hexagon:2610.00714v1

Every real primitive Dirichlet L-function has de Bruijn–Newman constant at most 9/32: a kernel-checked proof in Lean 4

Peter W. Murphy

Version 1.0.0, 31 October 2026
DOI 10.5281/zenodo.23840082
Repository https://github.com/DrMurphyIsIn/real-dirichlet-dbn-lean
The text of this paper is released under CC-BY-4.0; the Lean code under Apache-2.0.

Abstract

Let χ be a real primitive Dirichlet character of conductor $N \geq 3$ and parity $k \in \{0,1\}$, let $\Lambda(s,\chi) = (N/\pi)^{(s+k)/2}\Gamma(\frac{s+k}{2})L(s,\chi)$ be its completed L-function, and let

$$H_t^\chi(z) \;=\; \int_0^\infty e^{tu^2}\,\Phi_\chi(u)\cos(zu)\,du$$

be the de Bruijn–Newman heat flow of $\Lambda(\cdot,\chi)$, normalised so that $H_0^\chi(z) = \frac{1}{8}\Lambda(\frac{1}{2}+\frac{iz}{2},\chi)$. We give a proof, checked by the Lean 4 kernel, that for every such χ and every $t \geq 9/32$ all zeros of H_t^χ are real; equivalently, the generalised de Bruijn–Newman constant satisfies $\Lambda_\chi \leq 9/32$.

The mathematics is not new, and we say exactly whose it is. De Bruijn's strip-contraction theorem (1950) gives $\Lambda_\chi \leq D^2/2$ whenever the zeros of H_0^χ lie in $|\operatorname{Im} z| \leq D$. Freire, González-Riquelme, Muñoz, Poblete and Salinas observed (arXiv:2610.27343, Remarks 3.1 and 3.4) that the zero-free half-plane $\operatorname{Re} s > 7/8$ for Dirichlet L-functions asserted by OpenAI would give $D = 3/4$, and hence $\Lambda_\chi \leq 9/32$. What this paper adds is a complete formal proof of that composition. For conductors up to 400, FGMPS's unconditional bounds are stronger for 136 of the 242 characters; what is new is that 9/32, for every conductor, is a kernel-checked theorem. OpenAI's half-plane enters as their own kernel-checked Lean theorem. Every other step is proved in Lean on top of Mathlib: de Bruijn's theorem for a general class of kernels, the identification of the theta integral with Mathlib's completed L-function, and Gauss's sign theorem for real primitive characters. The final theorem depends only on the axioms `propext`, `Classical.choice` and `Quot.sound`. In local, unsandboxed replays five checkers accept it (Lean's kernel, run twice, `nanoda`, and the `con-leche`/`con-ron` pair), and all five reject a deliberately unsound control; and a sandboxed Comparator run against a separately written challenge accepts it (made by another agent session; the auditor reproduced its solution export byte for byte).

The result rests on OpenAI's Lean proof, whose accompanying manuscript is unrefereed. It is not progress on the Generalised Riemann Hypothesis, which is the statement $\Lambda_\chi \leq 0$. A short, separable section records two companion results for ζ that use no OpenAI code: $\Lambda \leq 225/512$ and $\psi(x) = x + O_\varepsilon(x^{31/32+\varepsilon})$, both from Cureton and Thorner's half-plane $\operatorname{Re} s > 31/32$.

Contents

1 Introduction **2**
1.1 The de Bruijn–Newman constant 2

1.2 The Dirichlet analogue . . . 3
1.3 OpenAI's half-plane . . . 3
1.4 What this paper does . . . 4

2 The statement **5**
2.1 The definitions . . . 5
2.2 The theorem . . . 6
2.3 Mathlib's completed L-function, and the classical one . . . 7
2.4 Relation to the FGMPS normalisation . . . 8

3 The mathematics **8**
3.1 The theta kernel and the bridge . . . 8
3.2 Gauss's sign theorem . . . 9
3.3 De Bruijn's strip theorem . . . 10
3.4 From the half-plane 7/8 to the strip $D = 3/4$ . . . 11
3.5 An example: χ_{-4} . . . 12
3.6 What the proof does not use . . . 12

4 The formalization **12**
4.1 Architecture . . . 12
4.2 Two environments . . . 14
4.3 The trust base . . . 14
4.4 How to read and check it . . . 15

5 Verification **15**
5.1 Axioms . . . 15
5.2 Five checkers from three lineages, and a control they must reject . . . 16
5.3 Tampered statements . . . 17
5.4 Statement identity with and without OpenAI's code . . . 17
5.5 Independent numerics . . . 17
5.6 The sandboxed Comparator run . . . 18

6 Optional: companion results for ζ that use no OpenAI code **19**

7 Limitations **20**

8 Use of AI **21**

A The auditor's records **24**

1 Introduction

1.1 The de Bruijn–Newman constant

Let $\xi(s) = \frac{1}{2}s(s-1)\pi^{-s/2}\Gamma(s/2)\zeta(s)$ be Riemann's entire function. Following Rodgers and Tao [30] and Polymath15 [29], put

$$H_t(z) = \int_0^\infty e^{tu^2}\,\Phi(u)\cos(zu)\,du, \qquad \Phi(u) = \sum_{n\geq 1}\bigl(2\pi^2 n^4 e^{9u} - 3\pi n^2 e^{5u}\bigr)e^{-\pi n^2 e^{4u}}. \tag{1}$$

Then $H_0(z) = \frac{1}{8}\xi(\frac{1}{2} + \frac{iz}{2})$, so the Riemann Hypothesis says that every zero of H_0 is real. The family H_t solves the backward heat equation $\partial_t H_t = -\partial_z^2 H_t$. De Bruijn [10] showed that real zeros stay real as t increases and that every zero of H_t is real for $t \geq 1/2$. Newman [22] showed that some H_t has non-real zeros. Together these give a constant $\Lambda \in (-\infty, 1/2]$ such that every zero of H_t is real exactly when $t \geq \Lambda$. The Riemann Hypothesis is $\Lambda \leq 0$. Rodgers and Tao proved $\Lambda \geq 0$ [30]. On the other side, Ki, Kim and Lee made de Bruijn's bound strict [16], Polymath15 proved $\Lambda \leq 0.22$ [29], and Platt and Trudgian proved $\Lambda \leq 0.2$ [27].

1.2 The Dirichlet analogue

For a real primitive character χ of conductor $N \geq 3$ the same construction applies to $\Lambda(s, \chi)$. We follow the normalisation of Freire, González-Riquelme, Muñoz, Poblete and Salinas [13], cited below as FGMPS. Their heat flow is $H_t^L(z) = \int_{\mathbb{R}} e^{tu^2 + izu}\Phi_L(u)\,du$ with $H_0^L(z) = \xi_L(\frac{1+iz}{2})$, where ξ_L is our $\Lambda(\cdot, \chi)$ ([13, (1.2), (1.3), (2.6)]). Our H_t^χ equals $H_t^L/8$ (Section 2.4), so the two families have the same zeros at every time t.

FGMPS state that there is a finite constant Λ_χ (their Λ_L) such that every zero of H_t^χ is real if and only if $t \geq \Lambda_\chi$; they credit the existence of Λ_χ to Newman's work [22]. The Generalised Riemann Hypothesis for $L(s, \chi)$ is equivalent to $\Lambda_\chi \leq 0$. Dobner proved $\Lambda_\chi \geq 0$ for this family as part of a theorem for the whole extended Selberg class [12], and de Bruijn's argument gives $\Lambda_\chi \leq 1/2$ from the critical strip. For the lower-bound side, FGMPS also survey work of Stopple [31], of Andrade, Chang and Miller [1], and of Newman and Wu [23]; we have not read these papers and cite them only through FGMPS.

Before October 2026, the trivial bound $1/2$ was the only known upper bound [13, §1.4]. FGMPS then proved the first non-trivial ones. They extend Polymath15's method to Dirichlet L-functions, using Platt's verification of GRH [28] and interval arithmetic. They obtain the unconditional uniform bound

$$\Lambda_\chi \leq 953/3200 = 0.2978125 \qquad (3 \leq N \leq 400),$$

together with stronger individual bounds for each of the 242 real primitive characters in that range [13, Theorem 1.1, §7.7]. They also record a conditional statement, which is the one this paper formalizes. Remark 3.1 of [13] reads:

> "the zero-free half-plane $\operatorname{Re} s > 7/8$ asserted in [24, Theorem 1.1], together with the functional equation, would imply the initial strip with $D = 3/4$. The conclusions reported for this value of D are conditional on that global strip hypothesis."

Here [24] is OpenAI's preprint [25]. Remark 3.4 of [13] adds: "The initial strip hypothesis alone gives $\Lambda_L \leq D^2/2$ by Theorem 2.6". Their Theorem 2.6 is de Bruijn's strip contraction, "essentially contained in Theorem 13 of [9]", where [9] is [10]. At $D = 3/4$ this gives $\Lambda_\chi \leq 9/32$, for every conductor.

1.3 OpenAI's half-plane

On 30 September 2026 OpenAI released a preprint whose Theorem 1.1 asserts that every Dirichlet L-function, ζ included, has no zero in $\operatorname{Re} s > 7/8$, apart from the pole of a principal character at $s = 1$ [25]. A companion preprint of 5 October, "written with human assistance", isolates the weaker half-plane $\operatorname{Re} s > 11/12$ as an intermediate step and proves it separately [24]; FGMPS note that it would give $D = 5/6$ [13, Remark 3.1]. The release includes a Lean 4 library [26] with the theorem

```
theorem OAI.DirichletCharacter.LFunction_ne_zero_of_seven_eighths_lt_re
    {q : ℕ} [NeZero q] (χ : DirichletCharacter ℂ q) {s : ℂ}
```

```
    (hs : (7 / 8 : ℝ) < s.re) (hpole : ¬ (χ = 1 ∧ s = 1)) :
    _root_.DirichletCharacter.LFunction χ s ≠ 0
```

The statement is copied from OpenAI's challenge file at commit `adc7f124`,

`lean/ComparatorChallenges/DirichletSevenEighths.lean`,

where its proof is `sorry`. The proof itself is in `lean/OAI/NumberTheory/DirichletL/Nonvanishing.lean`. The statement is about Mathlib's own `DirichletCharacter.LFunction`, holds for every modulus and every character, and has no constants. Neither manuscript has been refereed. The Lean proof was replayed by the kernel in this programme on 7 October 2026; Section 4.3 says exactly what that trust consists of.

1.4 What this paper does

Theorem 1.1 (main theorem; Lean: `bb_real_primitive_meaning_oai`)**.** *Let χ be a primitive Dirichlet character modulo N, with $\chi \neq 1$ and χ real-valued, and let $k \in \{0,1\}$ be its parity. Then:*

(a) for every real $t \geq 9/32$, every zero of H_t^χ is real;

(b) for every $z \in \mathbb{C}$, with $s = \frac{1}{2} + \frac{iz}{2}$,

$$H_0^\chi(z) = \tfrac{1}{8}\, N^{(s+k)/2}\, \Lambda_{\mathrm{M}}(s,\chi) = \tfrac{1}{8}\, \Lambda(s,\chi),$$

where $\Lambda_{\mathrm{M}}(s,\chi)$ is Mathlib's `DirichletCharacter.completedLFunction`, the entire function equal to $\Gamma_{\mathbb{R}}(s+k)L(s,\chi)$ away from the poles of $\Gamma_{\mathbb{R}}(s+k)$, and $\Gamma_{\mathbb{R}}(s) = \pi^{-s/2}\Gamma(s/2)$.

