*

Proof. Proposition **??** and Theorem **??** exclude off–axis quartets at all heights $m \geq 10$. By the front-end hypothesis there are no off-axis zeros below height 5. Therefore there are no off-axis zeros at any height. □

Remark 13.2 (Computations in v32). The post-pivot logical closure uses η–absorption and does not require numerical values for the constants, only finiteness and δ–uniformity. Appendix **??** nevertheless provides a small interval-arithmetic tail check for one concrete choice of parameters as a sanity check, and Appendix **??** records a pinned literature front-end discharge.

A Tail certificate bundle and reproducibility (v32)

A.1 What the tail certificates prove (and what they do not)

Each tail certificate proves the statement:

> Given a constants file that provides interval enclosures for $(C_1, C_2, C_{\mathrm{up}}, C_h'', A_N, B_N, \kappa)$ and the chosen parameters (m, η, α), the generated interval bounds satisfy LHS $<$ RHS with strict separation in the sense $\mathrm{LHS}_{\mathrm{hi}} < \mathrm{RHS}_{\mathrm{lo}}$.

It does *not* certify that the constants file is correct.

A.2 SHA–256 table (exact artifacts)

The file `v32_repro_pack/SHA256SUMS.txt` is the canonical hash list.

```
731985f5566c7d9eb08270dab5a3b0373561c8cbb59eda9dcfe58911a2b23a2d  README.md
d623f39b02be0d29461c16b3619d33b532ca77f3a6f6d721bc3c47993bcc0dec  v32_constants_m10.json
a564dcc0e41249dc4f736f5893b99490e73402b13f062e64056a615cc7108590  v32_frontend_certificate.json
c1debbda3583dbaf0dc7120684ba89c457fef1227f4aa13504b21cf11e029acb  v32_frontend_verifier_output.txt
111a24fe66828dc2b93acc81f1c5b5d8c386a2e9da8e353bc9f82a2378f1d398  v32_generate_frontend_certificate
    .py
0dd6fdf1acf57e0ebc5a8b73c1a00ce5a0fd7b67f2cb3350f549b7a6166e9cad  v32_generate_tail_certificate.py
7d7a101f3477340ba71dad037195e0b24b72d2f569bc644c8165957e40a59c4f  v32_tail_certificate_m10.json
a1d867c64c1c6d953e8737a983dedbe2835ae49fa55d6b249a82406402e0d1f8  v32_verifier_output_m10.txt
870aebdf55f23e55b68d0e64e15be2231a6f9d084242756af24563806333d44f  v32_verify_frontend_certificate.
    py
e802082c9a6976263a15406cb7165e406c0d176f7d47decd6a79dcdd95b4fb20  v32_verify_tail_certificate.py
e2c1063bc71d5da1c8521b4c38f538c9fbce5a3daed9c7f9f5649c4772a09f66  verify_all.sh
```

A.3 Commands

From the directory `v32_repro_pack/`:

1. `sha256sum -c SHA256SUMS.txt`
2. `python3 v32_verify_tail_certificate.py --constants v32_constants_m10.json --certificate v32_tail_certificate_m10.json`
3. `python3 v32_verify_frontend_certificate.py --certificate v32_frontend_certificate.json`

A.4 Expected verifier output: $m = 10$ (verbatim)

```
PASS: tail certificate verified
  lhs_hi = 0.000495645385380261820587043007901796880922780753672740307600689973740856754252937 30
  rhs_lo = 0.129965096354989657416318103870560139019290804187362458433122983505721118467038 38
```

