Calcul des zéros de ζ par la construction de Connes et al., d'autres exécutions du programme rust téléchargé en juillet 2026
Denise Vella-Chemla pilotant l'ia claude, 11 novembre 2026

On rappelle que les programmes en langage python des ia lumo et mistral pour le calcul des premiers zéros de la fonction ζ de Riemann ont été stockés à ces adresses :

https://denisevellachemla.eu/zeta-lumo-20261005.pdf
https://denisevellachemla.eu/zeta-mistral-20261005.pdf

La suite de cette note a été intégralement rédigée par l'ia claude, en réponse à des prompts de Denise Vella-Chemla.

Résumé

Ces notes consignent ce qui a été observé et dit au sujet du programme Rust d'Andrews (construction de Connes et al.) lancé avec $\lambda^2 = 250$, et au sujet de l'article Andrews-Connes-La Gatta[1]. Chaque affirmation est marquée : *(mesuré)* pour ce qui sort des exécutions, *(article)* pour ce que disent les articles, *(estimation)* pour ce qui est une extrapolation. Les trois exécutions complètes figurent en annexe.

1. Ce qui a été lancé

Programme : l'archive "ronnie-andrews-2-ccm-reproduction-and-convergence-main" téléchargée au départ (version initiale, voir la section **??**), compilée avec l'option `--features hp`, 16 fils de calcul (Rayon).

Paramètres communs : $\lambda^2 = 250$, 68 puissances de nombres premiers $k \leq 250$, précision 100 chiffres décimaux (397 bits).

La commande type est :

```
[fontsize\small]
cargo run --release --features hp -p ccm-reproduction -- run
  --lambda-sq 250 --n-modes n --precision-digits 100
  --display-digits 100 --top T
```

Exécutions : $(n, T) = (150, 100)$, qui s'est arrêtée avec le message "found only 84 positive roots" ; puis $(150, 84)$, $(400, 100)$ et $(500, 100)$. On note $L = \log \lambda^2 = 2 \log \lambda \approx 5{,}5215$.

2. Temps de calcul

exécution	τ	LU	état propre	racines	total
$n = 150$, premier calcul	11,6 s	0,7 s	35,4 s	–	–
$n = 150$, relance (caches)	0,6 s	–	0,05 s	3,5 s	5,8 s
$n = 400$	217,6 s	3,2 s	201,0 s	4,1 s	423,7 s
$n = 500$	201,2 s	4,8 s	273,0 s	5,7 s	480,9 s

1. https://arxiv.org/pdf/2611.13129.pdf du 9 novembre 2026.

(mesuré) : pour $n = 400$, 250 des 267 tables de quadrature de Gauss-Legendre ont été calculées ; pour $n = 500$, 267 des 367 tables venaient du cache (100 calculées), ce qui explique en partie que $n = 500$ n'ait pas eu un temps d'exécution plus long.

Une estimation faite en cours de route donnait 10 à 12 minutes pour $n = 500$ et 15 à 25 minutes pour $n = 600$ (croissance entre n^2 et n^3) *(estimation)* : la durée mesurée pour $n = 500$ est de 8 minutes, donc l'estimation était trop pessimiste. Le cas $n = 600$ n'a pas été lancé.

3. Pourquoi la première exécution s'arrête à 84 racines

Dans la construction, la plus haute fréquence de la base est $2\pi n/L$. Pour $n = 150$ elle vaut 170,7 ; la 84$^{\text{e}}$ racine trouvée est 169,912, juste en dessous. Pour $n = 400$ et $n = 500$ elle vaut 455,2 et 569,0.

Cette règle se retrouve ailleurs :

- *(article)* : dans l'article Andrews-Connes-La Gatta, à $\lambda^2 = 256$ et $n = 1500$, l'accord avec les zéros tombe à zéro vers $t \approx 1700$, pour $2\pi n/L = 1699{,}6$;
- *(mesuré)* : dans le programme en langage python de l'ia mistral à $\lambda^2 = 13$ et $n = 20$, $2\pi n/L = 49{,}0$ et le 10$^{\text{e}}$ zéro (49,77) est absent.

La limite à 84 vient donc de n (et non de λ^2 comme je (*on rappelle que c'est ici l'ia claude qui s'exprime*) l'avais écrit au départ) : le nombre de zéros accessibles est celui des zéros inférieurs à $2\pi n/L$.

