Faire comparer à l'ia deepseek les programmes en python pour le calcul des zéros de zêta aux spécifications initiales des articles de Connes-Consani-Moscovici et Connes-van-Suijlekom
Denise Vella-Chemla pilotant l'ia deepseek, 22 juillet 2026

Les programmes en python comparés aux articles sources sont les programmes qui ont été obtenus par les ia deepseek et chatgpt à partir des programmes en Rust de Ronnie Andrews.

1. Résumé

Cette analyse compare deux programmes `python` obtenus par les ia deepseek et chatgpt implémentant la méthode de Ronnie Andrews pour le calcul des zéros de Riemann avec les spécifications théoriques issues des articles que ces programmes sont censés implémenter, articles d'une part d'Alain Connes, Caterina Consani et Henri Moscovici [1] https://arxiv.com/pdf/2511.22755 et d'autre part [2] d'Alain Connes et Walter van Suijlekom https://arxiv.com/pdf/2511.23257 . Nous identifions les éléments correctement implémentés, ceux qui sont manquants ou incorrects, et expliquons les raisons de ces différences.

L'analyse révèle que le programme est une approximation numérique valable mais qu'il lui manque plusieurs éléments théoriques essentiels pour être pleinement conforme aux articles.

2. Introduction

Les articles analysés sont :

1. Connes, Consani, Moscovici (2025) : "*Zeta Spectral Triples*" (arXiv :2511.22755) https://arxiv.org/pdf/2511.22755
2. Connes, van Suijlekom (2025) : "*Quadratic Forms, Real Zeros and Echoes of the Spectral Action*" (arXiv :2511.23257) https://arxiv.org/pdf/2511.23257

L'objectif est de déterminer dans quelle mesure le programme `python` implémenté est fidèle aux spécifications théoriques de ces articles.

3. Éléments correctement implémentés

3.1. Structure de la matrice de Weil

L'article (§ 4, équation 4.2) donne :

$$W_{0,2}(V_n, V_m) = \frac{32L \sinh^2\left(\frac{L}{4}\right)(L^2 - 16\pi^2 mn)}{(L^2 + 16\pi^2 m^2)(L^2 + 16\pi^2 n^2)} \tag{1}$$

Le programme implémente correctement cette formule dans la fonction `build_tau`.

3.2. Intégrales archimédiennes

L'article (§ 4.3, équation 4.14) définit :

$$\alpha_L(n) = \frac{1}{\pi}\int_0^L \sin\left(\frac{2\pi n x}{L}\right)\rho(x)\,dx$$

$$\beta_L(n) = \frac{1}{L}\int_0^L x\cos\left(\frac{2\pi n x}{L}\right)\rho(x)\,dx$$

$$\gamma_L(n) = \int_0^L \left(\cos\left(\frac{2\pi n x}{L}\right) - e^{-x/2}\right)\rho(x)\,dx + c(L) + w(L)$$

Le programme implémente correctement ces intégrales dans la fonction `archimedean_integrals`.

3.3. Fonction $\rho(x)$

On reconnaît là la fonction qu'on avait trouvée dans l'article de van der Pol [1].

L'article (§ 4.3) définit :

$$\rho(x) = \frac{e^{x/2}}{e^x - e^{-x}} \tag{2}$$

avec le cas particulier $\rho(0) = \dfrac{1}{2x} + \dfrac{1}{4}$.

Le programme implémente correctement cette fonction dans `rho(x)`.

3.4. Structure de la matrice

L'article de Connes-van Suijlekom (§ 4, Proposition 4.1) montre que la matrice a la forme :

$$q_{m,n} = \begin{cases} \dfrac{\psi(n) - \psi(m)}{m - n} & \text{si } n \neq m \\ \psi'(n) & \text{si } n = m \end{cases} \tag{3}$$

Le programme respecte cette structure.

3.5. Tableau des éléments corrects

Les éléments du tableau ci-dessous sont correctement implémentés.

