

Des cercles concentriques aux cercles de logarithme : un poids de Gauss pour les décomposants de Goldbach

Denise Vella-Chemla pilotant l'ia claudé, 10 octobre 2026

Suite à l'écoute de la conférence d'Alain Connes à l'Ircam le 3 octobre 2026, où il parle du site musical, je fournis à l'ia claudé les documents suivants :

<https://denisevellachemla.eu/correspondance-2-3-claude-dvc-20261004.pdf>
<https://denisevellachemla.eu/concentriques40.pdf>
<https://denisevellachemla.eu/concentriques98.pdf>
<https://denisevellachemla.eu/concentriques400.pdf> .

La suite de cette note a été intégralement rédigée par l'ia claudé.

1. Les cercles concentriques

Soit n pair. Un entier $x \leq n$, $x > 1$, est premier si et seulement s'il n'est divisible par aucun premier $p \leq \sqrt{n}$. Sur le cercle associé à p on marque les multiples de p (angle $2\pi x/n$) ; les points hors de tous les cercles sont les premiers. Les décomposants de Goldbach de n sont les couples $(x, n-x)$ où x et $n-x$ sont tous deux hors des cercles ; l'involution $x \mapsto n-x$ est une réflexion, d'où les traits verticaux verts des figures pour $n = 40, 98, 400$. Notons $G(n) = \#\{p \leq n/2 : p, n-p \text{ premiers}\}$.

2. Du cercle linéaire au cercle de logarithme

En angle $2\pi \ln x / \ln n$, multiplier par p est une rotation d'angle $2\pi \ln p / \ln n$ (cas A), mais l'involution devient $t \mapsto \ln(n-e^t)$, qui n'est plus un mouvement rigide. Les entiers de $[n/2, n]$ se tassent dans un arc de proportion $\ln 2 / \ln n$. Le logarithme convient à la structure multiplicative, le cercle linéaire à la structure additive. Le logarithme est donc conservé comme poids (cas B) : $u(x) = 1/\ln x$.

3. Trois prédictions

$$P_1(n) = \sum_{x=2}^{n/2} \frac{1}{\ln x \ln(n-x)}, \quad (1)$$

$$P_2(n) = 2C_2 \mathfrak{S}(n) P_1(n), \quad \mathfrak{S}(n) = \prod_{\substack{p|n \\ p>2}} \frac{p-1}{p-2}, \quad C_2 = \prod_{p>2} \left(1 - \frac{1}{(p-1)^2}\right), \quad (2)$$

$$L(n) = \frac{n}{2} \cdot \frac{1}{2} \prod_{\substack{p \leq \sqrt{n} \\ p>2}} \left(1 - \frac{\omega_n(p)}{p}\right), \quad \omega_n(p) = \begin{cases} 1 & p \mid n \\ 2 & p \nmid n. \end{cases} \quad (3)$$

$L(n)$ est la densité du crible de Legendre à niveau $\sqrt{n}$, c'est-à-dire ce que les cercles "prédisent" naïvement.

4. Test numérique

Le programme en annexe 1 calcule $G(n)$ exactement (autoconvolution par FFT de la fonction indicatrice des premiers) pour tous les n pairs jusqu'à $N = 4 \cdot 10^6$, ainsi que P_1, P_2, L . Rapports moyens par tranche (chaque tranche multiplie n par $\approx 3,76$) :

tranche de n	G/P_1	G/P_2	G/L
1414 à 5318	1,889	0,9468	1,086
5318 à $2 \cdot 10^4$	1,945	0,9732	1,040
$2 \cdot 10^4$ à $7,5 \cdot 10^4$	1,973	0,9869	0,997
$7,5 \cdot 10^4$ à $2,8 \cdot 10^5$	1,987	0,9936	0,967
$2,8 \cdot 10^5$ à $1,06 \cdot 10^6$	1,994	0,9968	0,944
$1,06 \cdot 10^6$ à $4 \cdot 10^6$	1,997	0,9984	0,927

$G/P_2 \rightarrow 1$, et $G/P_1 \approx 2$ en moyenne, ce qui correspond à une moyenne de $2C_2\mathfrak{S}(n)$ égale à 2 ; G/P_1 est dispersé (écart-type $\approx 0,77$) précisément par $\mathfrak{S}(n)$.

5. Les résidus $G/P_2 - 1$

Moyenne. Elle est négative et décroît en $n^{-\alpha}$ avec $\alpha \approx 0,52$ (ajustement sur huit tranches), soit

$$\overline{G/P_2} - 1 \approx -\frac{c}{\sqrt{n}}, \quad c \approx 2,3 \text{ à } 2,8.$$

Ce n'est pas une décroissance en $1/\ln n$. L'exposant $1/2$ est celui du biais $\pi(x) < \text{li}(x)$: le poids $1/\ln x$ somme à li et non à π . *Test* (programme en annexe 2) : on remplace $1/\ln x$ par la dérivée de la fonction de Riemann $R(x) = \sum_k \frac{\mu(k)}{k} \text{li}(x^{1/k})$, soit $\frac{1}{\ln x} \left(1 - \frac{1}{2\sqrt{x}} - \frac{1}{3x^{2/3}} - \dots\right)$. Le biais moyen disparaît : $\overline{G/P_2^R} - 1$ vaut $+0,035; +0,013; +0,004; +0,001; 0,000; \dots$ sur les tranches successives (contre $-0,188; -0,102; \dots; -0,0016$). L'hypothèse est donc confirmée : le terme $-c/\sqrt{n}$ est entièrement le terme $-\frac{1}{2\sqrt{x}}$ de R' . La dispersion est inchangée ; le minimum de G/P_2^R ne s'améliore que légèrement ($-0,033$ au lieu de $-0,035$ sur la dernière tranche), donc le déficit maximal n'est pas un effet de biais mais de fluctuation.