Consequently $\Lambda_\chi \leq 9/32$.

Part (a) is the bound. Part (b) is there so that a reader need check only the outer shape of our definition: that $H_t^\chi(z) = \int_0^\infty e^{tu^2} F(u)\cos(zu)\,du$ for one function F, independent of t and z. Part (b) proves that at $t = 0$ this is $\frac{1}{8}$ of the completed L-function of χ, as Mathlib defines it. Since the identity holds for all $z \in \mathbb{C}$ (at $z = iy$ with $y < -1$ it forces $F\cosh(yu)$ to be integrable) and the cosine transform determines F, F is the classical Φ_χ, whatever the formula in the definition says. Section 2 gives the Lean statement and a guide to reading it line by line.

What is ours, and what is not. The theorem is FGMPS's conditional observation with its hypothesis discharged. The steps and their sources are:

Step	Mathematics due to	In Lean
Zero-free half-plane $\mathrm{Re}\, s > 7/8$ for $L(s,\chi)$	OpenAI [25]	OpenAI [26]
Half-plane plus functional equation give $D = 3/4$	FGMPS, Rem. 3.1 [13] (classical)	this work
Strip D gives $\Lambda_\chi \leq D^2/2$	de Bruijn [10]; FGMPS Thm. 2.6, Rem. 3.4	this work (generic)
Theta kernel, its Mellin transform, and the bridge to $\Lambda(s,\chi)$	classical	this work, on Mathlib's L-function API
Root number $\varepsilon(\chi) = 1$ for real primitive χ	Gauss [14]	this work
L-functions, Gauss sums, functional equation, Hurwitz kernels	classical	Mathlib [32]
de Bruijn's theorem for ζ, Hadamard products, Hurwitz's theorem	de Bruijn; classical	the `dbn` island [20]

The contribution of this paper is the kernel proof from end to end, together with three reusable formal components that, as far as we know, were not previously available in Lean:

(i) de Bruijn's strip-contraction theorem for an abstract even kernel with double-exponential decay of arbitrary rate (Section 3.3). It generalises the `dbn` island's theorem for ζ [20], with a Fourier-injectivity argument in place of the positivity of Φ;

(ii) the bridge from the Dirichlet theta kernel to Mathlib's `completedLFunction`, with the theta functional equation for real primitive characters (Section 3.1);

(iii) Gauss's sign theorem, $\varepsilon(\chi) = 1$ for every real primitive χ, in Mathlib's normalisation, including the closed form of the quadratic Gauss sum for every $N \geq 1$ (Section 3.2). At Mathlib revision `0493595e` there is no sign lemma for `rootNumber` [3, NOTE_D2].

What this paper does not do. There is no new analytic input. The bound 9/32 is the number FGMPS already wrote down conditionally. For $3 \leq N \leq 400$ it is not the best available: FGMPS's unconditional individual bounds beat 9/32 for 136 of the 242 characters in that range [13, §7.7, Table 8]. One example is χ_{-4}, for which they prove $\Lambda_{\chi_{-4}} \leq 0.26$ [13, §7.5]. Their bounds also do not depend on OpenAI. Our bound is uniform in N, but it is only as good as OpenAI's theorem, whose manuscript is unrefereed. Granting OpenAI's half-plane, FGMPS's own $D = 3/4$ search improves on 9/32 for 149 of the 242 characters [13, Table 8], and their Remark 3.4 gives 9/32 for the rest and for every larger conductor. So no bound in this paper is better than what FGMPS's paper gives once OpenAI's theorem is granted; what is new is that the bound 9/32, for every conductor, is a kernel-checked theorem. Nothing here bears on GRH, which is $\Lambda_\chi \leq 0$. We do not formalize Dobner's lower bound $\Lambda_\chi \geq 0$, nor the existence of Λ_χ as an infimum (Remark 2.1).

How to read this paper. Section 2 states the theorem formally and explains how to read it. Section 3 gives the mathematics in full, for a number theorist who does not read Lean. Section 4 describes the formalization and its trust base, Section 5 the checks it passed, and Section 6 the optional companion results for ζ. Section 7 lists the limitations and Section 8 states how AI was used.

2 The statement

2.1 The definitions

Four non-Mathlib definitions appear in the statement. Here they are, excerpted verbatim from the challenge file `ChallengeDefs.lean` (md5 `67e538bc`; only its `namespace` and `open` lines and the docstring of `Hchi` are omitted), which copies them from the audited sources `GenTheta.lean` (md5 `3dbf6dec`) and `GenThetaFE.lean` (md5 `8aeec9b0`):

```
variable (c : ℕ → ℝ) (k : ℕ) (q : ℝ)

/-- `θ_c(x) = Σ_n c(n) n^k exp(−π n² x / q)`. -/
noncomputable def thetaC (x : ℝ) : ℝ :=
  ∑' n : ℕ, c n * (n : ℝ) ^ k * Real.exp (-π * (n : ℝ) ^ 2 * x / q)

/-- `Φ_c(u) = e^{(1+2k)u} θ_c(e^{4u})`. -/
noncomputable def PhiC (u : ℝ) : ℝ :=
  Real.exp ((1 + 2 * k) * u) * thetaC c k q (Real.exp (4 * u))
```

```
variable {N : ℕ} [NeZero N] (χ : DirichletCharacter ℂ N)

variable {χ} {c : ℕ → ℝ} {k : ℕ}

open scoped Classical in
/-- The parity weight `k(χ) = 0` (even) / `1` (odd). -/
noncomputable def parityK (χ : DirichletCharacter ℂ N) : ℕ := if χ.Even then 0 else 1

open scoped Classical in
noncomputable def Hchi (χ : DirichletCharacter ℂ N) (t : ℝ) (z : ℂ) : ℂ :=
  ∫ u in Ioi (0 : ℝ), ((Real.exp (t * u ^ 2) : ℝ) : ℂ) *
    ((PhiC (fun n ↦ (χ n).re) (parityK χ) N u : ℝ) : ℂ) * Complex.cos (z * u)
```

In mathematical notation, with $c(n) = \operatorname{Re}\chi(n)$, $k = k(\chi)$ and $q = N$:

$$\theta_\chi(x) = \sum_{n\geq 0} \chi(n)\, n^k e^{-\pi n^2 x/N}, \qquad \Phi_\chi(u) = e^{(1+2k)u}\,\theta_\chi(e^{4u}),$$
$$H_t^\chi(z) = \int_0^\infty e^{tu^2}\Phi_\chi(u)\cos(zu)\,du. \tag{2}$$

Three points about these transcriptions:

- The sum is a Mathlib `tsum` over all $n \in \mathbb{N}$, including $n = 0$. The $n = 0$ term vanishes because $\chi(0) = 0$ when $N \neq 1$, and $\chi \neq 1$ forces $N \neq 1$. The proofs derive this internally.
- For real-valued χ, $c(n) = \operatorname{Re}\chi(n) = \chi(n)$. The definition takes the real part so that Φ_χ is real-valued by construction. It makes sense for every Dirichlet character, but it is only meaningful for real ones.
- The integral is a Bochner integral over $(0, \infty)$ for Lebesgue measure. `Complex.cos` is the entire cosine, so $H_t^\chi(z)$ is defined for complex z. Integrability for every real t and complex z is proved, from the decay estimate (4) below. A Bochner integral of a non-integrable function would silently be 0, so this proof matters.

2.2 The theorem

The theorem, as stated and proved in OpenAI's environment (file `BbOaiMeaning.lean`, md5 `23ea25e9`), is:

```
open Complex in
theorem bb_real_primitive_meaning_oai :
    (∀ {N : ℕ} [NeZero N] (χ : DirichletCharacter ℂ N), χ.IsPrimitive → χ ≠ 1 → (∀
    n : ℕ, (χ n).im = 0) →
      ∀ t : ℝ, 9 / 32 ≤ t → ∀ z : ℂ, DBN.Gen.Hchi χ t z = 0 → z.im = 0) ∧
    (∀ {N : ℕ} [NeZero N] (χ : DirichletCharacter ℂ N), χ.IsPrimitive → χ ≠ 1 → (∀
    n : ℕ, (χ n).im = 0) → ∀ z : ℂ,
      DBN.Gen.Hchi χ 0 z = (1 / 8) * (N : ℂ) ^ ((1 / 2 + I * z / 2 + DBN.Gen.parityK χ)
     / 2) *
        DirichletCharacter.completedLFunction χ (1 / 2 + I * z / 2)) :=
  ⟨bb_real_primitive_real_zeros_oai, fun χ hprim hχ1 hreal z ↦ DBN.Gen.Hchi_bridge χ
    hprim hχ1 hreal z⟩
```

Reading guide.

Line	Reading
1	Inside this declaration, `I` means the imaginary unit `Complex.I`.
2	The name of the theorem. The prefix `bb_` records that the auditor wrote this composition file.
3	For every modulus $N \geq 1$ (`NeZero N` says $N \neq 0$; the braces make N implicit) and every Dirichlet character χ modulo N with values in $\mathbb{C}$ that is primitive in Mathlib's sense (`IsPrimitive`: its conductor is N), is not the trivial character, and takes only real values at the natural numbers, ...
4	... and for every real $t \geq 9/32$ and every complex z with $H_t^\chi(z) = 0$, we have $\operatorname{Im} z = 0$. This is part (a) of Theorem 1.1. The numerals `9 / 32` are elaborated in $\mathbb{R}$; `#check` with `pp.numericTypes` prints `(9 / 32 : ℝ)`.
5	And, with the same hypotheses on χ, for every complex z:
6–7	$H_0^\chi(z) = \frac{1}{8} \cdot N^{(s+k)/2} \cdot \Lambda_{\mathrm{M}}(s,\chi)$, where $s = \frac{1}{2} + \frac{iz}{2}$, $k = k(\chi)$, and $N^{(\cdot)}$ is Mathlib's complex power with base the natural number N cast to $\mathbb{C}$. Here `completedLFunction` is Mathlib's, from `Mathlib.NumberTheory.LSeries.DirichletContinuation`. This is part (b).
8	The proof: the pair of the two theorems `bb_real_primitive_real_zeros_oai` and `Hchi_bridge`, described in Section 4.

Part (a) alone uses 43 constants (29 from Mathlib, 13 from `Init`, and `Hchi`); the whole conjunction uses 59 (37 from Mathlib, 20 from `Init`, and `Hchi` and `parityK` from `GenThetaFE`). Every one of them comes from Lean's core library (`Init`), from Mathlib, or from `GenThetaFE` (`Hchi` and `parityK`). None comes from OpenAI's `OAI.*` namespace. The auditor's program, which lists each constant of the statement together with the module that declares it, confirms this [3, REVIEW_GAUSS_SIGN_AND_OAI §B.3; ADD1 §3].

