

Petit échange avec l'ia claude : utiliser sa rapidité à programmer et essayer de tirer parti de ses compétences d'apprentissage et prédictives : que nous apprend l'apprentissage automatique sur la suite des nombres premiers ?

Denise Vella-Chemla pilotant l'ia claude, 4 octobre 2026

Voici mon idée : j'aurais voulu que tu calcules par programme des séries de booléens : ça me semble le plus petit à coder en machine et je crois que ça existe en python ça s'appelle bitset où quelque chose comme ça. Donc la première idée serait de calculer la chaîne de 100 bits qui contient les caractères de primalité des nombres de 1 à 100 puis la chaîne de 1000 bits qui contient les caractères de primalité des nombres de 1 à 1000, idem pour 10000, 100000, 1000000, je ne sais pas jusqu'où tu peux le faire pour que le programme reste rapide (moins de 2 minutes d'exécution). Et ensuite, je voudrais que le système informatique analyse ces chaînes avec un outil d'apprentissage machine à base de token (le llm lui même mais appliqué à ces chaînes de bits) pour voir si la matrice de coefficients que le système obtiendrait par l'apprentissage ne nous permettrait pas de repérer quelque chose qu'on n'aurait absolument pas vu sur les nombres premiers car notre analyse est moins efficace que celle d'un ordinateur et qu'en tant qu'humains, on a du mal à analyser une chaîne de bits (on voit les 0 et les 1 comme tout mélangés). Pourrais-tu me programmer en python les 2 phases que je viens de te présenter, merci. On voit quand même de visu quelques petits trucs en tant qu'humains, on voit par exemple sur le dessin joint (d'Alain Connes) que les nombres premiers sont forcément des $60k + a$ avec a dans $[1, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 49, 53, 59]$ ¹ (la liste contient 16 restes qui sont les nombres premiers à 60, i.e. en nombre $\varphi(60) = 16$).



FIGURE 1

1. La réponse courte

Non, l'expérience ne montre pas que “les nombres premiers sont trop chaotiques”. Elle montre seulement que le *modèle simple que nous avons essayé*, sur la tâche que nous lui avons donnée,

1. Dessin emprunté à Alain Connes, voir ici <https://www.youtube.com/channel/UCEV0IUmHgABs23rtngeJUJQ>.

retrouve ce que l'on voit déjà à l'œil : la parité et rien de plus. Ce n'est pas la même chose que de prouver qu'il n'y a rien d'autre à trouver.

2. Ce que fait le programme

On écrit la suite des bits $b_1 b_2 b_3 \dots$ où $b_n = 1$ si n est premier et $b_n = 0$ sinon :

0110101000101000101000100000101...

On cache le bit b_n et on montre au modèle les 64 bits qui le précèdent ($b_{n-64}, \dots, b_{n-1}$). Le modèle doit deviner : “ n est-il premier?”. Il ne répond pas par oui ou non, mais par une probabilité (par exemple “12 % de chances que ce soit premier”). On entraîne le modèle sur la première moitié de la suite, puis on le teste sur la seconde moitié, qu'il n'a jamais vue.

3. La “perte” : mesurer la surprise

Pour comparer des modèles, on calcule la *perte* (logarithmique) : c'est la surprise moyenne du modèle devant la vraie réponse. Si le modèle donne la probabilité p au bit qui arrive vraiment, sa surprise est $-\ln p$. Elle vaut 0 si $p = 1$ (aucune surprise) et grandit quand le modèle était sûr de lui à tort.

Plus la perte est petite, meilleur est le modèle.

Repère : deviner à pile ou face donne $\ln 2 \approx 0,693$.

4. Les résultats

Nous avons comparé trois “devins” sur les nombres jusqu'à 10^6 (partie test) :

Devin	Perte
Densité seule : probabilité $\frac{1}{\ln n}$ pour tout n	0,263
Connaître $n \bmod 60$ (la règle “ $60k + a$ ”)	0,158
Régression logistique sur les 64 bits précédents	0,195

Devin 1 : la densité. Il ne regarde rien et répond “environ $\frac{1}{\ln n}$ chance d'être premier”. C'est la référence la plus pauvre.