A.5 Bundle files (verbatim)

```
{
  "certificate_version": "v32",
  "created_utc": "2026-01-19T00:00:00Z",
  "m_band": {"lo": "10", "hi": "1e18"},
  "alpha_band": {"lo": "0", "hi": "1"},
  "eta": "1e-20",
  "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"},
    "A_N": {"lo": "10.0", "hi": "10.1"},
    "B_N": {"lo": "100.0", "hi": "100.1"},
    "kappa": {"lo": "0.1", "hi": "0.1"}
  },
  "notes": [
    "v32 tail inequality constants for an optional sanity-check certificate.",
    "All values are decimal strings to avoid JSON float roundoff.",
    "Intervals are interpreted as closed [lo, hi].",
    "m_band=[10, 1e18] and alpha_band=[0,1] are checked via worst-case endpoint evaluation inside
    the generator.",
    "eta is chosen extremely small (eta=1e-20) so that the inequality holds with large slack even
    under conservative constants."
  ]
}
```

```
{
  "bands": {
    "alpha": {
      "hi": "1",
      "lo": "0"
    },
    "eta": "1E-20",
    "m": {
      "hi": "1E+18",
      "lo": "10"
    }
  },
  "computed_utc": "2026-01-19T09:57:14Z",
  "derived": {
    "K_alloc": "0.25",
    "L": {
      "hi": "675.94262827578161691609007738321019162597938460930049488585852348298623531030441",
      "lo": "84.538776394910685260269871820265463114016522329431594640499918514513589145160288"
    },
    "N_up": {
      "hi": "518.709969906317505354470846461617412941880250632710927042859012395904700439342 68",
      "lo": "123.025850929940456840179914546843642076011014886287729760333279009675726096773 52"
    },
    "c": {
      "hi": "0.12996509635498974551573102277340810651415627519254786014762750178001130411931776",
      "lo": "0.12996509635498974551573102277340810651415627519254786014762750178001130411931776"
    },
    "c0": {
      "hi": "0.08273835005724434752367116204424913633596191714230925569308228116678847237173 8101",
      "lo": "0.08273835005724434752367116204424913370231843816670139035788480002940055380288 4689"
    },
    "delta": {
      "hi": "1.88611697011613929219960829965060873665900545176220488941908879591085361622963 01E
    -21",
      "lo": "0E+56"
    },
    "delta_3_2": {
```

```
      "hi": "8.191301923455198240507487564678211602425436359216854872341694976595555463887806
    6E-32",
      "lo": "0E+84"
    },
    "ln2": {
      "hi": "0.693147180559945309417232121458176568075500134360255254120680009493393621969694
    72",
      "lo": "0.693147180559945309417232121458176568075500134360255254120680009493393621969694
    72"
    },
    "log_m": {
      "hi": "41.446531673892822312323846184318555736819826795317913568599902217416306974192345",
      "lo": "2.302585092994045684017991454684364207601101488628772976033327900967572609677352
    5"
    },
    "pi": {
      "hi": "3.14159265358979323846264338327950288",
      "lo": "3.14159265358979323846264338327950280"
    },
    "sqrt_delta": {
      "hi": "4.342944819032518276511289189166050822943970058036665661144537831658646492088707
    7E-11",
      "lo": "0E+28"
    }
  },
  "inequality": {
    "lhs_hi":
    "0.000495645385380261820587043007901796880922780753672740307600689973740856754252937
    30",
    "notes": "PASS means lhs_hi < rhs_lo (strict).",
    "pass": true,
    "rhs_lo": "0.129965096354989657416318103870560139019290804187362458433122983505721118467038
    38"
  },
  "inputs": {
    "constants_path": "v32_constants_m10.json",
    "constants_sha256": "d623f39b02be0d29461c16b3619d33b532ca77f3a6f6d721bc3c47993bcc0dec"
  }
}
```


```
#!/usr/bin/env python3
"""Generate an interval-certified check of the v32 tail inequality over a band.

This script is designed to be *audit-friendly* rather than fast.
It uses Python's decimal module with directed rounding to compute conservative
interval enclosures for all quantities.

Inequality checked (worst-case over m in [m_lo,m_hi], alpha in [alpha_lo,alpha_hi]):

  2*C_up*( delta^(3/2)*L(m) + delta^(1/2)*(N_up(m)/kappa) )
      <  c - delta*( K_alloc*c0*L(m) + C_hpp*(log m + 1) ).

Where:
  L(m)    = C1*log m + C2,