2.3 Mathlib's completed L-function, and the classical one

Mathlib's $\Lambda_{\mathrm{M}}(s,\chi) =$ `completedLFunction` χ `s` omits the conductor factor. Its relevant facts, read from Mathlib's source at revision `0493595e` [3, ADD1 §5], are:

- `gammaFactor` χ `s = if` χ`.Even then Gamma`$\mathbb{R}$ `s else Gamma`$\mathbb{R}$ `(s + 1)`, where $\Gamma_{\mathbb{R}}(s) = \pi^{-s/2}\Gamma(s/2)$;
- `LFunction_eq_completed_div_gammaFactor`: $L(s,\chi) = \Lambda_{\mathrm{M}}(s,\chi)/\Gamma_{\mathbb{R}}(s+k)$ when $s \neq 0$ or $N \neq 1$;
- `IsPrimitive.completedLFunction_one_sub`: for primitive χ,

$$\Lambda_{\mathrm{M}}(1-s,\chi) = N^{s-1/2}\,\varepsilon(\chi)\,\Lambda_{\mathrm{M}}(s,\chi^{-1}), \tag{3}$$

  with `rootNumber` $\chi = \varepsilon(\chi) = \tau(\chi)/(i^k N^{1/2})$, where $\tau(\chi) = \sum_{a \bmod N} \chi(a) e^{2\pi i a/N}$ (`gaussSum` χ `stdAddChar`).

Hence $N^{(s+k)/2}\Lambda_{\mathrm{M}}(s,\chi) = (N/\pi)^{(s+k)/2}\Gamma(\frac{s+k}{2})L(s,\chi) = \Lambda(s,\chi)$ is the classical completed L-function. It satisfies $\Lambda(1-s,\chi) = \varepsilon(\chi)\Lambda(s,\bar{\chi})$, because the powers of N in (3) combine. Since $N^{(s+k)/2} \neq 0$, part (b) says that the zeros of H_0^χ are exactly the points z with $\Lambda(\frac{1}{2} + \frac{iz}{2}, \chi) = 0$. These points are in bijection with the non-trivial zeros of $L(s,\chi)$. The identification of $N^{(s+k)/2}\Lambda_{\mathrm{M}}$ with the classical Λ is this reading of Mathlib's definitions; the Lean statement itself is about Mathlib's constant.

2.4 Relation to the FGMPS normalisation

In [13, (1.3)], $\Phi_L(u) = 4\varphi_L(e^{4u})$ with $\varphi_L(x) = x^{1/4+a/2}\psi_L(x)$ and $\psi_L(x) = \sum_{n\geq 1} n^a \chi(n) e^{-n^2\pi x/m}$. Here $m = N$ and $a = k$, and $x^{1/4+a/2}$ at $x = e^{4u}$ is $e^{(1+2k)u}$, so $\Phi_L = 4\Phi_\chi$. For even Φ_χ (Section 3.1), $H_t^L(z) = \int_{\mathbb{R}} e^{tu^2}\Phi_L(u)e^{izu}\,du = 8H_t^\chi(z)$ for every real t. So both families have the same zeros at every time, and the constant Λ_χ is the same in both conventions. This identity for $t \neq 0$ is a one-line unfolding of the definitions, not a Lean theorem: FGMPS's H_t^L is not defined in Lean. What is formal is part (b), which anchors Φ_χ to $\Lambda(s,\chi)$.

Remark 2.1 (the infimum-free form)**.** As in our earlier papers [20, 21], "$\Lambda_\chi \leq 9/32$" is stated as "for every $t \geq 9/32$ every zero of H_t^χ is real". Once Λ_χ exists and de Bruijn's monotonicity holds, the two are equivalent. Monotonicity is in the formalization (`dbn_real_zeros_upset`); it is stated for abstract kernels and transfers to H_t^χ through the rescaling of Section 3.3. Existence, which rests on Newman's theorem, is not. Stating the bound without an infimum keeps the theorem a statement about the zeros of an explicit entire function.

3 The mathematics

This section is self-contained. Every theorem, proposition and corollary in it is proved in the formalization; the two places where a remark is not formalized say so. Lean names are given in brackets for readers who want to find them.

3.1 The theta kernel and the bridge

Throughout, χ is a non-trivial real Dirichlet character modulo N with parity k: $\chi(-1) = (-1)^k$, so $\Lambda_{\mathrm{M}}(s,\chi) = \Gamma_{\mathbb{R}}(s+k)L(s,\chi)$ away from the poles of $\Gamma_{\mathbb{R}}(s+k)$.

Decay. Since $|\chi(n)| \leq 1$ and $\chi(0) = 0$, for $x \geq 1$

$$|\theta_\chi(x)| \leq \sum_{n\geq 1} n^k e^{-\pi n^2 x/N} \leq e^{-\pi x/(2N)} \sum_{n\geq 1} n^k e^{-\pi n^2/(2N)},$$

using $n^2 x \geq (n^2+x)/2$ for $n, x \geq 1$. Hence, for $u \geq 0$,

$$|\Phi_\chi(u)| \leq S_{k,N}\, e^{(1+2k)u} \exp\!\big(-\tfrac{\pi}{2N}e^{4u}\big) \qquad [\texttt{abs_PhiC_le}]. \tag{4}$$

The true rate is π/N; the weaker rate suffices for everything below. Φ_χ is continuous [`continuous_PhiC`]. It is not identically zero, because the $n = 1$ term dominates as $u \to \infty$ [`exists_PhiC_ne_zero`].

The Mellin transform. For $\operatorname{Re} s > 1$, integrating term by term,

$$\int_0^\infty \theta_\chi(x)\, x^{(s+k)/2}\,\frac{dx}{x} = \sum_{n\geq 1} \chi(n) n^k \Big(\frac{N}{\pi n^2}\Big)^{(s+k)/2} \Gamma\Big(\frac{s+k}{2}\Big) = \Lambda(s,\chi)$$

[`mellin_thetaC`, `div_pi_cpow_mul_Gamma`]. Put $x = e^{4u}$ and $s = \frac{1}{2} + \frac{iz}{2}$. Then $x^{(s+k)/2} = e^{(1+2k)u}e^{izu}$ and $dx/x = 4\,du$, so

$$\Lambda(\tfrac{1}{2} + \tfrac{iz}{2}, \chi) = 4\int_{\mathbb{R}} \Phi_\chi(u)\, e^{izu}\, du \qquad (\operatorname{Im} z < -1) \qquad [\texttt{fourier_PhiC_eq_mellin}]. \tag{5}$$

Evenness and the bridge. If Φ_χ is even, the two-sided integral in (5) is twice the one-sided cosine integral [`Hd_zero_eq_half_fourier`]. So $H_0^\chi(z) = \frac{1}{8}\Lambda(\frac{1}{2} + \frac{iz}{2}, \chi)$ for $\operatorname{Im} z < -1$. Both sides are entire: the left by differentiation under the integral, using (4), and the right because $\chi \neq 1$. Since they agree on an open set, they agree everywhere [`DBN.Gen.bridge`]. Primitivity is not used here, and evenness is used only to fold the integral. The formula holds for every real $\chi \neq 1$ whose kernel is even.

The theta functional equation. For real primitive χ, Poisson summation gives

$$\theta_\chi(1/x) = \varepsilon(\chi)\, x^{(1+2k)/2}\, \theta_\chi(x) \qquad (x > 0). \tag{6}$$

The formalization proves this with $\varepsilon(\chi) = 1$ as a hypothesis [`theta_fe`]. It splits $\sum_{n\in\mathbb{Z}}$ into residue classes modulo N, applies Mathlib's functional equations for the even and odd Hurwitz theta kernels, and recombines the classes with Mathlib's Gauss-sum identity for the Fourier transform of a primitive character (`IsPrimitive.fourierTransform_eq_inv_mul_gaussSum`), using $\chi^{-1} = \chi$. Substituting $x = e^{4u}$ into (6) gives

$$\Phi_\chi(-u) = \varepsilon(\chi)\, \Phi_\chi(u) \qquad [\texttt{PhiC_even_of_rootNumber}, \text{ for } \varepsilon = 1].$$

So Φ_χ is even exactly when $\varepsilon(\chi) = 1$. Gauss's sign theorem supplies this.

3.2 Gauss's sign theorem

Theorem 3.1 (Gauss; Lean: `DBN.Gen.rootNumber_eq_one_of_real_primitive`)**.** *Every real primitive Dirichlet character χ has $\varepsilon(\chi) = 1$, that is, $\tau(\chi) = \sqrt{N}$ if χ is even and $\tau(\chi) = i\sqrt{N}$ if χ is odd.*

```
theorem DBN.Gen.rootNumber_eq_one_of_real_primitive {N : ℕ} [NeZero N] (χ :
    DirichletCharacter ℂ N)
    (hprim : χ.IsPrimitive) (hreal : ∀ n : ℕ, (χ n).im = 0) :
    DirichletCharacter.rootNumber χ = 1
```

The statement uses only constants from Mathlib and `Init`; `rootNumber` is Mathlib's own definition [3, REVIEW_GAUSS_SIGN_AND_OAI §A.3]. Gauss found the sign of the quadratic Gauss sum in 1805 (his letter to Olbers of 3 September 1805 is the usual reference); the proof, *Summatio quarumdam serierum singularium*, was presented in 1808 and printed in 1811 [14]. We have not checked these dates against the originals; for the history of the theorem see Berndt, Evans and Williams [4]. The classical deduction for real primitive characters goes through their identification with Kronecker symbols of fundamental discriminants; see [19, Theorems 9.13, 9.16], as cited in [13, Lemma 2.9]. The formalization follows a different, self-contained route in three stages, 1,419 lines in all:

S1. *The quadratic Gauss sum.* For every $N \geq 1$, $G(N) = \sum_{j=0}^{N-1} e^{2\pi i j^2/N} = \frac{1}{2}(1+i)(1+(-i)^N)\sqrt{N}$ [`gaussG_eq`]. The proof is analytic. It evaluates the Jacobi theta function at $\tau_t = 2/N + it$ as $t \to 0^+$ in two ways: directly, by residue classes, and through the transformation $\tau \mapsto -1/\tau$.

S2. *Prime conductor.* For an odd prime p, a real character $\chi \neq 1$ modulo p is the Legendre symbol and is primitive. Its Gauss sum is $G(p)$, which gives $\varepsilon = 1$ [`rootNumber_eq_one_of_prime`].