Dispersion. L'écart-type de G/P_2 vaut environ 0,5 à 0,6 fois $P_2^{-1/2}$ (de 0,45 à 0,59 sur les huit tranches, en croissance lente) : les fluctuations sont sous-poissoniennes, la série singulière ayant déjà absorbé une grande part de la variabilité.

Dépendance en $\omega(n)$ (nombre de premiers impairs distincts divisant n , tranche $n \geq 4 \cdot 10^5$) : la moyenne de G/P_2 est constante (0,998 pour $\omega = 1, \dots, 6$) ; seule la dispersion diminue avec ω (de 0,0070 à 0,0024). La série singulière capte donc correctement l'effet des facteurs premiers de n .

Position de n par rapport aux p^2 . Soit p le plus grand premier avec $p^2 \leq n$ et q le suivant ; on découpe $[p^2, q^2)$ en cinq parts. Pour $n \geq 10^6$, la moyenne de G/P_2 varie de 0,99834 à 0,99842 et l'écart-type reste 0,0051 : aucun effet détectable. Pour G/L la variation est de 0,9276 à 0,9262, négligeable devant la dérive en n .

6. Minorant empirique

Le minimum de G/P_2 sur chaque tranche est

$$0,478; 0,605; 0,713; 0,784; 0,866; 0,911; 0,942; 0,965,$$

et le minimum de $G(n)$ lui-même est 3, 7, 20, 52, 155, 452, 1346, 4155 : il croît partout. Sur les cinq dernières tranches, $(1 - \min G/P_2) n^{0,35}$ reste entre 4–5.5, soit un déficit maximal empirique $\approx 4,7 n^{-0,35}$. Le déficit décroît moins vite que la moyenne $(n^{-1/2})$: c’est l’extrême d’une queue, pas une moyenne. Il s’agit d’une description numérique, non d’un minorant démontré.

7. Ce que dit G/L (correction)

La version précédente de ce texte présentait la limite $e^{2\gamma}/4 \approx 0,793$ de G/L comme “l’expression numérique de l’obstruction de parité”. C’est inexact. Cette constante s’explique par l’écart entre produit de Mertens et crible exact au niveau $z = \sqrt{n}$: pour une variable, la densité criblée vaut $\omega(2)/\ln z$ avec $\omega(2) = 1/2$ (fonction de Buchstab), alors que le produit de Mertens donne $e^{-\gamma}/\ln z$; le rapport $e^\gamma/2 \approx 0,8905$, élevé au carré (deux variables x et $n-x$), donne $e^{2\gamma}/4$. Les cercles “naïfs” surestiment donc d’un facteur constant $\approx 1/0,793$, indépendamment de la parité. La convergence est lente (en $1/\ln n$) : à $4 \cdot 10^6$ on est encore à 0,927.

8. Où l’argument cesse d’être une preuve

1. **Le poids $1/\ln x$ est un modèle probabiliste** (Cramér), pas un théorème. Les cercles de logarithme ne démontrent pas que x et $n-x$ sont “indépendants” : c’est exactement ce qu’il faudrait prouver.
2. **$G \sim P_2$ est la conjecture de Hardy–Littlewood**, non démontrée. L’accord numérique ($G/P_2 = 1 - O(n^{-1/2})$ en moyenne) la corrobore, ne la prouve pas.
3. **Le crible exact n’est pas contrôlable**. La formule d’inclusion-exclusion sur les cercles a $2^{\pi(\sqrt{n})}$ termes ; les termes d’erreur d’un crible de dimension 2 ne sont maîtrisés qu’à un niveau $z \leq n^\theta$ avec θ petit. Au niveau $\sqrt{n}$, nécessaire pour lire “premier”, on est hors de la portée du lemme fondamental.
4. **Obstruction de parité (Selberg)**. Même avec les meilleurs cribles de dimension 2, on ne sait pas distinguer un entier ayant un nombre pair ou impair de facteurs premiers, d’où l’impossibilité d’obtenir une minoration strictement positive de $G(n)$ par cette voie (on obtient des résultats du type $n = p + P_2$, Chen). C’est ici, et non dans la valeur de $e^{2\gamma}/4$, que se situe la parité.
5. **Un minorant empirique n’est pas un minorant**. Le déficit $4,7 n^{-0,35}$ observé jusqu’à $4 \cdot 10^6$ n’établit rien pour n arbitraire.

Conclusion : le dispositif donne une prédiction précise et très bien vérifiée, et localise le point exact où une preuve devrait apporter une idée nouvelle (briser la parité), mais n’en fournit pas une.