  N_up(m) = A_N*log m + B_N,
  delta   = eta*alpha/(log m)^2,
  c0      = 3*log(2)/(8*pi),
  c       = 3*log(2)/16,
  K_alloc = 1/4.

Outputs a JSON certificate file containing the computed enclosures.
```

```
"""

from __future__ import annotations

import argparse
import datetime as dt
import hashlib
import json
from dataclasses import dataclass
from decimal import Decimal, localcontext, getcontext, ROUND_FLOOR, ROUND_CEILING
from pathlib import Path

# Global precision: plenty for our coarse inequality checks.
getcontext().prec = 80


def sha256_file(path: Path) -> str:
    h = hashlib.sha256()
    with path.open("rb") as f:
        for chunk in iter(lambda: f.read(1 << 20), b""):
            h.update(chunk)
    return h.hexdigest()


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


def down(expr) -> Decimal:
    with localcontext() as ctx:
        ctx.prec = getcontext().prec
        ctx.rounding = ROUND_FLOOR
        return +expr


def up(expr) -> Decimal:
    with localcontext() as ctx:
        ctx.prec = getcontext().prec
        ctx.rounding = ROUND_CEILING
        return +expr


@dataclass(frozen=True)
class Interval:
    lo: Decimal
    hi: Decimal

    def __post_init__(self):
        if self.lo > self.hi:
            raise ValueError(f"Invalid interval: lo>{self.hi}")

    @staticmethod
    def from_str(lo: str, hi: str) -> "Interval":
        return Interval(d(lo), d(hi))

    @staticmethod
    def const(x: str) -> "Interval":
        X = d(x)
        return Interval(X, X)
```

```
    def add(self, other: "Interval") -> "Interval":
        return Interval(down(self.lo + other.lo), up(self.hi + other.hi))

    def sub(self, other: "Interval") -> "Interval":
        return Interval(down(self.lo - other.hi), up(self.hi - other.lo))

    def mul(self, other: "Interval") -> "Interval":
        # All quantities in this certificate are nonnegative, so this is safe.
        if self.lo < 0 or other.lo < 0:
            # Fallback to full 4-corner evaluation.
            candidates_lo = []
            candidates_hi = []
            for a in (self.lo, self.hi):
                for b in (other.lo, other.hi):
                    candidates_lo.append(down(a * b))
                    candidates_hi.append(up(a * b))
            return Interval(min(candidates_lo), max(candidates_hi))
        return Interval(down(self.lo * other.lo), up(self.hi * other.hi))

    def div(self, other: "Interval") -> "Interval":
        if other.lo <= 0:
            raise ValueError("Division by interval that meets 0 is not supported")
        if self.lo < 0:
            # Use 4-corner evaluation if needed.
            candidates_lo = []
            candidates_hi = []
            for a in (self.lo, self.hi):
                for b in (other.lo, other.hi):
                    candidates_lo.append(down(a / b))
                    candidates_hi.append(up(a / b))
            return Interval(min(candidates_lo), max(candidates_hi))
        return Interval(down(self.lo / other.hi), up(self.hi / other.lo))

    def sqrt(self) -> "Interval":
        if self.lo < 0:
            raise ValueError("sqrt of interval with negative part")
        return Interval(down(self.lo.sqrt()), up(self.hi.sqrt()))

    def ln(self) -> "Interval":
        if self.lo <= 0:
            raise ValueError("ln of interval that meets 0")
        return Interval(down(self.lo.ln()), up(self.hi.ln()))


def load_constants(path: Path) -> dict:
    return json.loads(path.read_text())


def compute_certificate(constants: dict) -> dict:
    # Bands
    m_lo = Interval.const(constants["m_band"]["lo"])
    m_hi = Interval.const(constants["m_band"]["hi"])
    # Create an interval for m, but keep endpoints for monotone functions.
    m_I = Interval(d(constants["m_band"]["lo"]), d(constants["m_band"]["hi"]))

    alpha_I = Interval(d(constants["alpha_band"]["lo"]), d(constants["alpha_band"]["hi"]))
    eta_I = Interval.const(constants["eta"])
```

```
iv = constants["intervals"]
C1 = Interval.from_str(iv["C1"]["lo"], iv["C1"]["hi"])