S3. *Induction on the conductor.* Strong induction on N peels off one odd prime $p \parallel N$ at a time, writing $\chi = \chi_p \chi'$ by the Chinese remainder theorem. Alongside $\varepsilon = 1$ it carries the invariant $\chi(\ell) = \left(\frac{\chi(-1)N}{\ell}\right)$ for odd primes $\ell \nmid N$. With this invariant, the cross term $\chi_p(N/p)\chi'(p)$ reduces to $\left(\frac{\chi'(-1)}{p}\right)$ without a further appeal to reciprocity; quadratic reciprocity is used only to establish the invariant for the Legendre factor. The base cases are the levels 1, 4 and 8, that is, the characters $1, \chi_{-4}, \chi_8, \chi_{-8}$. The levels 2 and 2^e with $e \geq 4$, and any level divisible by p^2 for an odd prime p, carry no real primitive character [`gauss_sign_aux`].

The theorem quantifies over Mathlib's `IsPrimitive` directly. It does not use the correspondence between real primitive characters and fundamental discriminants.

Remark 3.2 (the sign is needed for evenness, not for the strip)**.** The zero strip of Section 3.4 does not depend on ε, because (3), with $\chi^{-1} = \chi$, sends a zero at s to a zero at $1 - s$ whatever the value of ε. The sign matters only because the formalized de Bruijn theorem is stated for even kernels. The auditor noted an alternative [3, NOTE_D2 §2]. For real primitive χ, applying (3) twice gives $\varepsilon^2 = 1$ without any Gauss-sum work, and the case $\varepsilon = -1$ would need a version of de Bruijn's theorem for odd kernels ($z\,K(z^2)$ in place of $K(z^2)$). This route was not formalized (it is one of the two unformalized remarks of this section); Gauss's theorem was proved instead.

3.3 De Bruijn's strip theorem

Definition 3.3 (Lean: `DBN.Gen.Kernel`)**.** A *kernel* is a continuous even function $\Phi : \mathbb{R} \to \mathbb{R}$, not identically zero, with $|\Phi(u)| \leq Ce^{9u}\exp(-\frac{\pi}{2}e^{4u})$ for $u \geq 0$. Its heat flow is $H_t(z) = \int_0^\infty e^{tu^2}\Phi(u)\cos(zu)\,du$.

The template $e^{9u}\exp(-\frac{\pi}{2}e^{4u})$ is the decay of Riemann's Φ in (1). The definition abstracts the `dbn` island's Φ [20]: the auditor instantiated the generic library with the island's Φ and recovered the island's theorem verbatim [3, REVIEW_GEN_D1_D2a §1].

Theorem 3.4 (de Bruijn [10]; Lean: `DBN.Gen.dbn_debruijn_parametric`)**.** *Let Φ be a kernel, and let $t_0, Y \in \mathbb{R}$. If every zero of H_{t_0} satisfies $(\operatorname{Im} z)^2 \leq Y$, then for every $t \geq t_0$ every zero of H_t satisfies $(\operatorname{Im} z)^2 \leq \max(Y - 2(t - t_0), 0)$.*

This is the form stated in [29, Theorem 3.2] and [13, Theorem 2.6], both of which attribute it to [10, Theorem 13]; we have not consulted de Bruijn's paper directly, and the formal proof does not depend on it. The formalized proof has four steps:

1. *Discretise the heat kernel.* For $s = t - t_0 > 0$, $N \geq 1$ and $\delta = \sqrt{2s/N}$, put
$$G_N(z) = \int_0^\infty \cosh(\delta u)^N\, e^{t_0 u^2}\Phi(u)\cos(zu)\,du.$$
Multiplying the kernel by $\cosh(\delta u)$ corresponds to the vertical shift average $(T_\delta f)(z) = \frac{1}{2}(f(z + i\delta) + f(z - i\delta))$. So $G_N = T_\delta^N H_{t_0}$. Moreover $\cosh(u\sqrt{2s/N})^N \to e^{su^2}$, and $G_N \to H_t$ locally uniformly on $\mathbb{C}$ [`m1W_succ`, `m1_tendstoLocallyUniformly_G`]. The island's design notes record that the cruder discretisation $(1 + su^2/N)^N$ preserves a zero strip but does not shrink it; this is not formalized, and that discretisation is not used.

2. *The step lemma.* Let f be entire, real on $\mathbb{R}$ and even, with an even Hadamard product. If its zeros satisfy $(\operatorname{Im} z)^2 \leq \Delta$, then the zeros of $T_\delta f$ satisfy $(\operatorname{Im} z)^2 \leq \max(\Delta - \delta^2, 0)$ [`shiftAvg_zero_im_sq_le`]. The proof works factor by factor. For $\operatorname{Im} w > 0$ beyond the threshold, each factor of the product is strictly larger at $w + i\delta$ than at $w - i\delta$. So $|f(w + i\delta)| > |f(w - i\delta)|$, which rules out $f(w + i\delta) = -f(w - i\delta)$. Iterating N times lowers Δ by $N\delta^2 = 2s$.

3. *Hadamard.* Each G_N is even, real and entire of order at most $3/2 < 2$, so it has an even Hadamard product. The bound on the order holds for every real t_0, negative included, because the double-exponential decay of Φ absorbs $e^{t_0 u^2}$ [`m1_evenHadamardData_W`].

4. *Hurwitz.* The G_N have no zeros in the open set $\{(\operatorname{Im} z)^2 > \max(Y - 2s, 0)\}$ and converge locally uniformly to H_t. By Hurwitz's theorem H_t has no zeros there either, provided $H_t \not\equiv 0$ [`hurwitz_ne_zero_of_entire`].

Fourier injectivity in place of positivity. For ζ, the island proved $H_t \not\equiv 0$ from $\Phi > 0$, which gives $H_t(0) > 0$. A Dirichlet kernel Φ_χ need not be positive. And $H_0^\chi(0) \neq 0$ is equivalent to $L(\frac{1}{2}, \chi) \neq 0$, which is Chowla's conjecture and open in general. The generic library therefore uses Fourier injectivity instead [`exists_cosTransform_ne_zero`, `H_exists_ne_zero`]. Let w be even, continuous, integrable and not identically zero. If its cosine transform vanished on $\mathbb{R}$, then, w being even, its whole Fourier transform would vanish, and Fourier inversion (Mathlib's `Continuous.fourierInv_fourier_eq`) would give $w = 0$. Applying this to $w(u) = e^{tu^2}\Phi(u)$ shows $H_t \not\equiv 0$ for every real t, with no arithmetic input.

Other decay rates. A *decay kernel* [`DBN.Gen.DecayKernel`] is a continuous even $\Phi \not\equiv 0$ with $|\Phi(u)| \le Ce^{au}\exp(-be^{4u})$ for $u \ge 0$, for some real a and some $b > 0$. Its rescaling $\Phi(2\,\cdot)$ is a kernel in the sense above: with $x = e^{4u}$,

$$2au - bx^2 \le \frac{(|2a-9| + \pi/2)^2}{4b} + 9u - \frac{\pi}{2}x \qquad (u \ge 0),$$

and the substitution $v = 2u$ gives $H_t^\Phi(z) = 2H_{4t}^{\Phi(2\cdot)}(2z)$ [`DecayKernel.template`, `DecayKernel.Hd_eq`]. Under the rescaling a strip $(\operatorname{Im} z)^2 \le Y$ at time 0 becomes $(\operatorname{Im} z)^2 \le 4Y$, and time t becomes $4t$. Theorem 3.4 then gives:

Corollary 3.5 (Lean: `DecayKernel.real_zeros_of_strip`)**.** *Let $Y \ge 0$. If every zero of H_0^Φ satisfies $(\operatorname{Im} z)^2 \le Y$, then every zero of H_t^Φ is real for $t \ge Y/2$.*

By (4), Φ_χ is a decay kernel with $a = 1 + 2k$ and $b = \pi/(2N)$, once it is known to be even [`thetaDecayKernel`].

3.4 From the half-plane $7/8$ to the strip $D = 3/4$

Proposition 3.6 (Lean: `DBN.Gen.zero_strip`)**.** *Let χ be real, primitive and non-trivial, with Φ_χ even, and suppose $L(s, \chi) \neq 0$ for $\operatorname{Re} s > 7/8$. Then every zero z of H_0^χ satisfies $(\operatorname{Im} z)^2 \le 9/16$.*

Proof. Let $H_0^\chi(z) = 0$ and $s = \frac{1}{2} + \frac{iz}{2}$, so $\operatorname{Re} s = \frac{1}{2} - \frac{1}{2}\operatorname{Im} z$. By the bridge and $N^{(s+k)/2} \neq 0$, $\Lambda_{\mathrm{M}}(s, \chi) = 0$. If $\operatorname{Re} s > 7/8$, then $\Gamma_{\mathbb{R}}(s+k) \neq 0$ and $L(s, \chi) = \Lambda_{\mathrm{M}}(s, \chi)/\Gamma_{\mathbb{R}}(s+k) = 0$, a contradiction; so $\operatorname{Re} s \le 7/8$. A real-valued character is its own inverse: $|\chi(a)| = 1$ on units together with $\chi(a) \in \mathbb{R}$ forces $\chi(a) = \pm 1$ [`inv_eq_self_of_real`]. So (3) gives $\Lambda_{\mathrm{M}}(1 - s, \chi) = N^{s-1/2}\varepsilon(\chi)\Lambda_{\mathrm{M}}(s, \chi) = 0$, and the same argument applied to $1 - s$ gives $\operatorname{Re}(1 - s) \le 7/8$, that is, $\operatorname{Re} s \ge 1/8$. Hence $|\operatorname{Im} z| \le 3/4$. □

Primitivity enters in three places: here, through (3); in the theta functional equation (6); and in Gauss's theorem. With $Y = 9/16$, Corollary 3.5 gives real zeros for $t \ge 9/32 = 2(7/8 - 1/2)^2$ [`DBN.Gen.dirichlet_real_zeros`]. Discharging evenness by Theorem 3.1 [`DBN.Gen.real_primitive_real_zeros`], and the half-plane by OpenAI's theorem with $\chi \neq 1$ excluding the pole, proves Theorem 1.1(a).