9. Parité : ce que les cercles voient et ce qu'ils ne voient pas

Les cercles contiennent toute l'information de parité. Soit $x \leq n$ et $m(x)$ sa partie $\sqrt{n}$ -friable (produit des $p^{v_p(x)}$ avec $p \leq \sqrt{n}$, exposants compris). Le cofacteur $x/m(x)$ vaut 1 ou un unique premier $> \sqrt{n}$ (deux seraient de produit $> n$). Donc

$$\lambda(x) = (-1)^{\Omega(x)} = \lambda(m(x)) \cdot (-1)^{[x \neq m(x)]},$$

et x est premier si et seulement si $m(x) = 1$ et $x > 1$. Les cercles, lus avec multiplicités, déterminent donc $\lambda(x)$ exactement : l'obstruction de parité n'est *pas* un manque d'information. C'est un manque de contrôle des *sommes* : les poids d'un crible de dimension 2 ne distinguent pas les ensembles où $\lambda(x(n-x)) = +1$ de ceux où il vaut -1 (Selberg), et la quantité laissée libre est la corrélation de Chowla

$$C(n) = \sum_{x=1}^{n-1} \lambda(x) \lambda(n-x).$$

Mesure (programme en annexe 3, $n \leq \cdot 10^6$). Chaque terme de $C(n)$ apparaît deux fois (x et $n-x$), d'où le modèle aléatoire $E|C|/\sqrt{n} = \sqrt{2/\pi} \sqrt{2} = 1,128$; on observe 1,12–1,13 sur les trois dernières tranches. Le maximum de $|C(n)|/n$ décroît (0,158; 0,096; 0,051; 0,029; 0,0145; 0,0090). Ce que Chowla prédit, et que personne ne sait démontrer *pour chaque* n , est $|C(n)| = o(n)$.

Lien avec G . La corrélation entre le résidu $G/P_2^R - 1$ et $C(n)/\sqrt{n}$ est positive et stable : 0,25; 0,27; 0,26; 0,24; 0,21; 0,19 par tranche, soit environ 4 à 7 % de la variance du résidu. Je ne sais pas si ce lien est causal ou passe par un facteur commun (structure de n); il n'a pas été testé à $\omega(n)$ fixé.

Conclusion honnête. Contourner la parité exigerait une information de “type II” (sommes bilinéaires), ou une borne $|C(n)| \leq \varepsilon n$ pour tout n pair. Je n'ai pas de voie qui le fasse. Ce qui est établi ici : où est le trou (cancellation de $C(n)$, non information), et que numériquement il se comporte comme le modèle aléatoire.

10. Examen critique de la note “Retrouver les décomposants de Goldbach sur la droite des logarithmes” (Mistral, 8 octobre 2026)

Cet examen portait sur la version de l'ia mistral du 8 octobre; elle a permis d'améliorer le texte de l'ia mistral.

La note définit le “microscope” $T_p = n \ln(n/p) = s + B(s)$, avec $s = n - p$ et $B(s) = n \ln(1 + s/n) - s \simeq s^2/2n$, et un “détecteur” : pour un premier $q \leq \sqrt{n}$, $q \mid s$ si et seulement si $\text{dist}(T_p, q\mathbb{Z}) = \text{fold}(B(s), q)$. Les résultats corrects et les réserves sont séparés ci-dessous.

Ce qui est correct. Le développement $T_p = s + B(s)$ est exact, avec $0 < B(s) < s^2/(2(n-s))$. Le point (1) du théorème 3 est exact : si $q \mid s$, alors $\text{dist}(T_p, q\mathbb{Z}) = \text{fold}(B, q)$, car $T_p = qm + B$. Le

tableau pour $n = 98$ est correct.

1. Le détecteur est circulaire. Pour évaluer $B(s)$ il faut connaître $s = n - p$. Le test compare $\text{dist}(s + B, q\mathbb{Z})$ à $\text{dist}(B, q\mathbb{Z})$: B s'élimine, et il reste “ q divise-t-il s ?”. Le microscope ne fait donc que traduire le crible affine ; la note le reconnaît en partie (section 8, point 2), mais l'affirmation de la section 7.1, selon laquelle l'objection de symétrie est “réfutée”, ne tient pas : une bijection comme $\exp$ ne perd jamais d'information, et l'objection portait sur la géométrie (l'involution $x \mapsto n - x$ n'est plus rigide), non sur l'information.

2. La “conjecture de non-collision” est triviale en arithmétique exacte. Une collision avec $q \nmid s$ exige $2B \equiv -r \pmod{q}$ où $r \equiv s \pmod{q}$, $r \not\equiv 0$, c'est-à-dire que $2n \ln(n/p)$ soit entier. Or $\ln(n/p)$ est transcendant (Hermite–Lindemann, n/p rationnel différent de 1), donc irrationnel. Il n'y a aucune collision, quel que soit n ; les “zéro collision jusqu'à 10^6 ” de la section 5 dépendent de la tolérance numérique choisie, non d'un fait arithmétique.

3. Le seuil est faux. $B(s) < \frac{1}{2}$ équivaut, au premier ordre, à $s^2 < n$, c'est-à-dire $s \lesssim \sqrt{n}$, et non $s \lesssim \sqrt{2n}$ comme écrit. La réciproque du théorème 3 n'est donc établie que pour les petits partenaires ; pour presque tous les décomposants s est grand (pour $n = 98$, $p = 19$: $s = 79$, $B = 81,8$), et la “zone utile” annoncée n'existe pas. Le point 2 ci-dessus rend heureusement cette lacune sans conséquence.

4. La constante de la section 7.2 est inversée. La note écrit que G/L dérive vers $4e^{-2\gamma} \approx 1,26$; la limite est $e^{2\gamma}/4 \approx 0,793$ (section “Ce que dit G/L ” ci-dessus). Elle l'attribue à la parité de Selberg ; c'est l'effet Buchstab–Mertens.