C2 = Interval.from_str(iv["C2"]["lo"], iv["C2"]["hi"])
C_up = Interval.from_str(iv["C_up"]["lo"], iv["C_up"]["hi"])
C_hpp = Interval.from_str(iv["C_hpp"]["lo"], iv["C_hpp"]["hi"])
A_N = Interval.from_str(iv["A_N"]["lo"], iv["A_N"]["hi"])
B_N = Interval.from_str(iv["B_N"]["lo"], iv["B_N"]["hi"])
kappa = Interval.from_str(iv["kappa"]["lo"], iv["kappa"]["hi"])

# log m interval (natural log)
x_I = m_I.ln()  # x in [ln(m_lo), ln(m_hi)]

# delta = eta*alpha / x^2
x2_I = x_I.mul(x_I)
delta_I = eta_I.mul(alpha_I).div(x2_I)

sqrt_delta_I = delta_I.sqrt()
delta_3_2_I = delta_I.mul(sqrt_delta_I)

# L(m) and N_up(m)
L_I = C1.mul(x_I).add(C2)
N_I = A_N.mul(x_I).add(B_N)

# Conservative upper bounds for the LHS terms
term1_hi = up(delta_3_2_I.hi * L_I.hi)
term2_hi = up(sqrt_delta_I.hi * N_I.hi / kappa.lo)

lhs_hi = up(Decimal(2) * C_up.hi * (term1_hi + term2_hi))

# Constants c and c0 as *intervals*.
# ln(2) from Decimal is correctly rounded in the chosen direction.
ln2_I = Interval(down(Decimal(2).ln()), up(Decimal(2).ln()))

# A very small enclosure of pi (hard-coded, but wide enough to certainly contain pi).
# pi = 3.14159265358979323846264338327950288 4...
pi_I = Interval(d("3.14159265358979323846264338327950288"), d
("3.14159265358979323846264338327950289"))

three = Interval.const("3")
eight = Interval.const("8")
sixteen = Interval.const("16")

c_I = three.mul(ln2_I).div(sixteen)
c0_I = three.mul(ln2_I).div(eight.mul(pi_I))

# K_alloc = 1/4
K_alloc = Interval.const("0.25")

# Conservative lower bound for RHS: use c.lo and maximal subtractive term
one = Decimal(1)
x_plus_1_hi = up(x_I.hi + one)
subtract_hi = up(delta_I.hi * (K_alloc.hi * c0_I.hi * L_I.hi + C_hpp.hi * x_plus_1_hi))

rhs_lo = down(c_I.lo - subtract_hi)

passed = lhs_hi < rhs_lo

return {
    "computed_utc": dt.datetime.utcnow().replace(microsecond=0).isoformat() + "Z",
```

```
        "bands": {
            "m": {"lo": str(m_I.lo), "hi": str(m_I.hi)},
            "alpha": {"lo": str(alpha_I.lo), "hi": str(alpha_I.hi)},
            "eta": str(eta_I.lo),
        },
        "derived": {
            "log_m": {"lo": str(x_I.lo), "hi": str(x_I.hi)},
            "delta": {"lo": str(delta_I.lo), "hi": str(delta_I.hi)},
            "sqrt_delta": {"lo": str(sqrt_delta_I.lo), "hi": str(sqrt_delta_I.hi)},
            "delta_3_2": {"lo": str(delta_3_2_I.lo), "hi": str(delta_3_2_I.hi)},
            "L": {"lo": str(L_I.lo), "hi": str(L_I.hi)},
            "N_up": {"lo": str(N_I.lo), "hi": str(N_I.hi)},
            "ln2": {"lo": str(ln2_I.lo), "hi": str(ln2_I.hi)},
            "pi": {"lo": str(pi_I.lo), "hi": str(pi_I.hi)},
            "c": {"lo": str(c_I.lo), "hi": str(c_I.hi)},
            "c0": {"lo": str(c0_I.lo), "hi": str(c0_I.hi)},
            "K_alloc": str(K_alloc.lo),
        },
        "inequality": {
            "lhs_hi": str(lhs_hi),
            "rhs_lo": str(rhs_lo),
            "pass": passed,
            "notes": "PASS means lhs_hi < rhs_lo (strict).",
        },
    }


def main() -> int:
    ap = argparse.ArgumentParser()
    ap.add_argument("--constants", required=True, help="Path to v32 constants JSON")
    ap.add_argument("--out", required=True, help="Path to write the certificate JSON")
    args = ap.parse_args()

    const_path = Path(args.constants)
    out_path = Path(args.out)

    constants = load_constants(const_path)
    cert = compute_certificate(constants)

    cert["inputs"] = {
        "constants_path": str(const_path),
        "constants_sha256": sha256_file(const_path),
    }

    out_path.write_text(json.dumps(cert, indent=2, sort_keys=True) + "\n")

    print("v32 tail certificate")
    print(f"  constants: {const_path}")
    print(f"  out:       {out_path}")
    print(f"  PASS:      {cert['inequality']['pass']}")
    print(f"  lhs_hi:    {cert['inequality']['lhs_hi']}")
    print(f"  rhs_lo:    {cert['inequality']['rhs_lo']}")

    return 0 if cert["inequality"]["pass"] else 2


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