3.5 An example: χ_{-4}

The non-trivial character modulo 4, $\chi_{-4}(n) = (-1)^{(n-1)/2}$ for odd n, is odd ($k = 1$), so

$$\theta_{\chi_{-4}}(x) = \sum_{n \text{ odd}} (-1)^{(n-1)/2}\, n\, e^{-\pi n^2 x/4}, \qquad \Phi_{\chi_{-4}}(u) = e^{3u}\, \theta_{\chi_{-4}}(e^{4u}).$$

Its Gauss sum is $\tau(\chi_{-4}) = e^{2\pi i/4} - e^{2\pi i\cdot 3/4} = 2i$, so $\varepsilon(\chi_{-4}) = 2i/(i \cdot 2) = 1$, and $\theta_{\chi_{-4}}(1/x) = x^{3/2}\theta_{\chi_{-4}}(x)$. In Lean this character is `chiM4 := ZMod.χ4.ringHomComp (Int.castRingHom ℂ)`. Its primitivity, non-triviality, reality, oddness, $\tau = 2i$ and $\varepsilon = 1$ are proved by direct computation in `GenChi4.lean`, independently of the general Gauss theorem [`chiM4_isPrimitive`, `gaussSum_chiM4`, `rootNumber_chiM4`]. The corresponding special case was the first one to be proved in OpenAI's environment:

```
theorem bb_chiM4_real_zeros_oai :
    ∀ t : ℝ, 9 / 32 ≤ t → ∀ z : ℂ, DBN.Gen.Hchi DBN.Gen.chiM4 t z = 0 → z.im = 0
```

As an independent numerical sanity check of the bridge, the auditor compared both sides in `mpmath` at 30 digits [3, REVIEW_D2b_BRIDGE]. The left side was $H_0^{\chi_{-4}}(z)$, computed by quadrature; the right side was $\frac{1}{8}4^{(s+1)/2}\Gamma_{\mathbb{R}}(s+1)L(s,\chi_{-4})$:

z	both sides	|difference|
0	0.122589201757214188	$5 \cdot 10^{-31}$
1	0.120217919764671738	$1 \cdot 10^{-32}$
$0.5 + 0.25i$	$0.122140362295467626 - 0.000595884686946844\, i$	$5 \cdot 10^{-31}$

At $z = 0$ the value is $\frac{1}{8}\Lambda(\frac{1}{2}, \chi_{-4})$, numerically non-zero; the proof does not use this. FGMPS prove the stronger unconditional bound $\Lambda_{\chi_{-4}} \le 0.26$ [13, §7.5]. Moreover, once OpenAI's theorem is granted, 9/32 for χ_{-4} already follows on paper from FGMPS's Remark 3.4. The example is here to make the objects concrete; what is new for χ_{-4}, as for every conductor, is only that the bound is a kernel-checked theorem.

3.6 What the proof does not use

Apart from OpenAI's theorem, whose internals we do not describe, the proof uses no numerical verification of GRH, no zero computation, no Riemann–Siegel formula, no interval arithmetic, no positivity of Φ_χ, and no non-vanishing at the central point. Its only arithmetic inputs are OpenAI's half-plane and Gauss's sign theorem. Everything else is the functional equation and complex analysis.

4 The formalization

4.1 Architecture

The proof has five layers. Each imports only Mathlib and the layers above it; the composition layer also imports OpenAI's `OAI.NumberTheory.DirichletL.Nonvanishing`. Line counts were taken from the audited sources; the md5 prefixes are those recorded in the auditor's reviews [3] and were rechecked for this draft.

Layer	Files, lines	Content
`dbn` island [20] (commit `afd94c7`)	7, 2,193	`DBNStep` (the shift average and the step lemma), `DBNHurwitz`, and `DBNHadamard` with its four sub-modules (Hadamard factorisation of even entire functions of order < 2). Pure complex analysis, with no Φ.
`Gen`	13, 2,788	`GenFourier`, `GenDefs`, `GenHeatApprox`, `GenM1Approx`, `GenM1Parametric`: Theorem 3.4 for abstract kernels. `GenScale`: Corollary 3.5. `GenTheta`, `GenMellin`, `GenMellin2`, `GenBridge`: Section 3.1. `GenStrip`: Proposition 3.6. `GenThetaFE`: (6), `Hchi`. `GenChi4`: Section 3.5.
`Gauss`	6, 1,419	`GaussQuad` (S1), `GaussPrime` (S2), `GaussCRT`, `GaussPow`, `GaussTwo`, `GaussSign` (S3): Theorem 3.1 and `real_primitive_real_zeros`, the main theorem with the half-plane as a hypothesis.
`GaussBridge`	1, 36	`Hchi_bridge`: part (b) with no extra hypotheses.
Composition	2	`BbOaiRealPrim`, `BbOaiMeaning`: OpenAI's theorem discharges the half-plane; the conjunction.

The seven island files (in the order of the table) have md5 prefixes `e76f64ba`, `1ca47174`, `8069c2da`, `da8f08cd`, `f6e28cd1`, `b4d07462`, `46c3b886` (`DBNHadamardCount`, `DBNHadamardLinear`, `DBNHadamardMean`, `DBNHadamardProduct`, `DBNStep`, `DBNHurwitz`, `DBNHadamard`). The thirteen `Gen` files and six `Gauss` files have md5 prefixes `261aa493`, `c5c6e856`, `282fe6c7`, `23f22b65`, `99f5dfdc`, `6f608628`, `3dbf6dec`, `cb5c2820`, `67e1af81`, `5691f87d`, `8aa81558`, `8aeec9b0`, `9acc262c` and `51b4e494`, `6fb94967`, `dce09dba`, `16bd30ef`, `3800b2cc`, `03883647`; `GaussBridge` is `4f746054`; the composition files are `BbOaiRealPrim` `4c569252` and `BbOaiMeaning` `23ea25e9`. Four of the `Gen` files (`GenDefs`, `GenHeatApprox`, `GenM1Approx`, `GenM1Parametric`) are generated mechanically from the island by a text transform that abstracts Φ. The auditor re-ran the generator and obtained byte-identical outputs [3, REVIEW_GEN_D1_D2a].

The main theorem with the half-plane as a hypothesis is:

```
theorem DBN.Gen.real_primitive_real_zeros {N : ℕ} [NeZero N] (χ : DirichletCharacter ℂ
    N)
    (hprim : χ.IsPrimitive) (hχ1 : χ ≠ 1) (hreal : ∀ n : ℕ, (χ n).im = 0)
    (hL : ∀ s : ℂ, (7 / 8 : ℝ) < s.re → DirichletCharacter.LFunction χ s ≠ 0) :
    ∀ t : ℝ, 9 / 32 ≤ t → ∀ z : ℂ, Hchi χ t z = 0 → z.im = 0
```

The composition with OpenAI's theorem, in full, is:

```
import GaussSign
import OAI.NumberTheory.DirichletL.Nonvanishing

theorem bb_real_primitive_real_zeros_oai :
    ∀ {N : ℕ} [NeZero N] (χ : DirichletCharacter ℂ N), χ.IsPrimitive → χ ≠ 1 → (∀ n
    : ℕ, (χ n).im = 0) →
    ∀ t : ℝ, 9 / 32 ≤ t → ∀ z : ℂ, DBN.Gen.Hchi χ t z = 0 → z.im = 0 :=
  fun χ hprim hχ1 hreal ↦ DBN.Gen.real_primitive_real_zeros χ hprim hχ1 hreal (fun _
    hs ↦
    OAI.DirichletCharacter.LFunction_ne_zero_of_seven_eighths_lt_re χ hs (fun h ↦ hχ1
    h.1))
```

OpenAI's side condition $\neg(\chi = 1 \wedge s = 1)$ follows from $\chi \neq 1$.

4.2 Two environments

OpenAI's theorem lives at Lean `v4.34.1` with Mathlib `d13f23b`. Our chain was developed and first audited at Lean `v4.35.0-rc3` with Mathlib `0493595e`. Combining a theorem from one Mathlib revision with a hypothesis stated in another would leave a seam: the two revisions must define `DirichletCharacter.LFunction` in the same way. We avoided that question. The whole chain was rebuilt from source inside OpenAI's built environment, with an empty port diff: all 27 chain files, island included, are byte-identical to the versions audited at `v4.35.0-rc3` (the island files differ from the island's release `afd94c7` by two previously audited port lines). The final theorem was then proved there [3, REVIEW_GAUSS_SIGN_AND_OAI §B.1; ADD1 §3]. The OpenAI environment is a materialised copy of `openai/math`'s `lean/` directory from a replay on 7 October 2026. A recursive `diff` of its `OAI` tree against a git clone at `adc7f1241b42`, covering 121,734 files, is empty. The lakefile, manifest, toolchain and patches agree, and `Nonvanishing.lean` (md5 `ea8efdb1`) re-elaborates from source [3, REPORT_OAI_PORT (b)]. The dependency checkouts of that local tree (`.lake/packages`, Mathlib among them) were not hashed against the pinned commits; the sandboxed Comparator run re-created them from the public pins: Mathlib from its official build cache at `d13f23b`, and the two other packages OpenAI's theorem imports (`PrimeNumberTheoremAnd`, `rellich-kondrachov`) cloned at their pins and patched with OpenAI's patches. Its solution export is byte-identical to the one the auditor made from the local tree (Section 5.6), which closes this gap for the closure of the main theorem [3, REVIEW_COMPARATOR_LAMBDA_CHI §4].

4.3 The trust base

To believe Theorem 1.1 on the strength of this formalization, one has to trust:

1. *The Lean kernel*, plus the further checkers that replayed the export (Section 5), from three implementation lineages.

2. *OpenAI's Lean proof* of the 7/8 half-plane, as checked by the kernel. Its mathematical manuscript [25] is unrefereed, as is the companion 11/12 preprint [24]. Its proof term is in the replayed environment, so the kernel checks are a check of OpenAI's proof too. But no human has refereed the argument.

3. *Mathlib's definitions* at `d13f23b`, in particular `DirichletCharacter`, `IsPrimitive`, `LFunction`, `completedLFunction`, `Gamma`$\mathbb{R}$, the complex power, `tsum` and the Bochner integral, as compiled in Mathlib's official build cache at `d13f23b`, from which the sandboxed run took them; the local build tree gives a byte-identical export of the main theorem's environment (Section 5.6).

4. *Our four definitions* of Section 2 (`thetaC`, `PhiC`, `parityK`, `Hchi`). Part (b) of Theorem 1.1, with Fourier uniqueness, is the check that `Hchi` means what we say; what remains to be read is its outer heat-integral shape. The identification with FGMPS's normalisation for $t \neq 0$ (Section 2.4) is by hand.

5. *The checking runs themselves.* The local replays on macOS were unsandboxed. The sandboxed Comparator run of Section 5.6 ran as uid 0 with all capabilities dropped; it was made by another agent session, and the auditor read all of its logs but did not repeat its kernel runs.

Not trusted: any proof in our chain or in OpenAI's library. Every proof is checked by the kernel.

4.4 How to read and check it

A reader who wants to check the statement without reading proofs needs three things: `ChallengeDefs.lean` (the four definitions, 56 lines, md5 `67e538bc`); `Challenge.lean` (the statement of Section 2 with `:= sorry`, md5 `73c1bf94`); and the Mathlib definitions listed above. A reader who wants to follow the proof should read `GenStrip.lean` (89 lines) first: it is the proof of Proposition 3.6 and of the main theorem in hypothesis form. Then read `GenScale.lean` and `GenM1Parametric.lean` for de Bruijn, and `GenThetaFE.lean` and `GaussSign.lean` for the arithmetic. The release repository's README gives two routes for reproducing the result, summarised here.

1. *The auditor's build.* Clone `openai/math` and check out `adc7f124`; its Lean project is `lean/`, with toolchain `v4.34.1` and Mathlib `d13f23b`. In `lean/`, run `lake update`, `lake exe cache get` and `lake build OAI.NumberTheory.DirichletL.Nonvanishing`, which is OpenAI's documented procedure and applies their dependency patches. Afterwards, check that `lean/lake-manifest.json` is unchanged (md5 `8c3ec864`), for example with `<release>/scripts/check_manifest.sh <clone>/lean/lake-manifest.json`; otherwise the dependencies are no longer those pinned by OpenAI. Then compile the 29 chain modules of the release (`lean/island`, `lean/gen`, `lean/gauss`, `lean/main`) with plain `lean`, in the order listed in the README, against that build. Finally run `BbOaiMeaningCheck.lean` (axioms and closure walks for both halves), `BbOaiMeaningRef.lean` (the `pp.all` statement with no OpenAI module in scope) and `BbOaiMeaningTamper.lean`, which must fail with 3 errors, and compare the outputs with `logs/oai_env/`. This mirrors how the Lean results of Section 5 were obtained; the auditor used OpenAI's tree as built in the 7 October replay (Section 4) rather than a fresh clone.