5. “Oracle de phase” et Heisenberg (section 8.1). L'égalité de distances à un réseau ne fixe la phase $e^{2\pi i T_p/q}$ qu'à la conjugaison près, et suppose B connu. Le “flou” B est déterministe, sans variables conjuguées : ce n'est pas une relation d'incertitude. La comparaison avec l'oracle de Grover est une analogie de langage, non un résultat.

6. Section 9 (comète). La formule de Hardy–Littlewood pour G non ordonné est $C_2 \mathfrak{S}(n) n / \ln^2 n$ (et non $2C_2 \mathfrak{S}(n) n / \ln^2 n$, qui compte les couples ordonnés). L'épaisseur de la bande en log-log provient surtout du facteur 2 de $\mathfrak{S}(n)$ lorsque $3 \mid n$, d'où les deux bandes visibles ; $\mathfrak{S}$ lui-même croît comme $\ln \ln n$ au maximum.

Bilan. Contribution réelle : une écriture logarithmique exacte du crible affine, avec une formule explicite du flou. Contribution non établie : tout apport au problème d'existence des décomposants ; la note le dit elle-même (section 7.2), ce qui est exact.

11. Une borne combinatoire sur $C(n)$? État des lieux

Ce qui est établi (identité élémentaire, vérifiée numériquement pour $n < 400$). Posons $C^*(m) = \sum_{x < m, \gcd(x, m)=1} \lambda(x) \lambda(m - x)$. En regroupant les x selon $g = \gcd(x, n)$ et en utilisant λ

complètement multiplicative ($\lambda(g)^2 = 1$),

$$C(n) = \sum_{d|n, d>1} C^*(d).$$

Elle ramène $C(n)$ à des sommes sur les x premiers avec le module, mais ne fournit aucune borne.

Ce qui est connu dans la littérature (à ma connaissance, vérifié par recherche le 8 octobre 2026). La conjecture de Chowla à deux points, $\sum_{n \leq x} \lambda(n)\lambda(n+h) = o(x)$, reste ouverte et est tenue pour aussi difficile que la conjecture des premiers jumeaux. Tao (2016) l’a démontrée pour une moyenne *logarithmique*, en citant explicitement le franchissement de la “barrière de parité” pour ce problème ; l’outil central est un argument d’“entropy decrement” combiné à des résultats de Matomäki–Radziwiłł–Tao. La somme $C(n) = \sum_x \lambda(x)\lambda(n-x)$ pour n fixé est un problème additif de type Goldbach, que je ne sais pas réduire à ces résultats moyennés, et je n’ai trouvé aucune borne combinatoire pour $C(n)$ à n fixé.

Correction de ma remarque précédente. Une borne $|C(n)| \leq \varepsilon n$ n’est pas à elle seule le levier suffisant. Ce qu’un crible a besoin de contrôler est la somme *tordue*, restreinte aux x qui survivent au crible, de $\lambda(x(n-x))$; $C(n)$ n’en est que la version non restreinte. L’énoncé précis est en cours de formulation et n’est pas établi ici.

12. Explication simple : pourquoi contrôler la parité changerait tout

Imaginons les x entre 1 et n où ni x ni $n-x$ n’a de petit diviseur (disons aucun diviseur $\leq n^{0,4}$). Chacun des deux est alors soit premier, soit produit d’exactly deux grands premiers (trois grands premiers dépasseraient n). Les survivants sont donc de trois sortes : *les deux premiers* (ce que l’on veut, 2 facteurs au total), *un premier et un produit de deux premiers* (3 facteurs), *deux produits de deux premiers* (4 facteurs). Un crible sait compter combien il y a de survivants en tout, avec bonne précision, mais pas séparer ces trois sortes : c’est la barrière de parité. Chaque survivant porte un signe, +1 si le nombre total de facteurs premiers est pair (sortes 1 et 3), −1 s’il est impair (sorte 2). “Les cercles contiennent l’information” veut dire que l’on peut calculer ce signe pour chaque x en regardant quels cercles le touchent et avec quelle multiplicité. Mais le crible compte en bloc, et dans une somme de signes les +1 et les −1 peuvent se compenser sans que l’on sache le contrôler.

Si l’on savait démontrer que la somme de ces signes sur les survivants est petite devant leur nombre, on saurait qu’environ la moitié des survivants est de sorte 1 ou 3 : on aurait une minoration de “premier+premier ou produit+produit”. Il resterait à traiter séparément la sorte 3 (deux produits de deux premiers), par un crible à un autre niveau : ce n’est donc *pas* la fin du problème, seulement la levée de la partie parité. Et la difficulté de fond est de démontrer cette compensation pour *un seul* n : c’est ce que personne ne sait faire aujourd’hui.

13. Annexe 1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
goldbach_poids_log.py (version 2)

Pour tout n pair,  $6 \leq n \leq N$ , compare le nombre reel  $G(n)$  de decomposants de
Goldbach ( $p \leq n/2$ , p et n-p premiers) a trois predictions :
    P1(n) =  $\sum_{x=2}^{n/2} 1/(\ln x \ln(n-x))$  (poids de Gauss, cas B)
    P2(n) =  $2 \cdot C2 \cdot S(n) \cdot P1(n)$  (avec serie singuliere)
    L(n) =  $(n/4) \cdot \prod_{2 < p \leq \sqrt{n}} (1 - w_n(p)/p)$  (densite du crible de Legendre)
avec  $S(n) = \prod_{p|n, p>2} (p-1)/(p-2)$ ,  $w_n(p) = 1$  si  $p|n$ , 2 sinon.