4. Racines "en trop" et zéros manquants

4.1. Ce qui sort des trois exécutions

La colonne "matching digits" du programme compare la k-ième racine au k-ième zéro de ζ : dès qu'une racine est en trop, tous les rangs sont décalés et l'affichage indique 0,1 à 0,7 chiffre. Ce n'est pas la précision réelle. Par exemple, à $n = 150$ la racine 3 coïncide avec le zéro 1 (14,1347…) sur les 100 chiffres affichés, la racine 7 avec le zéro 2, etc. Le rang de 14,1347 est 3, 6 puis 5 pour $n = 150$, 400, 500.

Comparaison par la valeur (les racines ont été relevées à 5 décimales et rapprochées des zéros calculés par la bibliothèque `mpmath`, tolérance $2 \cdot 10^{-5}$) :

n	racines	dernière racine	vrais zéros ≤ dernière	trouvés	manquants
150	84	169,912	64	64	aucun
400	100	169,095	63	61	zéros 30 (101,318) et 31 (103,726)
500	100	165,537	61	60	zéro 12 (56,446)

Il y a donc 20, 39 et 40 racines qui ne sont pas des zéros de ζ ("fantômes") dans les 84, 100 et 100 premières racines. Passer de $n = 150$ à $n = 500$ n'a pas fait disparaître les fantômes : leur nombre a augmenté dans la même plage. *(mesuré)*

4.2. Les fantômes existent aussi dans le petit programme en langage python

(mesuré) : programme de l'ia mistral, $\lambda^2 = 13$, $n = 20$: racines trouvées 6,998, 11,849, 14,135, 18,189, 21,022, …, dont trois fantômes (6,998, 11,849, 18,189). Le premier fantôme vaut 7,264, 6,998, 6,974 pour $n = 16$, 20, 24 : il bouge un peu avec n, alors que les vrais zéros restent fixes à 10^{-6} près.

4.3. Un point à clarifier : la précision de 100 chiffres

(mesuré) : la plus petite valeur propre affichée vaut $\varepsilon_n = -9{,}78 \cdot 10^{-120}$, $7{,}02 \cdot 10^{-120}$ et $4{,}96 \cdot 10^{-120}$ pour $n = 150, 400, 500$. Avec 397 bits, le plancher d'arrondi est de l'ordre de $10^{-119{,}5}$: ces valeurs sont très probablement du bruit d'arrondi (le signe négatif à $n = 150$ va dans ce sens).

(article) : l'article Andrews-Connes-La Gatta donne $\varepsilon_n \approx 1{,}1 \cdot 10^{-1346}$ à $\lambda^2 = 250$, $n = 4000$; l'article d'Andrews (Zenodo) décrit, à sa section 4.8, que lorsque ε_n passe sous le plancher de précision, le vecteur propre calculé peut "tourner" dans un sous-espace numériquement plat. Je ne sais pas si les fantômes et les zéros manquants viennent de n trop petit, de ce plancher de précision, ou des deux.

5. Comment savoir qu'une racine est un vrai zéro

5.1. La question et ma réponse

Si le programme mélange des zéros de ζ et des racines en trop, on ne peut pas s'en remettre à une liste de référence dont la justesse ne serait pas elle-même démontrée. Le point est que la liste n'est pas nécessaire : un zéro est une propriété de ζ, que l'on teste directement.

Test d'une racine t_0 sur la droite critique. : on évalue $\zeta\left(\frac{1}{2} + it_0\right)$, ou la fonction Z de Hardy, réelle, avec $|Z(t)| = \left|\zeta\left(\frac{1}{2} + it\right)\right|$. En arithmétique d'intervalles, un changement de signe de Z prouve un zéro entre deux valeurs.

(mesuré) : sur les racines du programme de l'ia mistral ($\lambda^2 = 13$, $n = 20$) :

racine t_0	$\left\lvert\zeta\left(\frac{1}{2} + it_0\right)\right\rvert$
6,998254 (fantôme)	1,10
11,848708 (fantôme)	1,31
18,189141 (fantôme)	2,31
14,134725	$1{,}1 \cdot 10^{-7}$
21,02204	$4{,}1 \cdot 10^{-7}$
14,134725141734693790	$6{,}7 \cdot 10^{-16}$

Pour un vrai zéro, la valeur est proche de 0, l'écart venant de l'arrondi de t_0 ; pour un fantôme, ζ vaut de l'ordre de 1.

Et la liste d'Odlyzko ? : elle n'est qu'une table de résultats de calculs de ce type. Les zéros utilisés pour la comparaison ci-dessus sont ceux de `mpmath` (`zetazero`), qui les calcule à partir de ζ elle-même. Des vérifications rigoureuses en arithmétique d'intervalles existent pour des milliards de zéros.

5.2. Pourquoi $\frac{1}{2}$ et pas "un peu à gauche, un peu à droite"

Il y a deux questions distinctes.