2. *Through Lake.* `scripts/verify_md5.sh` checks every file under `lean/`, `logs/` and `provenance/` against its manifest, and the challenge files against the auditor's two manifests in `challenge/`. `scripts/materialize.sh` sets up `openai/math` at the pin with the chain inside it as five `lean_lib` targets; one then builds those targets and runs the same three checks with `lake env lean`. This route, and the continuous-integration workflow that runs it on Linux, had not been run at the time of this version. (OpenAI's package sets `autoImplicit` off; the stanza sets it back on for each of the five chain libraries, matching how the chain was compiled, but this override is untested. A mismatch could only fail loudly, as an unknown-identifier error, and could not change a statement.)

The five-checker replay of Section 5 exports the environment with `lean4export` and runs each checker on the export, as in the scripts under `logs/five_checker/`.

5 Verification

All results in this section were produced by the auditor (an AI agent session; see Section 8) and are recorded, with md5 sums for every input and output, in its reviews [3].

5.1 Axioms

`#print axioms` reports `[propext, Classical.choice, Quot.sound]` for `bb_real_primitive_meaning_oai`, `Hchi_bridge`, `bb_real_primitive_real_zeros_oai` and `rootNumber_eq_one_of_real_primitive`. `#print axioms` reads a cached summary, so the auditor also walked the term-level closure directly: types, values, constructors and recursor rules. The walk over the 126,656 constants

in the closure of the main theorem finds the same three axioms [3, ADD1 §3]. A static scan of the Gauss and bridge sources finds no `sorry`, `admit`, `axiom`, `native_decide`, `implemented_by`, `extern`, `unsafe`, `opaque`, `set_option`, `macro`, `elab` or `#eval`. The only textual hit, in a doc comment, is the word "admit" [3, §A.2; ADD1 §1].

5.2 Five checkers from three lineages, and a control they must reject

The environment containing the main theorem was exported with `lean4export` [6] (v4.34.1 build, export format 3.1.0) to a file of 1,730,561,283 bytes (md5 `6888b248`). The export was given to five checkers. These are the checkers bundled with the Lean `v4.35.0-rc3` toolchain, which read the same export format. Alongside it, a control was exported and checked: a "theorem" `BbNeg.bogus : 4 = 5` admitted into the environment by switching off the elaborator's kernel check (`debug.skipKernelTC`; export md5 `bf973608`). A checker that accepts the control is not checking anything. The runs were guarded so that an empty or failed export aborts the run, and any acceptance of the control marks it invalid. The guards exist because an earlier run in the same programme exported zero bytes because of a toolchain mismatch, and every checker then "accepted" the empty file, the control included [3, REVIEW_CTFREE_KERNELS].

Checker	main theorem (`6888b248`)	control `4 = 5`
`leanchecker` (Lean's kernel)	accept, 270 s	reject (rc 1)
`leanchecker-paranoid`	accept, 320 s	reject (rc 1)
`nanoda` [2]	accept, 57 s	reject (rc 101, panic in `def_eq`)
`con-leche`	accept, "accepted 124635 declarations (`--verified`)"	reject (rc 1)
`con-ron`	accept, "accepted 124635 declarations (`--verified`)"	reject (rc 1)
`lean4lean` [7]	N/A: crashes before reaching any target	N/A

All five accept, and all five reject the control [3, ADD1 §4]. The rc3 `lean4lean` cannot read v4.34.1 exports: it fails with "unknown constant `Decidable.intro`" on every file, the control included, so it is recorded as not applicable rather than as a rejection. The two `leanchecker` variants run Lean's own kernel. `nanoda` is an independent implementation in Rust. `con-leche` is a separate kernel implementation written in Lean, with its own term datatypes (`ConLeche.Expr`, `ConLeche.Level`, `ConLeche.Name`); its help text states a soundness theorem for its default `--verified` mode, "`ConLeche.no_proof_of_False`" (if the declaration fold accepts, no constant of type `False` is in the environment). `con-ron` describes itself as a Rust port of `con-leche`. Both are bundled with the Lean `v4.35.0-rc3` toolchain, whose `lake comparator --paranoid` runs them alongside `leanchecker-paranoid`, `lean4lean` and `nanoda` (`Lake/CLI/Check.lean` in the toolchain's sources). `con-leche` [17] was implemented, and proved consistent in Lean, by Claude under the supervision of Joachim Breitner at the Lean FRO; its README states the theorems `no_False_declaration` and `model_exists` (every accepted environment has a model in set theory, following Carneiro's model of Lean's type theory). `con-ron` [18] is a Rust port of it, written by AI under Breitner's supervision, with an Aeneas proof that it refines `con-leche`'s checking algorithm; it differs in term representation, caching, compiler and runtime. Both ship with Lean `v4.35.0` [11]. They are independent of Lean's C++ kernel, but not of each other. The five acceptances therefore come from three implementation lineages: Lean's kernel, `nanoda`, and the `con-leche`/`con-ron` pair. Separate runs of the same kind accepted part (a) alone (export md5 `ae0c51bd`, 124,633 declarations) and the χ_{-4} case (`7c142b2c`,

124,395 declarations), and rejected the control [3, REVIEW_GAUSS_SIGN_AND_OAI §B.6; REPORT_OAI_PORT (a)].

5.3 Tampered statements

Each headline statement was also compiled with one number changed, with the proof unchanged. We distinguish two kinds of rejection. A *genuine* rejection means the changed statement is false, or contradicts a proved lemma. A *proof-reach* rejection means only that the given proof does not reach the changed statement; it shows a hypothesis or constant is load-bearing, not that the changed statement is false.

Change	Result	Kind
half-plane $7/8 \to 3/4$ in the composition	rejected at OpenAI's theorem	proof-reach (OpenAI's theorem does not reach 3/4)
heat time $9/32 \to 1/4$	rejected	proof-reach
bridge constant $1/8 \to 1/4$	rejected	genuine
parity $k \to 1-k$ in the bridge	rejected	genuine
bridge without primitivity	rejected at `assumption`	proof-reach
$\varepsilon(\chi) = -1$ for real primitive χ	rejected	genuine
$(1+i) \to (1-i)$ in the closed form of $G(N)$	rejected	genuine
$\varepsilon(\chi_{-4}) = -1$; $\tau(\chi_{-4}) = -2i$	rejected	genuine

The auditor's earlier records called the first row "genuine", meaning that the rejection happens at OpenAI's interface; under the definition above it is proof-reach, since the changed hypothesis is implied by GRH and is not known to be false. Sources: [3, REVIEW_GAUSS_SIGN_AND_OAI §§A.6, B.5; ADD1 §§2, 3; REVIEW_D2bp_CHI4].

5.4 Statement identity with and without OpenAI's code

Loading OpenAI's library into the environment could, in principle, change what a statement means, for instance through an instance or a declaration that shadows a Mathlib name. To rule this out, the auditor printed the statements with `pp.all`, in which every implicit argument and instance is explicit, in two environments: one containing OpenAI's library, and one containing only our chain and Mathlib. The text of part (a) (4,017 characters) and the text of part (b) (7,778 characters) are byte-identical across the two [3, REVIEW_GAUSS_SIGN_AND_OAI §B.3; ADD1 §3]. At the χ_{-4} stage the OpenAI environment had 13,708 modules, with no module-name collisions and no `DBN.*` constant from a foreign module [3, REPORT_OAI_PORT].

5.5 Independent numerics

These computations check conventions and normalisations; they are plain floating point or `mpmath`, not certificates. With its own implementation of the Kronecker symbol, in Mathlib's normalisation $\varepsilon = \tau/(i^k\sqrt{N})$, the auditor checked [3, §A.7]:

- the closed form of $G(N)$ for $N = 1, \dots, 200$ (largest error $6.6 \cdot 10^{-38}$);
- $\varepsilon = 1$ for real characters of odd prime conductor $p < 500$ ($4.2 \cdot 10^{-41}$);
- $\varepsilon(\chi_D) = 1$ for every fundamental discriminant $0 < |D| \le 2000$, of both signs: 1,218 discriminants, largest error $4.3 \cdot 10^{-31}$.

The bridge was also checked numerically for χ_{-4} (Section 3.5).

5.6 The sandboxed Comparator run

What was run [3, COMPARATOR_RUN_SPEC]. Comparator [5] checks a solution against a separately written challenge. It elaborates the challenge statement against a minimal trusted definitions file. It requires that the solution's theorem have exactly that type and that every definition the statement uses be identical, as a `ConstantInfo`, to the challenge's. It then checks the axiom whitelist and replays the proof through the kernel and further checkers, inside a sandbox. For this theorem the auditor wrote:

- `Challenge.lean` (md5 `73c1bf94`): the statement of Section 2, verbatim, with `:= sorry`;
- `ChallengeDefs.lean` (md5 `67e538bc`): the trusted definitions. It introduces exactly six non-Mathlib constants: `thetaC`, `PhiC` with its two auxiliary proof terms, `parityK` and `Hchi`;
- `config.json` (md5 `d563faf7`): axioms `propext`, `Quot.sound`, `Classical.choice`; nanoda enabled;
- two tampered challenges that must be rejected: heat time $9/32 \to 1/4$ (`d6ef5198`) and bridge constant $1/8 \to 1/4$ (`31163137`).

The run was made inside a `bwrap` sandbox on a Linux machine, on OpenAI's environment plus our chain at the audited md5s, once with Comparator's default kernels and once with the paranoid kernel set; the auditor then read the run's logs in full and verified them [3, REVIEW_COMPARATOR_LAMBDA_CHI and its Addendum 1].

Caveat that the result carries. The run was made as uid 0 with all capabilities dropped (`--cap-drop ALL`), in an ephemeral single-tenant container; Comparator's assumption that the checker is not run by a privileged user is met in spirit, not in letter. Mathlib was taken from its official build cache at `d13f23b`. The challenge-side oleans were compared with a fresh rebuild after the run, because the queued runs share one build directory (Comparator's assumption that the solution has not been compiled before the challenge).