Sortie :
(1) tableau par tranches ;
(2) residus  $G/P2 - 1$  : moyenne, ecart-type,
min, max par tranche ;
(3) dependance en  $\omega(n)$  (nb de premiers impairs
distincts de n) ;
(4) decroissance de  $G/P2 - 1$  en loi de puissance ;
(5) minorant empirique
de  $G/P2$  ;
(6) rapport  $G/L$  et limite  $\exp(2 \gamma)/4$ .

Usage : python3 goldbach_poids_log.py [N] [--csv fichier.csv]
"""
import sys
import numpy as np
C2 = 0.6601618158468696
GAMMA = 0.5772156649015329

def crible(N):
    est = np.ones(N + 1, dtype=bool)
    est[:2] = False
    for i in range(2, int(N ** 0.5) + 1):
        if est[i]:
            est[i * i::i] = False
    return est

def autoconvolution(a, N):
    """(a*a)(n) pour n = 0..N, par FFT."""
    taille = 1 << (2 * N + 1).bit_length()
    F = np.fft.rfft(a, taille)
    return np.fft.irfft(F * F, taille)[:N + 1]

def main():
    args = sys.argv[1:]
    csv = None
    if "--csv" in args:
        i = args.index("--csv")
        csv = args[i + 1]
        del args[i:i + 2]
    N = int(float(args[0])) if args else 10 ** 6
```

```

N -= N % 2
P = crible(N)
premiers = np.nonzero(P)[0]
n = np.arange(N + 1)
# G(n) exact
r = np rint(autoconvolution(P.astype(float), N)).astype(np.int64)
demi = np.zeros(N + 1, dtype=np.int64)
demi[2::2] = P[1:N // 2 + 1]
G = (r + demi) // 2
# P1(n)
w = np.zeros(N + 1)
w[2:] = 1.0 / np.log(np.arange(2, N + 1))
P1 = autoconvolution(w, N) / 2.0
P1[2::2] += 0.5 * w[1:N // 2 + 1] ** 2
# S(n) et omega(n) (premiers impairs distincts)
S = np.ones(N + 1)
omega = np.zeros(N + 1, dtype=np.int64)
for p in premiers[1:]:
    S[p::p] *= (p - 1.0) / (p - 2.0)
    omega[p::p] += 1
# L(n)
L = n / 4.0
sq = int(N ** 0.5) + 1
for p in premiers[1:]:
    if p > sq:
        break
    facteur = np.where(n % p == 0, 1 - 1.0 / p, 1 - 2.0 / p)
    L = np.where(n >= p * p, L * facteur, L)

pairs = n[6:][n[6:] % 2 == 0]
g, p1, s, l, om = G[pairs], P1[pairs], S[pairs], L[pairs], omega[pairs]
p2 = 2 * C2 * s * p1

if csv:
    np.savetxt(csv, np.c_[pairs, g, p1, p2, l], delimiter=",",
                header="n,G,P1,P2,L", comments="", fmt="%6g")
    print(f"N = {N} ({len(pairs)} entiers pairs)\n")
    bords = np.unique(np.logspace(2, np.log10(N), 9).astype(int))
    tranches = list(zip(bords[:-1], bords[1:]))

    print("(1) Rapports moyens par tranche")
    print(f"{ 'tranche de n':>22} | { 'G/P1':>7} { 'sd':>6} | { 'G/P2':>7}")
    print(f"{ 'sd':>6} | { 'G/L':>7} { 'sd':>6}")
    for a, b in tranches:
        m = (pairs >= a) & (pairs < b)
        if m.sum() < 5:
            continue
        r1, r2, r3 = g[m] / p1[m], g[m] / p2[m], g[m] / l[m]
        print(f"{a:>10} — {b:<9} | {r1.mean():7.4f} {r1.std():6.4f} | "
              f"{r2.mean():7.4f} {r2.std():6.4f}|{r3.mean():7.4f} {r3.std():6.4f}")
    print("\n(2) Residus G/P2-1 par tranche:moyenne,sd,min,max,et sd*sqrt(P2)")
    lnm, moy = [], []
    for a, b in tranches:
        m = (pairs >= a) & (pairs < b)
        if m.sum() < 5:

```

```

        continue
    e = g[m] / p2[m] - 1
    print(f"{a:>10} — {b:<9} | moy {e.mean():+.4f} sd {e.std():.4f} "
          f"min {e.min():+.4f} max {e.max():+.4f} | sd*sqrt(P2moy) = "
          f"{e.std() * np.sqrt(p2[m].mean()):.3f}")
    lnm.append(np.log(np.sqrt(a * b)))
    moy.append(e.mean())
    print("\n(3) Dependance en omega(n), tranche haute (n >= N/10) : G/P2")
    m = pairs >= N // 10
    for k in sorted(set(om[m])):
        mk = m & (om == k)
        if mk.sum() >= 20:
            e = g[mk] / p2[mk]
            print(f"omega = {k}:{mk.sum():7d} entiers , G/P2 moyen {e.mean():.4f}, "
                  f"min {e.min():.4f}, sd {e.std():.4f}")
    print("\n(4) Ajustement en loi de puissance : ")
    print("| moyenne de G/P2 - 1 | ~ c * n^(-alpha)")
    if len(moy) >= 3:
        al, lc = np.polyfit(np.array(lnm), np.log(-np.array(moy)), 1)
        print(f" alpha = {-al:.3f}, c = {np.exp(lc):.2f} (alpha ~ 1/2 :)")
        print(f" biais de type li(x) - pi(x)")