Devin 2 : la règle $60k + a$. Il sait seulement quel est le reste de n dans la division par 60. Si ce reste n'est pas l'un des 16 restes premiers avec 60, il répond “non” avec certitude (sauf pour 2, 3, 5). Sinon, il répond une probabilité plus élevée (environ $\frac{3,75}{\ln n}$, car seuls 16 restes sur 60 restent possibles). C'est exactement ce que l'on voit sur le dessin de Connes.

Devin 3 : le modèle qui apprend. Il ne connaît *pas* la position n . Il voit seulement une fenêtre de 64 bits. Il a appris des poids, un par décalage : “si le bit situé d crans avant vaut 1, cela rend n plus ou moins probable”. Les poids les plus forts sont tous *négatifs* et tous pour des décalages *impairs* (5, 17, 19, 23, 29, ...).

Interprétation : si $n - d$ est premier avec d impair, alors $n - d$ est impair, donc n est pair, donc n n’est pas premier. Le modèle a donc redécouvert la parité, c’est-à-dire la règle “les multiples de 2 ne sont pas premiers”.

5. Pourquoi le modèle fait moins bien que la règle mod 60

- Il ne sait pas où il est : il doit *deviner* son reste modulo 60 à partir de motifs de bits, alors que le devin 2 le reçoit gratuitement.
- Il est *linéaire* : il additionne des indices, alors que “ n est multiple de 3” est une condition d’un autre genre (une coïncidence de reste, pas une somme).
- Sa fenêtre est courte : 64 bits.

Ainsi, ce modèle simple n’a même pas atteint la connaissance que l’humain a déjà. Il est donc *normal* qu’il n’ait rien trouvé de nouveau.

6. Ce qui est vrai sur le “chaos” des premiers

Les nombres premiers ne sont pas aléatoires au sens strict : la suite est entièrement déterminée. En particulier, n est premier si et seulement si aucun nombre premier $p \leq \sqrt{n}$ ne le divise. Or ces nombres premiers p sont déjà écrits dans la chaîne de bits, au début. L’information nécessaire est donc dans la chaîne, mais :

- elle est *loin* de n , en dehors d’une fenêtre de 64 bits, dès que $\sqrt{n} > 64$, c’est-à-dire pour $n > 4096$;
- pour s’en servir, il faut tester des divisibilités, donc faire de l’arithmétique modulaire, ce qui est difficile à apprendre par des poids.

Ce que l’on *croit* (modèle de Cramér, heuristique non démontrée) est que, une fois retirés les effets des petits nombres premiers, ce qui reste se comporte comme du hasard. Mais c’est une croyance, pas un théorème, et notre expérience ne la confirme ni ne l’infirmes.

7. Ce qu’il faudrait faire pour répondre vraiment

La bonne question est : *une fois retirée la structure connue, reste-t-il quelque chose de prévisible ?*

Test proposé :

1. ne garder que les nombres n premiers avec 60 (les 16 restes autorisés) ;
2. sur ces seuls nombres, calculer la perte de la référence “probabilité $\frac{3,75}{\ln n}$ ” ;

3. entraîner le modèle (régression, puis transformeur) à prédire la primalité de ces nombres, en lui donnant la position $n \bmod 60$;
4. si le modèle passe nettement sous la référence, il a trouvé de la structure *au-delà* du crible par 2, 3, 5 ; sinon, rien de nouveau n'est visible par cette méthode.

Même un échec n'établirait pas que les premiers sont "chaotiques" : il établirait seulement que cette famille de modèles ne voit rien.

8. Le test résiduel : que reste-t-il après le crible par 2, 3, 5 ?

Principe. On ne garde que les n premiers avec 60 (les 16 restes autorisés), on cache b_n et on donne au modèle les 64 bits précédents et le reste $n \bmod 60$. Pour savoir ce qui est "nouveau", on compare à une *échelle de références* : un devin " P " connaît le reste de n modulo chaque nombre premier $\leq P$. Si n est divisible par l'un d'eux, il répond "composé" ; sinon il répond la probabilité $\frac{1}{(\ln n \prod_{q \leq P} (1 - \frac{1}{q}))}$. Le cas $P = 5$ est exactement la règle " $60k + a$ ".