Results. The run was made on 30 October 2026 by another agent session, with Comparator `v4.34.0` and a `bwrap` sandbox in place of `landrun` (uid 0, all capabilities dropped, no network), on OpenAI's environment rebuilt from its public pins (Mathlib from its official build cache at `d13f23b`; `PrimeNumberTheoremAnd` and `rellich-kondrachov` cloned at their pins and patched with OpenAI's patches; OpenAI's sources unmodified) and our chain at the audited md5s. With the challenge files listed above, Comparator printed "Your solution is okay!" both with `nanoda` and Lean's kernel (log md5 `fc80db0a`) and with the paranoid set `leanchecker-paranoid`, `nanoda`, `con-leche`, `con-ron` and Lean's kernel (log md5 `fb164675`; `con-leche` and `con-ron` report 124,639 declarations); `lean4lean` cannot read `v4.34.1` exports and was not run. In Comparator `v4.34.0` that message is printed only after the statement comparison, the axiom check and every kernel have passed. Both tampered challenges were rejected with "Challenge and solution theorem statement do not match" (log md5s `78a706f6`, `d1f16445`). The auditor read all of the run's logs ([3, REVIEW_COMPARATOR_LAMBDA_CHI, Addendum 1]) and reproduced the solution export independently, from its own build: it is byte-identical (1,730,282,328 bytes, md5 `5ca44ad5`), so the sandboxed kernels checked, byte for byte, the export of the auditor's own build. The five-checker replay of Section 5 used an export of that build with fewer targets (md5 `6888b248`): its 124,635 declaration records contain the whole closure of the main theorem and are a subset of these 124,639. The extra four, `Nat.lor`, `Nat.xor`, `String.mk` and an auxiliary proof, lie outside that closure; Comparator exports them because it lists them as kernel primitives. The auditor also

ran `con-leche` on its own copy of the `5ca44ad5` export, and it accepted all 124,639 declarations [3, REVIEW_COMPARATOR_LAMBDA_CHI, Addendum 2]. The solution was compiled inside the sandbox, in an earlier attempt of the same Comparator command that a container restart interrupted (log md5 `144a7c6b`); the recorded run replayed those build outputs. Before that attempt the build directory was wiped and Mathlib's cache downloaded afresh, discarding an earlier attempt that had been made without the capability drop. Between the two attempts one unsandboxed `lake build Challenge` was run to test Lake's replay behaviour; it compiled nothing (these preparatory steps are from the runner's record; the logs are consistent with them). Because the queued runs share one build directory, the challenge-side oleans were rebuilt afterwards in a fresh directory and found identical (Comparator's assumption that the solution was not compiled before the challenge) [3, REVIEW_COMPARATOR_LAMBDA_CHI §§2–5 and Addenda 1–2].

6 Optional: companion results for ζ that use no OpenAI code

This section is independent of the rest of the paper and may be omitted.

Cureton and Thorner [8] proved, with the AxiomProver system, a zero-free half-plane for every Dedekind zeta function. Version 1 of their report states $\operatorname{Re} s > 1 - 1/6000$, as does their repository [9] at its first public commit `284348a4`. Commit `201bc40` (13 October 2026) strengthens the theorem `main` to $\operatorname{Re} s > 31/32$ for every number field; we use the repository at `09afbb4`, which also adds a Hecke-L corollary at 31/32. We have seen no revised report stating 31/32; that half-plane rests on the Lean repository. At $K = \mathbb{Q}$ this gives $\zeta(s) \neq 0$ for $\operatorname{Re} s > 31/32$, $s \neq 1$. The reduction from their statement to Mathlib's `riemannZeta` was written and audited in this programme [3, REVIEW_CTQ]. Their code imports no module of `openai/math` [3, REVIEW_CTQ]. Two classical consequences were then formalized:

1. $\Lambda \leq 225/512$. For the ζ heat flow (1), as defined on the `dbn` island: every zero of H_t is real for $t \geq 225/512 = 2(31/32 - 1/2)^2$. This is the same composition as Proposition 3.6, with $D = 15/16$.

2. *Ingham's theorem in the kernel.* The general statement `Ingham.ingham`:

   ```
   ∀ {a : ℝ}, 1/2 ≤ a → a < 1 → (∀ s : ℂ, a < s.re → riemannZeta s ≠ 0) →
     ∀ ε : ℝ, 0 < ε → ∃ C, ∀ x : ℝ, 2 ≤ x → |ψ x - x| ≤ C * x ^ (a + ε)
   ```

   with Mathlib's Chebyshev ψ and real powers, and C depending only on a and ε. The proof uses a Perron formula with the weight $1/(s(s+1))$. The main term comes from writing $-\zeta'/\zeta = 1/(s-1) - \zeta_1'/\zeta_1$ with $\zeta_1 = (s-1)\zeta$ and evaluating the $1/(s-1)$ piece exactly. The rest of the integrand, $(\zeta_1'/\zeta_1)(s)\, x^s/(s(s+1))$, is holomorphic on $\operatorname{Re} s > a$, $s = 1$ included, so the shift from $\operatorname{Re} s = 2$ to a line $\operatorname{Re} s = \sigma_0$ with $a < \sigma_0 < 1$ needs no residue theorem. A Borel–Carathéodory bound controls ζ_1'/ζ_1. At $a = 31/32$ it gives $\psi(x) = x + O_\varepsilon(x^{31/32+\varepsilon})$. This is the classical direction of Ingham [15]; nothing in it is new mathematics.

Neither bound is close to the state of the art: $225/512 \approx 0.439$ is far weaker than Platt and Trudgian's $\Lambda \leq 0.2$ [27] and Polymath15's 0.22 [29], and $\Lambda \leq 9/32$ for ζ has been kernel-checked from OpenAI's theorem [20]. Their only point is independence: both hold in an environment that contains no OpenAI code.

Status of the checks. In hypothesis form, with Cureton–Thorner's conclusion $\forall s,\ 31/32 < \operatorname{Re} s \to s \neq 1 \to \zeta(s) \neq 0$ as an explicit hypothesis, both compositions carry the standard three axioms. Six checkers (`leanchecker`, `leanchecker-paranoid`, `lean4lean`, `nanoda`, `con-leche`,

`con-ron`) accepted both, and all six rejected the same kind of `skipKernelTC` control. This is a same-version run, in which `lean4lean` can read the export [3, REVIEW_CTFREE_KERNELS]. Tampers (the exponent 15/16 from the 31/32 half-plane, and the quantifier swap $\exists C\,\forall\varepsilon$) were rejected [3, REVIEW_INGHAM]. The unconditional forms rest on Cureton–Thorner's theorem `main`. The auditor did not run their code. All three Comparator runs below were made on 30–31 October 2026 by another agent session. Each used the toolchain-provided `lake comparator` of Lean `v4.35.0-rc3` with `--paranoid` (`leanchecker-paranoid`, `lean4lean`, `nanoda`, `con-leche`, `con-ron` and Lean's kernel), in a `bwrap` sandbox as uid 0 with all capabilities dropped, on Cureton–Thorner's repository at `09afbb4` (its lakefile extended by 30 additive lines that only declare the libraries of our compositions and challenges) with Mathlib from its official build cache. The installed toolchain, checkers included, is file-for-file the published `v4.35.0-rc3` release. The auditor checked each run's relayed configuration, challenge and solution md5s, sandbox probe, verdicts and exit code against its own records, but did not see the logs [3, REVIEW_CT_RERUNS]. Cureton–Thorner's own challenge (`main` and `heckeCorollary`): "Your solution is okay!", all six kernels accepting (log md5 `a33d06b0`). The P2 and Ingham statements are kernel-checked from the type of `main` itself, whose reduction to Mathlib's `riemannZeta` was audited separately [3, REVIEW_CTQ, Addendum 1]. P2: PASS, six kernels (log `fed70f9a`); tamper (224/512) rejected, exit 1 (log `90424a05`); the trusted definitions of the P2 challenge match the audited `DBNDefs` (md5 `acfa852d`). Ingham: PASS, six kernels (log `dd6fa95b`); tamper (15/16) rejected (log `33d0ea1c`). Every solution module these runs used was compiled inside the sandbox, except Cureton–Thorner's vendored class-field-theory package, which had been compiled earlier without it; rebuilt inside the sandbox at the same path, it gave byte-identical build outputs. These runs replace earlier ones whose sandbox did not drop capabilities. The challenge-side build outputs were rebuilt afterwards in a fresh directory and found byte-identical (Comparator's assumption that the solution was not compiled before the challenge). The result rests on Cureton–Thorner `main` at `09afbb4`.

7 Limitations

1. *OpenAI's manuscript is unrefereed.* The 7/8 half-plane is a kernel-checked Lean theorem, but its mathematical proof has not been refereed, and FGMPS treat it as "asserted". Our theorem is exactly as reliable as that Lean proof and the definitions it is stated in.

2. *The replays are local and unsandboxed.* All the checks of Section 5 ran on one macOS machine, outside any sandbox, driven by AI agent sessions. The sandboxed Comparator run (Section 5.6) closes this gap only in part: it ran as uid 0 with all capabilities dropped, Mathlib came from its official build cache, `lean4lean` could not take part, and it was made by another agent session: the auditor read its logs and reproduced the solution export byte for byte, but did not repeat its kernel runs.

3. *Cross-version checking.* The export was made with the v4.34.1 exporter and checked by checkers bundled with v4.35.0-rc3, which read the same export format. `lean4lean` could not take part.

4. *Mathlib's definitions are trusted.* In particular `IsPrimitive`, `LFunction` and `completedLFunction` are taken to mean what their docstrings say. Part (b) of Theorem 1.1 makes our own definition of H_t^χ answer to Mathlib's L-function, but it cannot check Mathlib.

5. *Gauss's sign theorem is proved for real primitive characters only.* That is all the proof needs. We do not prove anything about the root number of a complex character, and the remark in Section 3.2 about avoiding the sign altogether is unformalized.

6. *The constant Λ_χ is not defined in Lean* (Remark 2.1). Nor is the identification of H_t^χ with FGMPS's H_t^L for $t \neq 0$, which is by hand (Section 2.4).

7. *The local build tree's dependencies were not hashed.* See `provenance/OAI_ENV.md` in the release. The kernel replays check every proof, but the definitions the statement uses were taken as they stood in that tree. For the main theorem's closure this gap is closed, relative to Mathlib's official build cache at `d13f23b`, by the byte-identical export of Section 5.6.

8. *The audits were done by AI sessions.* No human has read the Lean proofs; the kernel checks them, and a human should read the statement and `ChallengeDefs` (Section 2).

9. *The numerics are checks, not proofs.* Nothing in the formal proof depends on them.

8 Use of AI

This work was done with AI agent sessions (Claude, by Anthropic) under the direction of the author. We describe who did what as precisely as we can.

- *Proof engineering.* An agent session (the "contrarian") wrote the Lean sources of the `Gen` layer, the Gauss layer and `GaussBridge`, and the generator that derives four `Gen` files from the island. The `dbn` island itself was written in September and October 2026 with AI assistance under the author's direction, as recorded in [20, 21].

- *Audit and composition.* A second, separate agent session (the auditor, "bb") reviewed each layer as it arrived. It read the new sources in full and rebuilt the whole chain from source (OpenAI's library was used as built in the 7 October replay, with `Nonvanishing.lean` re-elaborated), and wrote its own check files, tampered statements and closure walks. It wrote the numerical checks of Section 5, wrote the two composition files and ran all the checker replays. Its reviews are labelled "AI review, NOT an operator read-back". They are the records cited as [3]. It also wrote the Comparator challenge files of Section 5.6 and read that run's seven logs in full, matched their md5s, and reproduced its solution export byte for byte.