    print("\n(5) Minorant empirique : min de G(n)/P2(n) par tranche , et min G(n)")
    for a, b in tranches:
        m = (pairs >= a) & (pairs < b)
        if m.sum() < 5:
            continue
        k = np.argmax(g[m] / p2[m])
        print(f"{a:>10} — {b:<9} | min G/P2 = {(g[m] / p2[m]).min():.4f} "
              f"(n = {pairs[m][k]}) | min G = {g[m].min()} | (1-min)*n^0.35 = "
              f"{(1 - (g[m] / p2[m]).min()) * pairs[m][k] ** 0.35:.2f}")

    print(f"\n(6) G/L doit tendre (lentement , en 1/ln n) vers exp(2 gamma)/4 = "
          f"{np.exp(2 * GAMMA) / 4:.4f}")

if __name__ == "__main__":
    main()

```

Son résultat pour $N = 10^6$

$N = 1000000$ (499998 entiers pairs)

(1) Rapports moyens par tranche

tranche de n	G/P1	sd	G/P2	sd	G/L	sd
100 — 316	1.5789	0.5334	0.8089	0.1305	1.2503	0.2009
316 — 1000	1.7418	0.6478	0.8791	0.0942	1.1849	0.1323
1000 — 3162	1.8615	0.7114	0.9345	0.0679	1.1132	0.0832
3162 — 10000	1.9228	0.7421	0.9627	0.0461	1.0592	0.0529
10000 — 31622	1.9582	0.7585	0.9796	0.0307	1.0207	0.0359
31622 — 100000	1.9781	0.7671	0.9892	0.0207	0.9899	0.0226
100000 — 316227	1.9878	0.7711	0.9940	0.0134	0.9629	0.0147
316227 — 1000000	1.9937	0.7734	0.9969	0.0086	0.9441	0.0098

(2) Residus $G/P2 - 1$ par tranche :

moyenne, sd, min , max , et sd*sqrt(P2)									
100 — 316	moy	-0.1911	sd	0.1305	min	-0.5219	max	+0.1943	sd*sqrt(P2moy) = 0.4
316 — 1000	moy	-0.1209	sd	0.0942	min	-0.4285	max	+0.1796	sd*sqrt(P2moy) = 0.4
1000 — 3162	moy	-0.0655	sd	0.0679	min	-0.2911	max	+0.1644	sd*sqrt(P2moy) = 0.4
3162 — 10000	moy	-0.0373	sd	0.0461	min	-0.2189	max	+0.1336	sd*sqrt(P2moy) = 0.4
10000 — 31622	moy	-0.0204	sd	0.0307	min	-0.1584	max	+0.1113	sd*sqrt(P2moy) = 0.5
31622 — 100000	moy	-0.0108	sd	0.0207	min	-0.1163	max	+0.0946	sd*sqrt(P2moy) = 0.5
100000 — 316227	moy	-0.0060	sd	0.0134	min	-0.0824	max	+0.0608	sd*sqrt(P2moy) = 0.5
316227 — 1000000	moy	-0.0031	sd	0.0086	min	-0.0498	max	+0.0452	sd*sqrt(P2moy) = 0.5

(3) Dependance en omega(n), tranche haute (n >= N/10) : G/P2
omega = 1: 78204 entiers ,G/P2 moyen0.9963,**min** 0.9176, sd 0.0122
omega = 2: 189144 entiers ,G/P2 moyen0.9961,**min** 0.9305, sd 0.0106
omega = 3: 144131 entiers ,G/P2 moyen0.9961,**min** 0.9364, sd 0.0086
omega = 4: 36301 entiers ,G/P2 moyen0.9962,**min** 0.9566, sd 0.0067
omega = 5: 2210 entiers ,G/P2 moyen0.9965,**min** 0.9769, sd 0.0049

(4) Ajustement en loi de puissance : |moyenne de G/P2 - 1| ~ c * n^(-alpha)
alpha = 0.515, c = 3.03
(alpha ~ 1/2 : biais de **type** li(x) - pi(x))

(5) Minorant empirique : **min** de G(n)/P2(n) par tranche, et **min** G(n)
100 — 316 | **min** G/P2 = 0.4781 (n = 128) | **min** G = 3 | (1-**min**)*n^0.35 = 2.85
316 — 1000 | **min** G/P2 = 0.5715 (n = 332) | **min** G = 6 | (1-**min**)*n^0.35 = 3.27
1000 — 3162 | **min** G/P2 = 0.7089 (n = 1412) | **min** G = 16 | (1-**min**)*n^0.35 = 3.69
3162 — 10000 | **min** G/P2 = 0.7811 (n = 4472) | **min** G = 37 | (1-**min**)*n^0.35 = 4.15
10000 — 31622 | **min** G/P2 = 0.8416 (n = 13892) | **min** G = 92 | (1-**min**)*n^0.35 = 4.46
31622 — 100000 | **min** G/P2 = 0.8837 (n = 33038) | **min** G = 223 | (1-**min**)*n^0.35 = 4.44
100000 — 316227 | **min** G/P2 = 0.9176 (n = 144128) | **min** G = 570 | (1-**min**)*n^0.35 = 5.27
316227 — 1000000 | **min** G/P2 = 0.9502 (n = 345502) | **min** G = 1494 | (1-**min**)*n^0.35 = 4.3

(6) G/L doit tendre (lentement, en 1/ln n) vers exp(2 gamma)/4 = 0.7931

14. Annexe 2

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
goldbach_poids_riemann.py

Teste l'hypothese : le biais moyen -c/sqrt(n) de G/P2 vient de ce que le poids
1/ln x somme a li(x) et non a pi(x). On remplace 1/ln x par la densite de la
fonction de Riemann R(x) = sum_k mu(k)/k li(x^(1/k)) :
R'(x) = (1/ln x) * sum_k mu(k)/k * x^(1/k - 1)
puis P1R(n) = demi-somme de l'autoconvolution de ce poids, P2R = 2*C2*S*P1R.
Compare, par tranche, la moyenne de G/P2 et de G/P2R, et le minimum.

Usage : python3 goldbach_poids_riemann.py [N]
"""
import sys
import numpy as np
```

```

C2 = 0.6601618158468696

def crible(N):
    est = np.ones(N + 1, dtype=bool)
    est[:2] = False
    for i in range(2, int(N ** 0.5) + 1):
        if est[i]:
            est[i * i::i] = False
    return est

def autoconv(a, N):
    t = 1 << (2 * N + 1).bit_length()
    F = np.fft.rfft(a, t)
    return np.fft.irfft(F * F, t)[:N + 1]

def mobius(K):
    mu = np.ones(K + 1, dtype=int)
    mu[0] = 0
    for p in range(2, K + 1):
        if all(p % d for d in range(2, int(p ** 0.5) + 1)):
            mu[p::p] *= -1
            if p * p <= K:
                mu[p * p::p * p] = 0
    return mu

def main():
    N = int(float(sys.argv[1])) if len(sys.argv) > 1 else 4 * 10 ** 6
    N -= N % 2
    P = crible(N)
    premiers = np.nonzero(P)[0]
    n = np.arange(N + 1)
    r = np rint(autoconv(P.astype(float), N)).astype(np.int64)
    demi = np.zeros(N + 1, dtype=np.int64)
    demi[2::2] = P[1:N // 2 + 1]
    G = (r + demi) // 2
    x = np.arange(2, N + 1, dtype=float)
    K = int(np.log2(N)) + 1
    mu = mobius(K)
    corr = sum(mu[k] / k * x ** (1.0 / k - 1) for k in range(1, K + 1))
    w1 = np.zeros(N + 1); w1[2:] = 1 / np.log(x)
    w2 = np.zeros(N + 1); w2[2:] = corr / np.log(x)
    def demi_somme(w):
        P1 = autoconv(w, N) / 2.0
        P1[2::2] += 0.5 * w[1:N // 2 + 1] ** 2
        return P1
    P1, P1R = demi_somme(w1), demi_somme(w2)
    S = np.ones(N + 1)
    for p in premiers[1:]:
        S[p::p] *= (p - 1.0) / (p - 2.0)
    pairs = n[6:][n[6:] % 2 == 0]
    g = G[pairs]
    p2, p2r = 2 * C2 * S[pairs] * P1[pairs], 2 * C2 * S[pairs] * P1R[pairs]
    print(f"N = {N}\n")
    print(f"{'tranche':>22} | {'moy G/P2-1':>11} {'sd':>7} {'min':>7} | "
          f"{'moy G/P2R-1':>11} {'sd':>7} {'min':>7} | {'moy*sqrt(n)':>11}")