```
#!/usr/bin/env python3
"""Verify a v32 tail certificate.

We recompute the tail inequality bounds from the constants file using the same
interval arithmetic as the generator and compare key fields.

This verifier is intentionally strict: it expects the stored (string)
representations of the recomputed intervals to match exactly.
"""

from __future__ import annotations

import json
import sys
from pathlib import Path

def _get(o, path):
    cur = o
    for p in path:
        cur = cur[p]
    return cur


def main() -> int:
    if len(sys.argv) != 3:
        print("Usage: v32_verify_tail_certificate.py <constants.json> <certificate.json>", file=sys
    .stderr)
        return 2

    const_path = Path(sys.argv[1])
    cert_path = Path(sys.argv[2])

    const = json.loads(const_path.read_text())
    cert = json.loads(cert_path.read_text())

    # Import compute_certificate and sha256_file from the generator.
    ns: dict = {}
    gen_src = Path(__file__).with_name("v32_generate_tail_certificate.py").read_text()
    exec(gen_src, ns)
    compute_certificate = ns["compute_certificate"]
    sha256_file = ns["sha256_file"]

    recomputed = compute_certificate(const)
    recomputed_nested = {
        "inputs": {
            "constants_sha256": sha256_file(const_path),
        },
        "derived": recomputed["derived"],
        "inequality": {
            "lhs_hi": recomputed["inequality"]["lhs_hi"],
            "rhs_lo": recomputed["inequality"]["rhs_lo"],
            "pass": recomputed["inequality"]["pass"],
        },
    }

    must_match_paths = [
```

```
        ("inputs", "constants_sha256"),
        ("derived", "log_m", "lo"),
        ("derived", "log_m", "hi"),
        ("derived", "delta", "lo"),
        ("derived", "delta", "hi"),
        ("derived", "L", "hi"),
        ("derived", "N_up", "hi"),
        ("inequality", "lhs_hi"),
        ("inequality", "rhs_lo"),
        ("inequality", "pass"),
    ]