- *cette racine est-elle un zéro sur la droite critique ?* On teste exactement $\frac{1}{2} + it_0$, puisque c'est ce qu'affirme le programme (ses racines sont réelles, donc prétendent être sur la droite critique).
- *n'y a-t-il pas de zéros ailleurs, et les a-t-on tous ?* On ne teste pas des points, on compte : ζ est holomorphe, donc sans saut brutal, et le principe de l'argument donne le nombre exact $N(T)$ de zéros dans le rectangle qui entoure la bande critique, quelle que soit leur abscisse. On le compare au nombre de changements de signe de Z sur la droite critique. S'ils sont égaux, il n'y a aucun zéro hors de la droite critique jusqu'à la hauteur T.

Un zéro hors de la droite critique, $\rho = \beta + it$, entraîne $1 - \overline{\rho} = 1 - \beta + it$ (par la symétrie de ζ), donc une paire de zéros à la même hauteur, de part et d'autre de $\frac{1}{2}$: il manquerait deux changements de signe et le décompte serait pris en défaut.

5.3. Ce que le programme démontre et ne démontre pas

(article) : les zéros de la fonction finie $t \mapsto \mathcal{M}\eta(it)$ sont réels dès que la plus petite valeur propre est simple et que son vecteur propre est pair (Théorème 7.1 de l'article Andrews-Connes-La Gatta). Que ces zéros convergent vers ceux de ζ n'est pas démontré ; la proposition 7.3 de cet article indique qu'une telle convergence (uniforme sur la bande), avec le théorème de Hurwitz, entraînerait l'hypothèse de Riemann, mais la remarque 7.4 précise que leurs résultats ne donnent pas de preuve (ils reposent sur la positivité de la forme de Weil). La comparaison avec ζ est donc la seule chose qui relie le calcul à ζ.

6. Ce que disent les articles

6.1. article de Ronnie Andrews, Alain Connes, Antonio La Gatta

(article)

- $\lambda^2 = 250$ (53 premiers $p < 250$), $n = 4000$ (aussi 3000 et 5000), état fondamental calculé avec environ 2900 chiffres décimaux ; pour situer les zéros, 1500 puis 1800 chiffres.
- les 1000 premiers zéros ont plus de 500 décimales correctes, le premier en a 1341, et les 2469 zéros de hauteur ≤ 3000 ont au moins 12 chiffres corrects. Un comptage certifié (arithmétique de boules, jusqu'à 7000 bits) montre que la fonction finie n'a aucun autre zéro, réel ou complexe, dans cette plage : pas de fantôme jusqu'à 3000.
- l'accord disparaît entre $t \approx 3050$ et 3130 (pour $n = 4000$, la fonction a 3907 zéros jusqu'à $2\pi n/L = 4551{,}8$, contre 4048 pour ζ).

- n doit dépasser environ $10\lambda^2$ pour la convergence des valeurs propres ; les calculs de $\lambda^2 = 200, 250, 300$ utilisent $n = 16\lambda^2$ (3200, 4000, 4800).
- règle de précision : environ $3{,}33\,(2|\log_{10}\varepsilon_\lambda(1)| + 40)$ bits ; avec $\varepsilon \approx 10^{-1346}$ cela fait environ 2700 chiffres (ils en utilisent 2900).
- balayage indépendant des changements de signe jusqu'à $t = 6000$, puis géométrique jusqu'à 10^7, pour $\lambda^2 = 250$, $n = 4000$.
- les calculs ont tourné sur un cluster équivalent à une centaine de GPU A100 ; à $\lambda^2 = 150$, $n = 2500$, la mémoire disponible à 5500 bits a été dépassée et les valeurs propres ont été calculées par lots.

6.2. article de Ronnie Andrews

(article) :[2] règle empirique $n/\sqrt{\lambda^2} \approx 28$ pour obtenir les premiers zéros avec beaucoup de chiffres ; exemples : $\lambda^2 = 13$, $n = 120$: 55,764 chiffres sur le premier zéro ; $\lambda^2 = 100$, $n = 500$: 460 chiffres (premier zéro) et 422 (vingtième) ; $\lambda^2 = 1000$, $n = 800$: 999 chiffres (précision 1000). Le nombre de chiffres correspond à $-\log_{10}\varepsilon_n$. Le calcul à froid de $(\lambda^2 = 1000, n = 800)$ a pris 3 à 4 heures sur un nœud de 512 cœurs ; avec les caches publics, environ 13 minutes. Ce critère $n \approx 28\sqrt{\lambda^2}$ vise les premiers zéros, alors que $n \gtrsim 10\lambda^2$ à $16\lambda^2$ est celui de l'article de novembre pour tous les zéros sans fantôme : ils ne se contredisent pas.