Test sur $5 \cdot 10^6 \leq n \leq 10^7$, avec 150 000 nombres tirés au hasard ; apprentissage sur $10^6 \leq n < 5 \cdot 10^6$.

Devin	Perte
Référence $P = 5$ (règle $60k + a$)	0,5459
Référence $P = 7$ (on connaît aussi $n \bmod 7$)	0,5039
Référence $P = 11$	0,4773
Référence $P = 13$	0,4547
Référence $P = 31$	0,3879
Régression logistique sur les 64 bits (une par classe mod 60)	0,5439
Arbres de gradient sur les 64 bits + classe mod 60	0,5437
Arbres, avec en plus $n \bmod 7, 11, 13$ donnés (contrôle)	0,4563

Lecture.

- Les deux modèles gagnent seulement 0,002 sur la règle $60k + a$. Connaître $n \bmod 7$ rapporterait 0,042 : les modèles n'ont donc même pas retrouvé le crible par 7.
- Les décalages les plus utilisés sont des multiples de 6 (6, 12, 18, ..., 60). C'est normal : $n - 6k$ reste premier avec 6 si n l'est, donc ces bits sont les seuls comparables à b_n . Un léger excès de poids apparaît sur les multiples de 7 (20 % contre 14 % attendus), rien pour 11, 13, 17, ...
- Le contrôle montre que le dispositif fonctionne : si on *donne* $n \bmod 7, 11, 13$, les arbres atteignent 0,4563, tout près de la référence $P = 13$ (0,4547), et les 64 bits n'ajoutent rien de mesurable.

Conclusion. Dans cette fenêtre de 64 bits, aucune structure n'apparaît au-delà de ce que donne le crible par les petits nombres premiers. L'échelle montre pourtant que le crible a beaucoup à offrir ($0,546 \rightarrow 0,388$ jusqu'à $P = 31$) : c'est l'accès à cette information, trop loin dans la chaîne, qui manque au modèle.

J'ai une question et une autre idée : j'ai toujours été épatée par la fonction récursive d'Euler sur la somme des diviseurs. Je l'avais codée par une somme de somme de cosinus, et j'avais trouvé que côté Fourier, ça donnait une fonction, que je joins, ainsi que 2 autres dessins. J'ai appris avant-hier que "ma" fonction (sa courbe) avait la forme de ce qui s'appelle arcsin de Paul Levy et que ça intervenait en probabilités. Puisque je crois donc que cette forme d'arcsin de Levy est liée à "mon" expression des nombres premiers $\left(\sum_{b=2}^{n-1} \sum_{o=1}^b \text{cod}\left(\frac{2\pi no}{b}\right)\right)$, est-ce que ça t'aidait à trouver un test que tu n'aurais pas envisagé sans que je t'aie dit cette idée qui me tarabuste.