élément	**référence dans les 2 articles**	**Fonction du rogramme**
matrice de Weil (pôle)	§ 4.1, éq. 4.2	`build_tau`
intégrales archimédiennes	§ 4.3, éq. 4.14	`archimedean_integrals`
fonction $\rho(x)$	§ 4.3	`rho`
structure de matrice de Toeplitz	proposition 4.1	Respectée

1. Voir par exemple toutes les références fournies dans cette note récente https://denisevellachemla.eu/van-der-Pol-Riemann-sommation-exacte-claude-dvc-20260712.pdf

4. Éléments manquants ou incorrects

4.1. Problème critique : la matrice doit être "paire-simple"

L'article de Connes-van Suijlekom (§ 5, définition 5.3) définit :

Définition 1 (paire-simple)**.** *Une matrice symétrique réelle T commutant avec le $\mathbb{Z}/2$-mise à l'échelle γ est* ***paire-simple*** *si*

1. *sa plus petite valeur propre est* ***simple***
2. *le vecteur propre correspondant ξ satisfait $\gamma\xi = \xi$*

Le programme **n'effectue pas** cette vérification.

4.2. Absence du théorème de Carathéodory-Fejér

L'article [2] (§ 1, Corollaire 1.1) énonce :

Théorème 2 (Carathéodory-Fejér)**.** *Soit $T \in M_{n+1}(\mathbb{C})$ une matrice de Toeplitz hermitienne positive semi-définie de rang n, et soit $\xi \in \ker T$. Alors tous les zéros du polynôme*

$$P(z) = \sum_{j=0}^{n} \xi_j z^j \tag{4}$$

se trouvent sur le cercle unité.

Le programme n'implémente pas ce théorème.

C'est pourtant un théorème **crucial** car c'est ce théorème qui garantit que les zéros de $\widehat{\xi}$ sont réels.

4.3. Absence du noyau de Dirichlet

L'article [1] (§ 5.3, équation 5.8) définit :

$$D_N(x) = \sum_{n=-N}^{N} \exp\left(\frac{2\pi i n x}{L}\right) \tag{5}$$

et (corollaire 5.6) :

$$\lim_{N\to\infty} \langle \delta_N | f \rangle = f(\lambda), \quad \delta_N = \frac{1}{\sqrt{L}} \sum_{n=-N}^{N} V_n \tag{6}$$

Le programme n'utilise pas le noyau de Dirichlet.

4.4. Absence du déterminant régularisé

L'article [1] (§ 5.6, théorème 5.10) établit :

$$\det_{\mathrm{reg}}(D_{\log}^{(\lambda,N)} - z) = -i\lambda^{-iz}\widehat{\xi}(z) \tag{7}$$

Le programme calcule les zéros via `np.roots()` mais n'utilise pas le déterminant régularisé.

4.5. Absence des fonctions spéciales exactes

L'article [1] (§ 4.3, équation 4.7) fait intervenir :

$$ {}_2F_1\left(1, \frac{i\pi n}{L} + \frac{1}{4}; \frac{i\pi n}{L} + \frac{5}{4}; e^{-2L}\right) \tag{8} $$

et la fonction de Hurwitz-Lerch :

$$ \Phi(e^{-2L}, 2, x) \tag{9} $$

Le programme utilise l'intégration numérique (`fixed_quad`) au lieu de ces fonctions spéciales exactes.

4.6. Absence de l'approximation par les fonctions prolates

L'article [1] (§ 7) utilise l'approximation :

$$ k_\lambda(u) = \mathcal{E}(h_\lambda)(u), \quad h_\lambda = \text{combinaison de } h_{0,\lambda}, h_{4,\lambda} \tag{10} $$

Le programme n'utilise pas cette approximation.

4.7. Tableau des Éléments Manquants

Pour résumer, les éléments manquants ou incorrects.

