Pourquoi $\lambda^2 = 100$ peut donner de moins bons résultats que $\lambda^2 = 5$ dans la construction numérique du programme python inspiré des programmes Rust pour le calcul des parties imaginaires des zéros de zêta

Denise Vella-Chemla, 22 juillet 2026

1. Résumé de l'observation

Dans le programme Python actuellement utilisé, on observe un phénomène qui peut sembler paradoxal :

- pour une valeur relativement petite de λ^2, par exemple $\lambda^2 = 5$, les zéros calculés peuvent sembler suivre assez correctement les premiers zéros non triviaux de la fonction zêta ;
- pour $\lambda^2 = 100$, on observe au contraire de nombreux zéros supplémentaires, ou "zéros fantômes", qui s'intercalent entre les zéros attendus.

La conclusion importante est la suivante :

> **Ce phénomène ne signifie pas nécessairement que la construction CCM devient mathématiquement moins bonne lorsque λ augmente. Il indique d'abord que le calcul numérique actuel n'est pas dans un régime où l'augmentation de λ peut être interprétée naïvement comme une amélioration.**

Dans le programme utilisé, plusieurs mécanismes peuvent produire précisément ce comportement.

2. Le point fondamental : augmenter λ ne suffit pas

La construction CCM dépend de plusieurs paramètres qui doivent être considérés ensemble :

λ, N, précision numérique, résolution de l'intégration, méthode de recherche des zéros.

Dans le programme de l'ia chatgpt, on modifie principalement λ^2, mais on laisse généralement inchangés les autres paramètres, notamment :

$$N = 10$$

et une arithmétique en double précision.

Or la dimension de la matrice est

$$2N + 1 = 21.$$

Il est donc possible que le régime

$$\lambda^2 = 5$$

soit numériquement relativement favorable, tandis que

$$\lambda^2 = 100$$

demande une résolution et une précision beaucoup plus importantes.

Il faut donc distinguer :

$$\boxed{\text{augmentation de } \lambda}$$

de

$$\boxed{\text{amélioration de l'approximation par le programme}}.$$

Ces deux phénomènes ne sont pas automatiquement équivalents dans une implémentation numérique finie.

3. Première cause probable : le nombre de modes N devient insuffisant

La construction utilise une troncature dans une base de Fourier :

$$\{-N, \dots, N\}.$$

La matrice a donc la dimension

$$2N + 1.$$

Dans le programme pédagogique, avec $N = 10$, on ne conserve que 21 modes.

Lorsque λ augmente, la structure de l'opérateur et la fonction ξ_λ deviennent plus exigeantes à représenter dans cette base finie. En particulier, le vecteur propre associé à la plus petite valeur propre peut nécessiter davantage de modes pour être représenté fidèlement.

Le problème général est donc :

$$\text{approximation exacte} \quad \longrightarrow \quad \text{projection sur un espace de dimension finie.}$$

Si N reste fixe tandis que λ augmente, on peut entrer dans un régime où

$$N \ll N_{\text{nécessaire}}(\lambda).$$

Dans ce cas, le calcul ne décrit plus correctement l'opérateur continu.

Les erreurs de troncature peuvent alors produire des oscillations supplémentaires dans les fonctions construites à partir de ξ_λ.

Ces oscillations supplémentaires peuvent à leur tour créer des changements de signe artificiels.

Or le programme cherche les zéros en étudiant les changements de signe d'une fonction échantillonnée. Il peut donc interpréter des oscillations numériques comme des zéros supplémentaires.

Cela fournit une première explication naturelle des "zéros fantômes".

4. Deuxième cause : le phénomène de type Nyquist

La fonction calculée sur la droite critique est évaluée sur une grille de valeurs de t :

$$s = \frac{1}{2} + it.$$

Supposons que la fonction calculée soit

$$F(t) = F_{\text{exact}}(t) + E(t),$$

où $E(t)$ est l'erreur numérique.

Un zéro réel de F_{exact} est un changement de signe authentique.

Mais si l'erreur $E(t)$ devient suffisamment grande ou suffisamment oscillante, on peut avoir :

$$F(t_j)F(t_{j+1}) < 0$$

alors qu'il n'existe aucun zéro mathématique pertinent entre t_j et t_{j+1}.

Le programme détecte alors un zéro.

Le phénomène est particulièrement dangereux lorsque la fonction numérique contient des composantes à fréquence élevée :

$$F_{\text{num}}(t) = F_{\text{phys}}(t) + A\sin(\omega t),$$

avec ω grand.