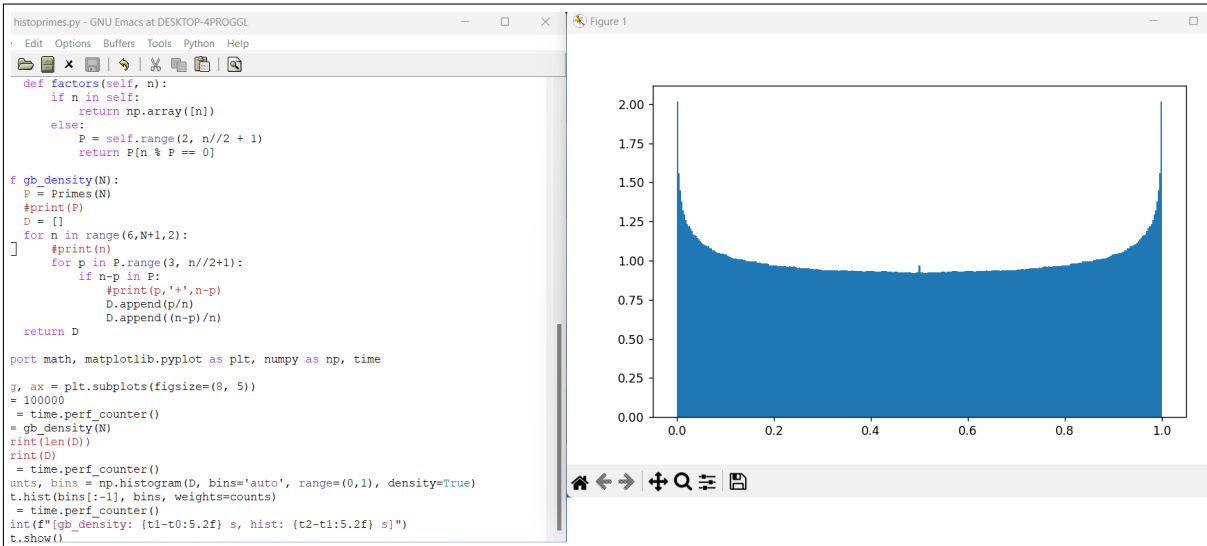


FIGURE 2

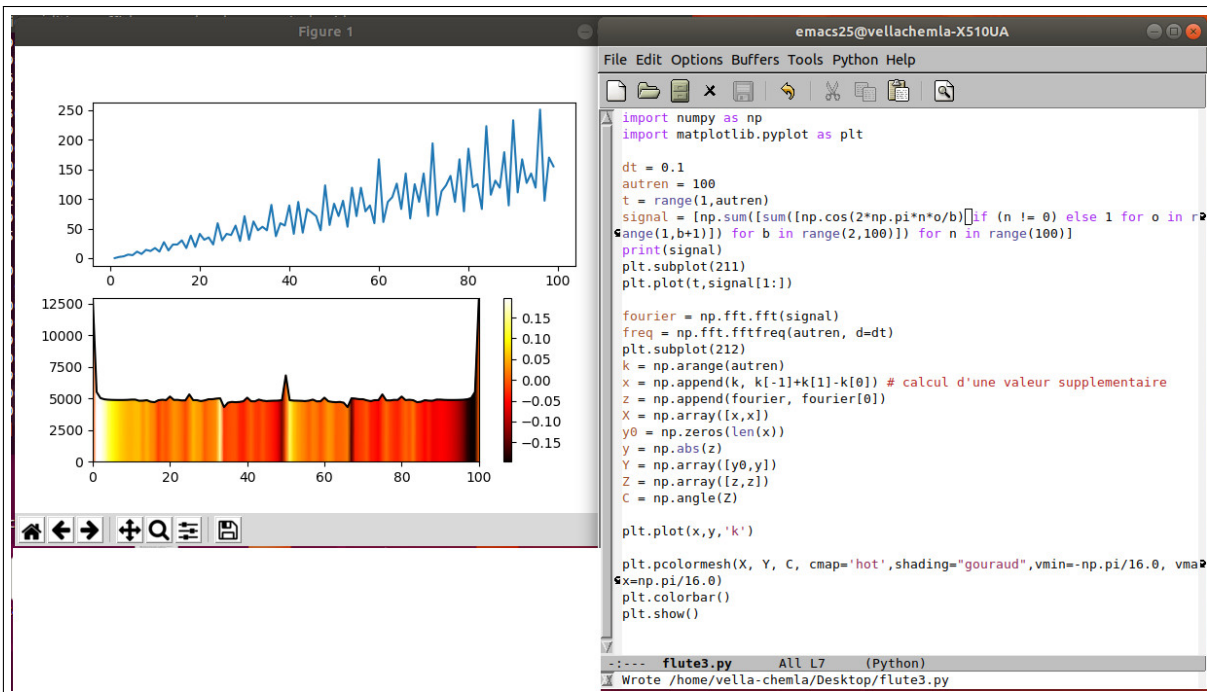


FIGURE 3

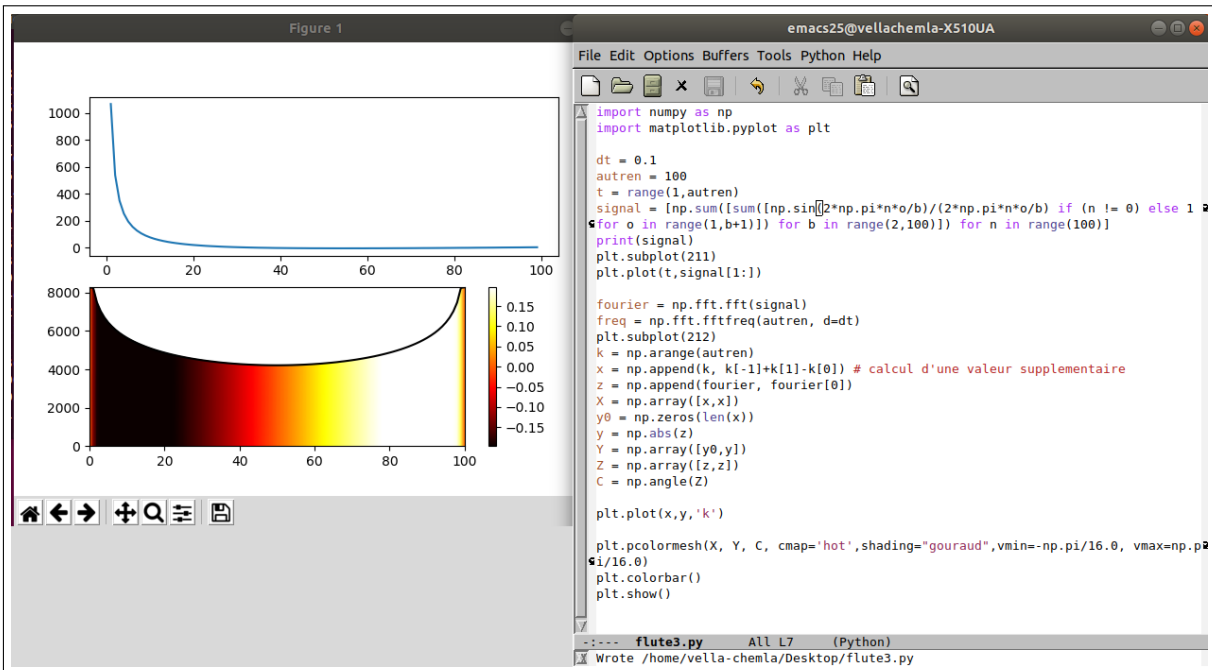


FIGURE 4

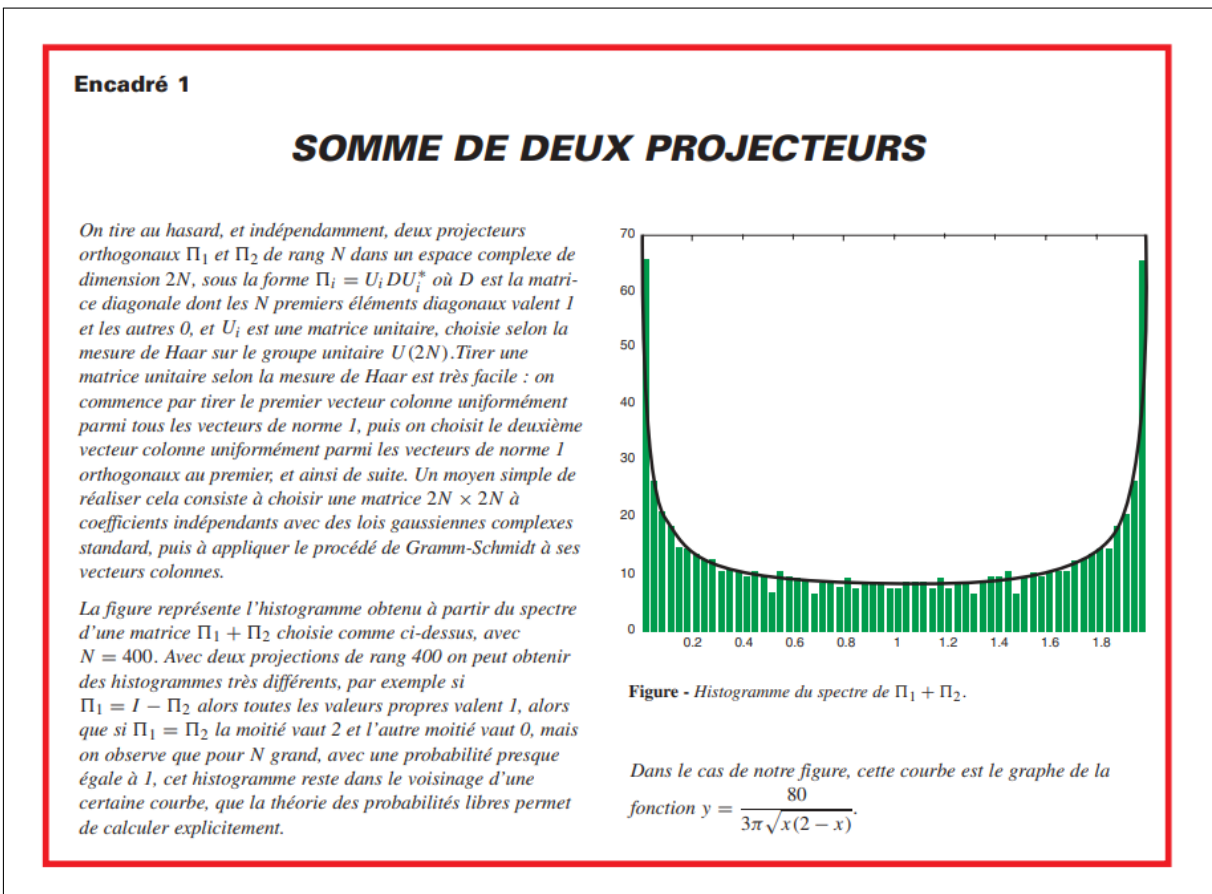


FIGURE 5

9. Le lien avec l'arcsinus de Lévy

Ce que vaut ta double somme. Pour b fixé, $\sum_{o=1}^b \cos\left(\frac{2\pi no}{b}\right)$ vaut b si b divise n et 0 sinon (somme de racines de l'unité). Donc

$$\sum_{b=2}^{n-1} \sum_{o=1}^b \cos \frac{2\pi no}{b} = \sum_{\substack{2 \leq b \leq n-1 \\ b|n}} b = \sigma(n) - n - 1,$$