- *Coordination and this paper.* A third session (the "lead") coordinated the work and wrote the publication plan. This paper was drafted by a further agent session, from the auditor's records and from sources it read. Every number in it is traceable to those records or sources, and anything it could not verify is marked as unresolved.

- *The author's role.* The author directed the programme: the author decided which results to pursue, approved running the work in OpenAI's environment and the publication of this result, chose the venue, and read and approved the final text.

The Lean kernel is the trusted component. No proof in this work is accepted on the word of an AI session. According to their READMEs, two of the checkers used in Section 5 were themselves written with AI under human supervision (Joachim Breitner, Lean FRO): `con-leche`, including its consistency proof, by Claude, and its Rust port `con-ron` by an AI system the README does not name [17, 18]; the five acceptances of Section 5 also include Lean's own kernel and `nanoda`, which do not share their code. The definitions and the statement are what a human reader must check, and Section 2 is written to make that check feasible. The definitions and the statement (Section 2) were read by the author before publication; no outside reader has checked them.

Acknowledgements

The bound is Freire, González-Riquelme, Muñoz, Poblete and Salinas's conditional observation; the engine is de Bruijn's theorem; the half-plane and its Lean proof are OpenAI's; the sign is Gauss's. Mathlib's L-function library made the bridge to $\Lambda(s, \chi)$ a statement about a shared definition rather than a private one. The independent checker `nanoda` is Chris Bailey's; `lean4export` and Comparator are from the Lean FRO (Comparator by Henrik Böving). The checker `con-leche` [17] was written by Claude, and its Rust port `con-ron` [18] by an AI system its README does not name, both under the supervision of Joachim Breitner at the Lean FRO. Cureton and Thorner's half-plane and Lean development make the companion section possible.

References

[1] J. Andrade, A. Chang, and S. J. Miller, *Newman's conjecture in various settings*, J. Number Theory **144** (2014), 70–91, Cited via [13].

[2] C. Bailey, *`nanoda_lib`: a type checker for Lean 4 export files*, https://github.com/ammkrn/nanoda_lib.

[3] bb (an AI auditor agent session, Claude), *Audit records for the $\Lambda_\chi \leq 9/32$ formalization*, Markdown reviews with md5-pinned inputs and outputs; see Appendix A, October 2026, Labelled "AI review, NOT an operator read-back". Archived in `provenance/` of the release repository.

[4] B. C. Berndt, R. J. Evans, and K. S. Williams, *Gauss and Jacobi sums*, Wiley, New York, 1998, Cited for the history of Gauss's sign theorem; not consulted directly.

[5] H. Böving, *`comparator`: a checker for formal challenges*, https://github.com/leanprover/comparator, Lean FRO; tag `v4.34.0`.

[6] J. Breitner, K. Morrison, H. Böving, et al., *`lean4export`: an exporter for Lean 4 environments*, https://github.com/leanprover/lean4export, Version 3.1.0 of the export format.

[7] M. Carneiro, *`lean4lean`: a Lean 4 kernel written in Lean 4*, https://github.com/digama0/lean4lean.

[8] D. Cureton and J. Thorner, *Report: A quasi-Riemann hypothesis for Dedekind zeta functions*, Hexagon preprint hexagon:2610.00200v1, October 2026, 12 October 2026. Version 1 states the half-plane $\operatorname{Re} s > 1 - 1/6000$.

[9] D. Cureton, J. Thorner, and Axiom Math, *`DedekindZetaZeroFree`: Lean formalization*, https://github.com/AxiomMath/DedekindZetaZeroFree, 2026, Commit `09afbb4ce657`. The first public commit `284348a4` states $\operatorname{Re} s > 5999/6000$; commit `201bc40` (13 October 2026) strengthens the theorem `main` to $\operatorname{Re} s > 31/32$ for every number field. Lean `v4.35.0-rc3`, Mathlib `0493595e`. Apache-2.0.

[10] N. G. de Bruijn, *The roots of trigonometric integrals*, Duke Math. J. **17** (1950), no. 3, 197–226.

[11] L. de Moura, *Lean now ships with verified kernels*, https://leodemoura.github.io/blog/2026-10-15-lean-now-ships-with-verified-kernels/, 2026, 15 October 2026.

[12] A. Dobner, *A proof of Newman's conjecture for the extended Selberg class*, Acta Arith. **201** (2021), no. 1, 29–62.

[13] R. Freire, C. González-Riquelme, C. Muñoz, F. Poblete, and V. Salinas, *Upper bounds on the de Bruijn–Newman constant for Dirichlet L-functions with primitive real-valued characters*, arXiv:2610.27343v1 [math.NT], October 2026, 19 October 2026.

[14] C. F. Gauss, *Summatio quarumdam serierum singularium*, Commentationes Societatis Regiae Scientiarum Gottingensis Recentiores **1** (1811).

[15] A. E. Ingham, *The distribution of prime numbers*, Cambridge Tracts in Mathematics and Mathematical Physics, vol. 30, Cambridge University Press, Cambridge, 1932.

[16] H. Ki, Y.-O. Kim, and J. Lee, *On the de Bruijn–Newman constant*, Adv. Math. **222** (2009), no. 1, 281–306.

[17] Lean FRO, *`con-leche`: an external checker for Lean, proven consistent*, https://github.com/leanprover/con-leche, 2026, Implemented and proven consistent by Claude under the supervision of J. Breitner (README).

[18] ———, *`con-ron`: a Rust port of `con-leche`*, https://github.com/leanprover/con-ron, 2026, Written by AI under the supervision of J. Breitner; refinement of con-leche proved with Aeneas (README).

[19] H. L. Montgomery and R. C. Vaughan, *Multiplicative number theory I: Classical theory*, Cambridge Studies in Advanced Mathematics, vol. 97, Cambridge University Press, Cambridge, 2006, Cited via [13] (Theorems 9.13, 9.16 and 10.8 there).

[20] P. W. Murphy, *A kernel-checked bound $\lambda \leq 9/32$ for the de Bruijn–Newman constant from OpenAI's quasi-Riemann hypothesis*, https://github.com/DrMurphyIsIn/qrh-debruijn-newman, 2026, doi:10.5281/zenodo.23269134. The `dbn` island used here is at commit `afd94c7`.

[21] ———, *Newman's conjecture, kernel-checked: $\lambda \geq 0$ for the de Bruijn–Newman constant in Lean 4 along Dobner's proof*, https://github.com/DrMurphyIsIn/newman-lambda-nonneg, 2026, doi:10.5281/zenodo.23287832 (concept DOI, all versions).

[22] C. M. Newman, *Fourier transforms with only real zeros*, Proc. Amer. Math. Soc. **61** (1976), no. 2, 245–251.

[23] C. M. Newman and W. Wu, *Constants of de Bruijn–Newman type in analytic number theory and statistical physics*, Bull. Amer. Math. Soc. (N.S.) **57** (2020), no. 4, 595–614, Cited via [13].

[24] OpenAI, *The quasi-Riemann hypothesis*, OpenAI Math Release preprint `OAI:The-Quasi-Riemann-Hypothesis-October-5-2026`, 2026, 5 October 2026; "written with human assistance". https://github.com/openai/math/blob/main/preprints/The-Quasi-Riemann-Hypothesis-October-5-2026/paper2.pdf.

[25] ———, *The quasi-Riemann hypothesis: A zero-free half-plane* $\mathrm{Re}(s) > 7/8$, OpenAI Math Release preprint `OAI:The-Quasi-Riemann-Hypothesis-September-30-2026`, 2026, 30 September 2026. https://github.com/openai/math/blob/main/preprints/The-Quasi-Riemann-Hypothesis-September-30-2026/paper.pdf.

[26] ———, *openai/math: Lean library accompanying the OpenAI math release*, https://github.com/openai/math, 2026, Commit `adc7f1241b42e322a6451854ab7e4b4c146bf78a`; Lean `v4.34.1`, Mathlib `d13f23b`. Theorem `OAI.DirichletCharacter.LFunction_ne_zero_of_seven_eighths_lt_re`, file `lean/OAI/NumberTheory/DirichletL/Nonvanishing.lean`; challenge `lean/ComparatorChallenges/DirichletSevenEighths.lean`. Apache-2.0.

[27] D. Platt and T. Trudgian, *The Riemann hypothesis is true up to* $3 \cdot 10^{12}$, Bull. London Math. Soc. **53** (2021), no. 3, 792–797.

[28] D. J. Platt, *Numerical computations concerning the GRH*, Math. Comp. **85** (2016), no. 302, 3009–3027, Cited via [13].

[29] D. H. J. Polymath, *Effective approximation of heat flow evolution of the Riemann ξ function, and a new upper bound for the de Bruijn–Newman constant*, Res. Math. Sci. **6** (2019), Paper 31, arXiv:1904.12438.

[30] B. Rodgers and T. Tao, *The de Bruijn–Newman constant is non-negative*, Forum Math. Pi **8** (2020), e6, arXiv:1801.05914.

[31] J. Stopple, *Notes on low discriminants and the generalized Newman conjecture*, Funct. Approx. Comment. Math. **51** (2014), no. 1, 23–41, Cited via [13].

[32] The mathlib Community, *The Lean mathematical library*, Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020), ACM, 2020, Revisions used here: `0493595e` and `d13f23b`, pp. 367–381.

A The auditor's records

The records cited as [3] are the following files. They are archived, each with its md5, in the `provenance/` directory of the release repository:

- `REVIEW_GEN_D1_D2a.md`: the generic de Bruijn library, and the theta kernel;
- `REVIEW_D2b_BRIDGE.md`: the Mellin bridge;
- `REVIEW_D2c_STRIP.md`: the zero strip;
- `REVIEW_D2bp_CHI4.md`: the theta functional equation, and χ_{-4};
- `NOTE_D2_GAUSS_SIGN.md`: Mathlib's root number, and the sign;
- `REPORT_OAI_PORT.md`: the port to OpenAI's environment, and provenance;
- `REVIEW_GAUSS_SIGN_AND_OAI.md`: Gauss's sign theorem, and the composition;
- `REVIEW_GAUSS_BRIDGE_ADD1.md` ("ADD1"): part (b), and the conjunction;
- `REVIEW_P1P6_BUNDLE.md`: including the literature status of the statement (its §6);
- `REVIEW_CTQ.md`, `REVIEW_INGHAM.md`, `REVIEW_CTFREE_KERNELS.md`: the companion results;
- `REVIEW_CT_RERUNS.md`: the capability-dropped Comparator re-runs of P2, Ingham and CT main;
- `COMPARATOR_RUN_SPEC.md`: the specification of the sandboxed run;

- `REVIEW_COMPARATOR_LAMBDA_CHI.md`: the verification of that run, with `REVIEW_COMPARATOR_LAMBDA_CHI_ADDENDUM1.md`, the inspection of its logs, and `REVIEW_COMPARATOR_LAMBDA_CHI_ADDENDUM2.md`, two precision corrections.