Même si le terme parasite $A\sin(\omega t)$ est petit en amplitude, il peut créer de nombreux changements de signe si la fonction principale est proche de zéro.

Ainsi, une augmentation de λ peut produire une fonction plus oscillante, alors que la grille d'échantillonnage reste inchangée.

Le programme voit alors davantage de zéros, mais une partie d'entre eux peut être due à une résolution insuffisante.

5. Troisième cause : les intégrales deviennent plus difficiles

La matrice CCM contient des coefficients construits à partir d'intégrales archimédiennes et de termes oscillants.

Schématiquement, on rencontre des expressions de la forme

$$\int_0^L x\,\rho(x)\cos\left(\frac{2\pi n x}{L}\right)\,dx.$$

Lorsque les paramètres varient, ces intégrales peuvent devenir plus oscillantes.

Une quadrature numérique avec un nombre fixe de points peut alors être excellente dans un cas et insuffisante dans un autre.

Il faut distinguer :

erreur de quadrature

et

erreur de troncature.

Les deux peuvent se cumuler.

Avec $\lambda^2 = 5$, l'erreur totale peut rester suffisamment faible pour que les premiers zéros soient correctement visibles.

Avec $\lambda^2 = 100$, la même configuration numérique peut produire une erreur plus importante dans les coefficients de la matrice.

Le vecteur propre obtenu est alors le vecteur propre d'une matrice légèrement erronée :

$$M_{\text{num}} = M_{\text{exact}} + \Delta M.$$

Si la plus petite valeur propre est très petite, le vecteur propre associé peut être particulièrement sensible à ΔM.

6. Quatrième cause : la petite valeur propre et la sensibilité du vecteur propre

Le programme cherche le vecteur propre associé à la plus petite valeur propre de la matrice.

Dans le cas idéal, on souhaite obtenir une valeur

$$\varepsilon_N \approx 0.$$

Mais en calcul numérique, on a en réalité

$$\varepsilon_N = \varepsilon_N^{\text{exact}} + \delta\varepsilon_N.$$

Si plusieurs valeurs propres sont proches, une petite perturbation de la matrice peut modifier sensiblement le vecteur propre.

C'est un phénomène classique de théorie spectrale numérique.

Une perturbation

$$M \mapsto M + \Delta M$$

peut produire une variation importante de l'eigenvecteur si le gap spectral

$$\text{gap} = \lambda_2 - \lambda_1$$

est petit.

Il ne suffit donc pas de regarder uniquement la valeur de la plus petite valeur propre.

Il faut aussi examiner :

$$\lambda_1, \qquad \lambda_2, \qquad \lambda_2 - \lambda_1.$$

Un diagnostic essentiel serait donc d'afficher les premières valeurs propres :

$$\lambda_1, \lambda_2, \lambda_3, \ldots$$

pour chaque valeur de λ^2.

Si le gap se réduit fortement lorsque λ^2 augmente, le vecteur propre ξ_λ devient naturellement plus sensible aux erreurs numériques.

7. Cinquième cause : l'extraction des zéros est particulièrement fragile

Dans la version actuelle du programme, les zéros sont recherchés à partir d'une représentation numérique de la fonction construite à partir de ξ_λ.

Une méthode typique est :

1. évaluer la fonction sur une grille ;
2. détecter les intervalles où le signe change ;
3. appliquer une méthode de recherche de racine dans ces intervalles.

Cette méthode est raisonnable pour une première exploration, mais elle ne permet pas de distinguer automatiquement :

$$\text{zéro structurel}$$

de

$$\text{zéro créé par une oscillation numérique.}$$

Il faudrait notamment vérifier chaque zéro candidat à plusieurs résolutions :

$$\Delta t, \qquad \frac{\Delta t}{2}, \qquad \frac{\Delta t}{4}.$$

Un zéro authentique devrait être stable :

$$t_{\text{zero}}(\Delta t) \approx t_{\text{zero}}(\Delta t/2) \approx t_{\text{zero}}(\Delta t/4).$$

Un zéro fantôme a souvent un comportement différent :

$$t_{\text{ghost}}(\Delta t) \not\approx t_{\text{ghost}}(\Delta t/2).$$

Il peut disparaître, se déplacer fortement ou être remplacé par plusieurs zéros.

8. Sixième cause : la comparaison avec les zéros de Riemann peut être trompeuse

Il faut également être très prudent avec l'indexation.