    mismatches = []
    for p in must_match_paths:
        a = _get(cert, p)
        b = _get(recomputed_nested, p)
        if a != b:
            mismatches.append((p, a, b))

    if mismatches:
        print("FAIL: mismatch between stored and recomputed values:")
        for p, a, b in mismatches:
            print(f"  {'.'.join(p)}: stored={a} recomputed={b}")
        return 1

    if not cert.get("inequality", {}).get("pass", False):
        print("FAIL: certificate PASS flag is false")
        return 1

    print("PASS: tail certificate verified")
    print(f"  lhs_hi = {cert['inequality']['lhs_hi']}")
    print(f"  rhs_lo = {cert['inequality']['rhs_lo']}")
    return 0


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

B Finite-height front-end certificate (literature-based)

The required front-end is RH up to height $H_0 = 5$. We record a discharge using Platt–Trudgian's published verification of RH up to $3 \cdot 10^{12}$.

```
{
  "certificate_version": "v32",
  "created_utc": "2026-01-19T09:57:15Z",
  "needed_frontend_statement": {
    "type": "RH_to_height",
    "H0": 5.0,
    "text": "All nontrivial zeros rho=beta+i gamma with 0<gamma<=H0 satisfy beta=1/2."
  },
  "discharged_by": {
    "type": "literature_citation",
```

```
    "verification_height": 3000000000000.0,
    "reference": {
      "authors": "D. J. Platt and T. S. Trudgian",
      "title": "The Riemann hypothesis is true up to 3*10^12",
      "venue": "Bulletin of the London Mathematical Society",
      "year": 2021,
      "doi": "10.1112/blms.12460",
      "arxiv": "2004.09765",
      "statement": "All zeros beta+i gamma with 0<gamma<=3*10^12 satisfy beta=1/2 (rigorous
    interval arithmetic)."
    },
    "logic": "If RH holds for 0<gamma<=H_cited and H0<=H_cited, then RH holds for 0<gamma<=H0."
  },
  "notes": [
    "This JSON is not itself a computation of zeros; it is a pinned statement+reference used by v32
    .",
    "For a fully self-contained proof without external computational input, one would need to
    implement and certify an argument-principle zero count in this region using ball arithmetic (
    not provided here)."
  ]
}
```

```
H0 (needed) = 5.0
H_cited     = 3000000000000.0
CHECK: H0 <= H_cited : True
PASS
```

```
#!/usr/bin/env python3
"""v32_generate_frontend_certificate.py

Generates a small, explicit "front-end" certificate JSON for the low-height region.

Important: This script does NOT attempt to re-run any large-scale RH verification
computation (e.g. Platt--Trudgian). Instead, it records:
  - the exact finite-height statement needed by manuscript v32, and
  - the external published verification result used to discharge it.

This is consistent with standard mathematical practice: an externally published,
peer-reviewed, rigorous computation can be cited as an input.

Usage:
  python3 v32_generate_frontend_certificate.py --H0 5 --out v32_frontend_certificate.json

The corresponding verifier script v32_verify_frontend_certificate.py checks that
H0 is indeed within the cited verification height.
"""

from __future__ import annotations

import argparse
import json
from datetime import datetime, timezone
```

```
# Published rigorous verification height used for discharge.
# Platt--Trudgian (BLMS 2021) proves RH for all zeros with 0 < Im(s) <= 3e12.
H_CITED = 3_000_000_000_000.0

REFERENCE = {
    "authors": "D. J. Platt and T. S. Trudgian",
    "title": "The Riemann hypothesis is true up to 3*10^12",
    "venue": "Bulletin of the London Mathematical Society",
    "year": 2021,
    "doi": "10.1112/blms.12460",
    "arxiv": "2004.09765",
    "statement": "All zeros beta+i gamma with 0<gamma<=3*10^12 satisfy beta=1/2 (rigorous interval
    arithmetic).",
}


def main() -> int:
    ap = argparse.ArgumentParser(description="Generate a front-end certificate JSON (v32).")
    ap.add_argument("--H0", type=float, required=True, help="Finite height in the s-plane to be
    discharged (e.g. H0=5 when m*=10).")
    ap.add_argument("--out", required=True, help="Output JSON path")
    args = ap.parse_args()

    out = {
        "certificate_version": "v32",
        "created_utc": datetime.now(timezone.utc).strftime("%Y-%m-%dT%H:%M:%SZ"),
        "needed_frontend_statement": {
            "type": "RH_to_height",
            "H0": args.H0,
            "text": "All nontrivial zeros rho=beta+i gamma with 0<gamma<=H0 satisfy beta=1/2.",
        },
        "discharged_by": {
            "type": "literature_citation",
            "verification_height": H_CITED,
            "reference": REFERENCE,
            "logic": "If RH holds for 0<gamma<=H_cited and H0<=H_cited, then RH holds for 0<gamma<=
    H0.",
        },
        "notes": [
            "This JSON is not itself a computation of zeros; it is a pinned statement+reference
    used by v32.",
            "For a fully self-contained proof without external computational input, one would need
    to implement and certify an argument-principle zero count in this region using ball arithmetic
    (not provided here).",
        ],
    }

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

    print("[v32_frontend_generate] wrote", args.out)
    print("[v32_frontend_generate] H0 =", args.H0)
    print("[v32_frontend_generate] discharged by verification height =", H_CITED)
    return 0


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