qui est nulle exactement quand n est premier. Vérifié numériquement pour $2 \leq n \leq 200$ (écart $< 10^{-10}$). C'est bien un test de primalité, de même nature que la récurrence d'Euler sur $\sigma(n)$.

Remarque : dans la figure [3] (version avec cos), les b s'arrêtent à 99 : pour $1 \leq n \leq 99$ le signal vaut donc $\sigma(n) - 1$ (le terme $b = n$ est inclus). La variante qui donne la courbe en "arcsinus" utilise $\frac{\sin\left(\frac{2\pi no}{b}\right)}{\frac{2\pi no}{b}}$: ce n'est pas cette somme.

Test 1 : histogramme de p/n . Sur $N = 200\,000$, au centre ($x = 0,5$) la densité observée vaut 0,94, alors que l'arcsinus $\frac{1}{\pi\sqrt{x(1-x)}}$ vaut 0,64 ; aux bords l'arcsinus explose (4,5 en $x = 0,005$) alors que l'observé vaut 1,6.

Erreur quadratique hors bords : arcsinus 0,23, loi plate 0,052.

En revanche, le modèle "théorème des nombres premiers seul" (le nombre de décompositions $p + q = n$ avec $p \approx xn$ est proportionnel à $\frac{n}{(\ln(xn) \ln((1-x)n))}$) donne une erreur de 0,0025 et reproduit aussi les bords (1,52 contre 1,58).

La forme en U vient donc de ce que les nombres premiers sont plus rares quand ils sont grands, pas d'un arcsinus.

(Le petit pic en 0,5 correspond très probablement aux $n = 2p$, où $p = q$.)