6.3. article de Akiva Groskin (https://arxiv.org/pdf/2607.02828.pdf)

(article) : il établit une identité exacte entre la forme de Weil tronquée et une somme sur les zéros de ζ, et des critères de certification pour les valeurs propres. Il ne change rien au calcul des zéros décrit ici.

7. Jusqu'où aller pour $\lambda^2 = 250$ et $\lambda^2 = 300$

$\lambda^2 = 250$. *(article)* : maximum "propre" : 2469 zéros (jusqu'à la hauteur 3000), 2596 si l'on prend l'accord jusqu'à son effondrement (figure 7 de l'article).

$\lambda^2 = 300$. *(article)* : l'article ne donne aucun nombre de zéros : seul le spectre est calculé ($n = 4800$, $\varepsilon_\lambda(1) \sim 10^{-1618}$). *(estimation)* À $\lambda^2 = 250$ l'effondrement a lieu vers $t \approx 4\pi\lambda^2 \approx 3140$ (observé : 3050–3130). Extrapolé à 300, cela donnerait $t \approx 3770$, soit environ 3200 zéros. Non vérifié.

Faisabilité sur une machine à 16 cœurs. (estimation) : non.

- temps : à partir des mesures, une croissance entre n^2 et n^3 donne 5 à 50 heures pour $n = 3000$ à 100 chiffres, avant de multiplier par le coût de 2900 chiffres.
- mémoire : une matrice 3001×3001 à 2900 chiffres occupe de l'ordre de 10 Go, et la factorisation en demande plusieurs copies.
- précision : la règle ci-dessus demande environ 2700 chiffres.

2. Voir https://zenodo.org/records/20427500.

8. La version du code

Les exécutions ci-dessus ont été faites avec la *version initiale* du programme (l'archive téléchargée au départ). Le dépôt actuel est le suivant :

https://github.com/TeamXcelerator/ccm-reproduction-and-convergence

D'après l'article d'Andrews, il s'appuie sur une bibliothèque partagée (Xcelerator Toolkit) et sur des caches publics (nœuds de Gauss-Legendre, matrices τ, vecteurs propres de Weil) récupérés automatiquement. L'article Andrews-Connes-La Gatta ne mentionne aucune modification du code par rapport à la version initiale, et je n'ai pas comparé les deux versions : rien ici ne dit ce que donnerait la version actuelle.

9. Les programmes en langage python (par les ia lumo et mistral, 5 octobre 2026)

mistral (programme en langage python pur, élimination de Jacobi). *(mesuré)* : $\lambda^2 = 13$, $n = 20$, moins d'une seconde : les 9 premiers zéros sont retrouvés avec des erreurs de $2 \cdot 10^{-15}$ (zéro 1) à $4 \cdot 10^{-9}$ (zéro 9) ; le zéro 10 est absent (voir section 3). À $n = 40$, 14 zéros sur 30 seulement sont reproduits à 10^{-6} près : la valeur propre minimale affichée est de l'ordre de 10^{-16}, soit le plancher de la double précision, alors que la vraie valeur est bien plus petite. Il faut donc de la haute précision (ce que fait le Rust) dès que n grandit.

lumo (numpy). *(mesuré)* : le programme affiche 10^{-14}, 3,71, 6,93 au lieu de 14,13. Cause (lecture du code) : le terme archimédien est absent (la matrice vaut "pôle moins premiers"), seuls les premiers jusqu'à 19 sont gardés, et les puissances de premiers $(4, 8, 9, 16, \dots)$ sont omises.

Passage à $\lambda^2 = 250$ en langage python. (estimation) : avec `mpmath`, la factorisation d'une matrice de dimension 801 à 100 chiffres prendrait des heures, voire des jours, contre 7 minutes pour le Rust parallèle ; non mesuré.

10. Conclusion

Les trois exécutions[3] donnent des racines dont une partie sont de vrais zéros de ζ (reproduits sur les 100 chiffres affichés pour les plus petits), mêlés à des racines en trop et, à $n = 400$ et 500, à quelques zéros manquants. Pour obtenir la liste complète sans fantôme, l'article indique n de l'ordre de $10\lambda^2$ à $16\lambda^2$ et plusieurs milliers de chiffres de précision, ce qui dépasse une machine individuelle. La décision est d'arrêter ici les tests numériques.

3. Voir le fichier https://denisevellachemla.eu/relancer-la-machine-rust-en-novembre-2026.pdf.