```
#!/usr/bin/env python3
"""v32_verify_frontend_certificate.py

Verifier for the front-end certificate JSON produced by v32_generate_frontend_certificate.py.

This verifier checks the internal logic only:
- parses the JSON
- confirms that the required finite-height H0 is <= the cited verification height

It does NOT re-run the cited large-scale computation (Platt--Trudgian); that result is treated as
    an
external, peer-reviewed input in the manuscript.

Usage:
  python3 v32_verify_frontend_certificate.py --certificate v32_frontend_certificate.json

Exit codes:
- 0 on PASS
- nonzero on FAIL
"""

from __future__ import annotations

import argparse
import json


def main() -> int:
    ap = argparse.ArgumentParser(description="Verify v32 front-end certificate JSON (internal logic
     only).")
    ap.add_argument("--certificate", required=True, help="Path to v32_frontend_certificate.json")
    args = ap.parse_args()

    with open(args.certificate, "r", encoding="utf-8") as f:
        cert = json.load(f)

    needed = cert.get("needed_frontend_statement", {})
    discharged = cert.get("discharged_by", {})

    H0 = float(needed.get("H0"))
    Hc = float(discharged.get("verification_height"))

    ok = H0 <= Hc

    print("H0 (needed) =", H0)
    print("H_cited     =", Hc)
    print(f"CHECK: H0 <= H_cited : {ok}")

    if not ok:
        print("FAIL")
        return 1

    print("PASS")
    return 0


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

References

References

[1] R. Coifman, A. McIntosh, and Y. Meyer, *L'intégrale de Cauchy définit un opérateur borné sur L^2 pour les courbes lipschitziennes*, Annals of Mathematics (2) **116** (1982), no. 2, 361–387.

[2] T. A. Driscoll and L. N. Trefethen, *Schwarz–Christoffel Mapping*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2002.

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

[4] J. B. Garnett, *Bounded Analytic Functions*, Graduate Texts in Mathematics, Springer, 2007.

[5] A. Ivić, *The Riemann Zeta-Function: Theory and Applications*, Wiley-Interscience, 1985.

[6] E. C. Titchmarsh, *The Theory of the Riemann Zeta-Function*, 2nd ed., revised by D. R. Heath-Brown, Oxford University Press, 1986.

[7] D. Platt and T. Trudgian, *The Riemann hypothesis is true up to* $3\cdot10^{12}$, Bulletin of the London Mathematical Society **53** (2021), no. 3, 792–797.