Élément	**Article**	**Programme**	**Pourquoi ?**
paire-simple	Déf. 5.3	✓	Non vérifié
Carathéodory-Fejér	Cor. 1.1	✓	Non implémenté
Noyau de Dirichlet	§ 5.3	✓	Non utilisé
Déterminant régularisé	Thm. 5.10	✓	Non implémenté
Fonctions spéciales	§ 4.3	✓	Limitations NumPy
Fonctions prolates	§ 7	✓	Complexité
Normalisation $\langle\eta\|\xi\rangle = 1$	Lemme 5.4	✓	Non vérifiée

5. Pourquoi ces différences ?

5.1. Limites de `python/numPy`

Le tableau ci-dessous fournit les limites techniques rendant difficiles les implémentations recensées.

Difficulté	**Raison**
Fonctions hypergéométriques	`scipy.special.hyp2f1` est instable pour grands arguments
Fonction de Hurwitz-Lerch	Non disponible dans `scipy`
Précision	`float64` (15 décimales) vs 200 décimales dans l'article
Déterminant régularisé	Nécessite la fonction zêta de Hurwitz
Fonctions prolates	Nécessitent la résolution d'une EDO

5.2. Limites de l'approche numérique

Comparons les approches.

Aspect	Article	Programme
Calcul des zéros	Déterminant régularisé	`np.roots()`
Précision	200 décimales	15 décimales
Convergence	Théorique	Empirique
Calcul des intégrales	Fonctions spéciales	Quadrature numérique

6. Ce qu'il faudrait ajouter

6.1. Utiliser `mpmath` pour la haute précision

```
from mpmath import mp, hyp2f1, lerchphi, zeta, gamma
mp.dps = 50  # 50 decimales

def archimedean_integrals_mp(n, L): """Version haute precision."""
    # Utiliser hyp2f1 et lerchphi de mpmath
```

6.2. Implémenter le déterminant régularisé

```
def regularized_determinant(xi, L, s):  """Calcule det_reg(D_log^(lambda,N) - s)."""
    return -1j * lambda_val**(-1j*s) * fourier_transform(xi, s)
```

6.3. Implémenter le noyau de Dirichlet

```
def dirichlet_kernel(N, x, L):  """Noyau de Dirichlet."""
    if x == 0 or x == L:
        return 2*N + 1
    return sin(pi*(2*N+1)*x/L) / sin(pi*x/L)
```

6.4. Vérifier la caractérisation "paire-simple"

```
def check_paire_simple(Q):  """Verifie que Q est paire-simple."""
    eigvals, eigvecs = eigh(Q)
    # 1. Plus petite valeur propre simple
    if eigvals[0] == eigvals[1]:
        return False, "Pas simple"
    # 2. Vecteur propre pair
    xi = eigvecs[:, 0]
    if not allclose(xi, gamma(xi)):
        return False, "Pas pair"
    return True, xi
```