```

```

bords = np.unique(np.logspace(2, np.log10(N), 9).astype(int))
for a, b in zip(bords[:-1], bords[1:]):
    m = (pairs >= a) & (pairs < b)
    if m.sum() < 5:
        continue
    e1, e2 = g[m] / p2[m] - 1, g[m] / p2r[m] - 1
    nm = np.sqrt(a * b)
    print(f"{a:>10} — {b:<9} | {e1.mean():+11.4f} {e1.std():7.4f} {e1.min():+7.3f} | "
          f"{e2.mean():+11.4f} {e2.std():7.4f} {e2.min():+7.3f} | "
          f"{e1.mean() * np.sqrt(nm):+7.2f} -> {e2.mean() * np.sqrt(nm):+5.2f}")

if __name__ == "__main__":
    main()

```

Son résultat

N = 4000000									
tranche		moy G/P2-1	sd	min	moy G/P2R-1	sd	min		moy*sqrt
100 — 376		-0.1876	0.1257	-0.522	+0.0346	0.1615	-0.345		
-2.61 -> +0.48									
376 — 1414		-0.1023	0.0868	-0.395	+0.0126	0.0962	-0.304		
-2.76 -> +0.34									
1414 — 5318		-0.0532	0.0578	-0.287	+0.0040	0.0608	-0.240		
-2.79 -> +0.21									
5318 — 20000		-0.0268	0.0374	-0.216	+0.0011	0.0381	-0.191		
-2.73 -> +0.11									
20000 — 75212		-0.0131	0.0234	-0.134	+0.0004	0.0236	-0.117		
-2.58 -> +0.08									
75212 — 282842		-0.0064	0.0143	-0.089	+0.0002	0.0143	-0.081		
-2.44 -> +0.07									
282842 — 1063659		-0.0032	0.0086	-0.058	+0.0000	0.0086	-0.053		
-2.37 -> +0.02									
1063659 — 4000000		-0.0016	0.0051	-0.035	+0.0000	0.0051	-0.033		
-2.27 -> +0.02									