Les zéros non triviaux de la fonction zêta sont

$$\frac{1}{2} + i\gamma_1, \quad \frac{1}{2} + i\gamma_2, \quad \frac{1}{2} + i\gamma_3, \dots$$

avec

$$\gamma_1 \approx 14.134725, \qquad \gamma_2 \approx 21.022040, \qquad \gamma_3 \approx 25.010858, \dots$$

Si la fonction numérique calculée possède des zéros supplémentaires, il ne faut pas comparer naïvement :

$$\text{zéro calculé numéro } k \quad \text{avec} \quad \gamma_k.$$

Il faut plutôt effectuer une comparaison par proximité :

$$\min_j |t_{\text{calculé}} - \gamma_j|.$$

Sinon, un seul zéro fantôme inséré au début décale toute la correspondance :

zéro calculé	zéro de Riemann correspondant
fantôme	γ_1
γ_1	γ_2
γ_2	γ_3
$\vdots$	$\vdots$

La comparaison par rang devient alors catastrophique même si certains zéros calculés sont individuellement très proches des zéros de Riemann.

9. Pourquoi $\lambda^2 = 5$ peut donc sembler meilleur

Le phénomène observé peut être résumé ainsi :

$$\boxed{\lambda^2 = 5 \quad \text{peut être un régime numériquement favorable}}$$

alors que

$$\boxed{\lambda^2 = 100,\ N = 10,\ \text{même quadrature et même précision}}$$

peut être un régime sous-résolu.

Ce n'est pas contradictoire.

Une approximation asymptotique peut être théoriquement meilleure lorsque λ augmente, tout en étant numériquement moins bonne si les paramètres de calcul ne sont pas augmentés simultanément.

On peut représenter la situation par :

qualité totale = qualité théorique−erreur de troncature−erreur de quadrature−erreur d'arithmétique−erreur

La théorie peut améliorer le premier terme tandis que les erreurs numériques augmentent plus rapidement.

10. Le test expérimental le plus important

Pour vérifier cette explication, il faudrait effectuer une expérience en maintenant $\lambda^2 = 100$ mais en augmentant progressivement N :

$$N = 10,\ 20,\ 40,\ 80,\ 120.$$

Pour chaque valeur, il faut mesurer :

- le nombre de zéros trouvés ;
- la position de chaque zéro ;
- la stabilité de ces positions ;
- la plus petite valeur propre ;
- le gap spectral ;
- l'erreur par rapport aux zéros de Riemann.

Le tableau expérimental devrait ressembler à :

N	ε_N	nombre de zéros	zéros fantômes	erreur
10	$\cdots$	$\cdots$	$\cdots$	$\cdots$
20	$\cdots$	$\cdots$	$\cdots$	$\cdots$
40	$\cdots$	$\cdots$	$\cdots$	$\cdots$
80	$\cdots$	$\cdots$	$\cdots$	$\cdots$
120	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Si les zéros fantômes disparaissent lorsque N augmente, ils sont très probablement liés à la troncature.

Si, au contraire, ils restent aux mêmes positions avec une grande stabilité, il faut alors chercher une autre explication : structure réelle de la fonction calculée, mauvais choix de fonction, normalisation incorrecte ou différence avec la fonction CCM exacte.

11. Conclusion

L'observation que

$$\lambda^2 = 100$$

donne davantage de zéros parasites que

$$\lambda^2 = 5$$

est donc parfaitement plausible dans le programme actuel.

L'explication la plus probable est une combinaison de :

troncature insuffisante
+
quadrature insuffisante
+
double précision
+
recherche de zéros par changements de signe

et éventuellement d'une mauvaise correspondance par rang lors de la comparaison avec les zéros de Riemann.

Le test décisif n'est donc pas simplement :

$$\lambda^2 = 5 \quad \text{versus} \quad \lambda^2 = 100.$$

Il faut comparer :

$$(\lambda^2, N, \text{précision}, \text{résolution})$$

dans un régime où les paramètres numériques sont suffisamment adaptés à λ.

La prochaine étape la plus informative serait donc de modifier le programme pour produire automatiquement, pour une même valeur de λ^2 :

$$N = 10, 20, 40, 80, 120,$$

puis de suivre la stabilité de chaque zéro.

C'est cette expérience qui permettra de déterminer si les "zéros fantômes" sont des artefacts numériques ou s'ils proviennent d'une différence structurelle entre la fonction construite par notre programme et la fonction effectivement étudiée dans les articles de Connes, Consani et Moscovici.