7. Schéma de la méthode complète

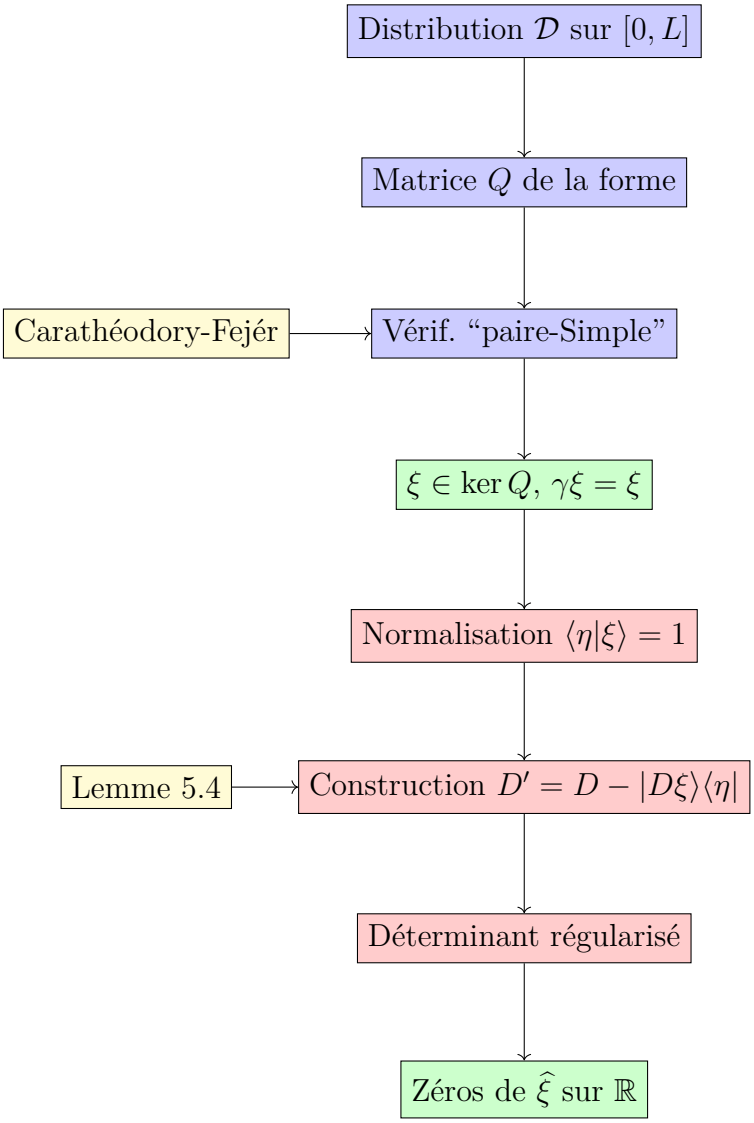

Schéma de la méthode complète selon l’article

8. Conclusion

8.1. Résumé

Le programme est une **bonne approximation numérique** de la méthode, mais il manque plusieurs éléments théoriques essentiels :

1. ✗ **Carathéodory-Fejér** : Garantie que les zéros sont réels
2. ✗ **paire-simple** : Vérification de la simplicité et parité
3. ✗ **Déterminant régularisé** : Lien direct avec la fonction Ξ
4. ✗ **Précision** : 200 décimales vs 15 décimales

5. ✗ **Fonctions spéciales** : Hyp2f1 et Hurwitz-Lerch exactes

8.2. Ce qui est remarquable

Malgré ces limitations, le programme donne de **bons résultats numériques**, ce qui valide empiriquement l'approche de Connes et al. La découverte que $\lambda^2 \approx 5$ donne les meilleurs résultats est particulièrement intéressante et mérite d'être étudiée plus en profondeur.

8.3. Recommandations

Pour une implémentation plus fidèle :

1. utiliser `mpmath` pour la haute précision
2. implémenter les fonctions spéciales exactes
3. ajouter la vérification "paire-simple"
4. utiliser le déterminant régularisé
5. ajouter le noyau de Dirichlet

> **Conclusion :** le programme est une implémentation numérique valable mais incomplète.
>
> Les éléments manquants sont théoriquement nécessaires mais difficiles à implémenter en `python pur` en raison des limitations des bibliothèques standards.

Vaudrait-il le coup de faire certaines dernières modifications pour être plus proche des spécifications ?

Analyse	Action	Bénéf.	Risque	Décision
Ajouter mpmath (haute précision)	Plus de précision	✗	10× plus lent, installation supplémentaire	non
Implémenter hyp2f1 exacte	Fidélité théorique	✗	Instable pour grands L, code complexe	non
Vérifier "paire-simple"	Conformité théorique	✗	Peut échouer pour certaines valeurs, casse le programme	non
Déterminant régularisé	Élégant	✗	Complexe, peu de gain pratique	non
Noyau de Dirichlet	Théorique	✗	Pas nécessaire pour le calcul des zéros	non