15. Annexe 3

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
goldbach_parite_chowla.py

```

Diagnostic de la parite : le crible ne voit pas la fonction de Liouville
 $\lambda(x) = (-1)^{\Omega(x)}$. La quantité que tout crible "aveugle a la parite"
laisse non contrôlée est la corrélation

$$C(n) = \sum_{x=1}^{n-1} \lambda(x) \lambda(n-x) \quad (\text{Chowla, 2 points}).$$

Le script calcule $C(n)$ pour tout n pair $\leq N$, la compare au modèle aléatoire
 $(|C| \sim \sqrt{n})$, et mesure sa corrélation avec le résidu $G/P2R - 1$ (poids de
Riemann, biais en $1/\sqrt{n}$ déjà retiré).

```

Usage : python3 goldbach_parite_chowla.py [N]
"""
import sys
import numpy as np

def omega_grand(N):
    """Omega(x) (avec multiplicité) pour x <= N, par crible."""
    om = np.zeros(N + 1, dtype=np.int32)
    for p in range(2, N + 1):
        if om[p] == 0 and p * p > N:          # p premier non marque et p^2 > N
            om[p::p] += 1                    # reste : tous les multiples (exposant 1)
            continue
        if om[p] == 0:                        # p premier (premiere visite)
            q = p
            while q <= N:
                om[q::q] += 1
                q *= p
    return om

def autoconv(a, N):
    t = 1 << (2 * N + 1).bit_length()
    F = np.fft.rfft(a, t)
    return np.fft.irfft(F * F, t)[:N + 1]

def main():
    N = int(float(sys.argv[1])) if len(sys.argv) > 1 else 10 ** 6
    N -= N % 2
    # — primes, G, P2R (poids de Riemann)
    est = np.ones(N + 1, bool); est[:2] = False
    for i in range(2, int(N ** .5) + 1):
        if est[i]:
            est[i * i::i] = False
    premiers = np.nonzero(est)[0]
    r = np rint(autoconv(est.astype(float), N)).astype(np.int64)
    demi = np.zeros(N + 1, dtype=np.int64); demi[2::2] = est[1:N // 2 + 1]
    G = (r + demi) // 2
    x = np.arange(2, N + 1, dtype=float)
    mu = np.zeros(21, int); mu[1] = 1
    for k, m in [(2, -1), (3, -1), (5, -1), (6, 1), (7, -1), (10, 1), (11, -1),
                 (13, -1), (14, 1), (15, 1), (17, -1), (19, -1)]:
        mu[k] = m
    corr = sum(mu[k] / k * x ** (1.0 / k - 1) for k in range(1, 21))
    w = np.zeros(N + 1); w[2:] = corr / np.log(x)
    P1 = autoconv(w, N) / 2.0
    P1[2::2] += 0.5 * w[1:N // 2 + 1] ** 2
    S = np.ones(N + 1)
    for p in premiers[1:]:
        S[p::p] *= (p - 1.0) / (p - 2.0)
    C2 = 0.6601618158468696
    # — Liouville
    lam = np.zeros(N + 1)
    lam[1:] = (-1.0) ** omega_grand(N)[1:]
    C = np rint(autoconv(lam, N))          # C[n] = sum_{a+b=n, a,b>=1} lam(a)lam(b)
    pairs = np.arange(6, N + 1, 2)
    res = G[pairs] / (2 * C2 * S[pairs] * P1[pairs]) - 1

```

```

c = C[pairs]
print(f"N = {N}\n")
print(f"{'tranche':>22} | {'moy |C|/sqrt(n)':>15} {'max |C|/n':>10} | {'corr(res , C/sqrt n)'}")
bords = np.unique(np.logspace(3, np.log10(N), 7).astype(int))
for a, b in zip(bords[:-1], bords[1:]):
    m = (pairs >= a) & (pairs < b)
    nn = pairs[m]
    cc = np.corrcoef(res[m], c[m] / np.sqrt(nn))[0, 1]
    print(f"{a:>10} — {b:<9} | {np.mean(np.abs(c[m]) / np.sqrt(nn)):15.3f} "
        f"{np.max(np.abs(c[m]) / nn):10.4f} | {cc:20.3f}")
print(f"\nModele aleatoire : chaque terme apparait deux fois (x et n-x), donc")
print(f"E|C|/sqrt(n) = sqrt(2/pi)*sqrt(2) = 1.128 ; et max |C|/n doit tendre vers 0.")
print(f"Si la correlation avec le residu est ~0, la parite ne se voit pas dans G/P2.")

if __name__ == "__main__":
    main()

```

Son résultat

N = 1000000

	tranche	moy C /sqrt(n)	max C /n	corr(res , C/sqrt n)
1000 —	3162	0.960	0.1580	0.234
3162 —	10000	1.055	0.0956	0.273
10000 —	31622	1.097	0.0514	0.270
31622 —	100000	1.115	0.0337	0.244
100000 —	316227	1.121	0.0178	0.218
316227 —	1000000	1.125	0.0102	0.197

Modele aleatoire : chaque terme apparait deux fois (x et n-x), donc
 $E|C|/\sqrt{n} = \sqrt{2/\pi} \cdot \sqrt{2} = 1.128$; et **max** $|C|/n$ doit tendre vers 0.
 Si la correlation avec le residu est ~0, la parite ne se voit pas dans G/P2.