Test 2 : la transformée de Fourier des sin. Le signal $n \mapsto \sum_b \sum_o \frac{\sin(x)}{x}$ (figure 4) décroît à peu près comme $1/n$. La transformée de Fourier d'une suite en $1/n$ a pour module $-\ln(2 \sin(\theta/2))$: une forme en U à singularité *logarithmique*, pas en racine carrée.

Ajustement par moindres carrés du module (valeur moyenne 4995) : écart 18 pour le logarithme, 277 pour l'arcsinus.

D'où viendrait un arcsinus. Si θ est uniforme sur $[0, \pi]$, alors $\cos \theta$ suit la loi arcsinus. Dès que l'on regarde *les termes* $\cos\left(\frac{2\pi j}{b}\right)$ d'une somme de cosinus, on obtient donc un arcsinus, quelle que soit la primalité de n . De même, le spectre de $\Pi_1 + \Pi_2$ (encadré de Voiculescu) est celui de $1 + \cos \theta$: deux projecteurs en position générique donnent un arcsinus. Ce phénomène est générique ; les tests ci-dessus ne montrent pas de lien spécifique avec les nombres premiers.

Ce que cela apporte. La leçon utile est la méthode : comparer une forme observée à des lois candidates explicites (ici arcsinus, plat, modèle des nombres premiers) plutôt qu'à l'œil. Ici le modèle $\frac{1}{\ln}$ gagne largement, et il est déjà "dans" la répartition des nombres premiers.

10. Jusqu'à 10^{10} : un gain réel mais qui s'efface

Protocole. Même test que ci-dessus, mais pour chaque puissance $N = 10^e$ on tire 240 blocs de 5 000 nombres consécutifs (crible segmenté) : 160 dans $[N/2, 0,75N)$ pour l'apprentissage et 80 dans $[0,8N, N)$ pour le test, donc sans aucun nombre commun ; fenêtre de 64 bits, nombres premiers avec 60 seulement (programme 5 en annexe, environ 30 secondes en tout).

e	densité	$P=5$	$P=7$	modèle	gain sur $P=5$	part de $5 \rightarrow 7$
6	0,272	0,5848	0,5353	0,5815	$+0,0033 \pm 0,0004$	7 %
7	0,234	0,5440	0,5024	0,5415	$+0,0024 \pm 0,0003$	6 %
8	0,204	0,5064	0,4708	0,5046	$+0,0019 \pm 0,0002$	5 %
9	0,182	0,4736	0,4425	0,4724	$+0,0012 \pm 0,0002$	4 %
10	0,163	0,4442	0,4166	0,4432	$+0,0010 \pm 0,0001$	4 %

Colonnes $P = 5$, $P = 7$ et modèle :

perte sur les données de test (meilleur des deux modèles). La part de $5 \rightarrow 7$ est la fraction de l'écart entre les références $P = 5$ et $P = 7$ récupérée par le modèle. Les écarts indiqués sont des erreurs types calculées sur les 80 blocs de test.

Lecture.

- Les gains sont réels : chacun vaut de 5 à 10 fois son erreur type, y compris à 10^{10} . Mais ils sont minuscules.
- Ils diminuent lentement quand N grandit : de 0,0033 à 10^6 à 0,0010 à 10^{10} , la part de l'écart vers "on connaît $n \bmod 7$ " récupérée passe de 7 % à 4 %. Une explication plausible est la raréfaction des nombres premiers : à 10^{10} seuls 16 % des candidats sont premiers (contre 28 % à 10^6), donc chaque bit de la fenêtre dit moins de choses sur n . Aucune loi simple en $\ln N$ n'a été trouvée.
- Le modèle reste très loin de la référence $P = 7$: une fenêtre de 64 bits ne permet presque pas d'exploiter le crible au-delà de 2, 3, 5.

Ce que cela ne montre pas. Cela ne prouve pas que les nombres premiers sont "aléatoires" : ils sont entièrement déterminés par les restes modulo les petits nombres premiers, mais ces restes sont hors de portée d'une fenêtre de 64 bits. Le résultat est seulement qu'un apprentissage à courte portée ne voit presque rien de plus que le crible par 2, 3, 5. La section 12 montre que lire beaucoup plus de bits change nettement ce constat.

11. La version du programme calculant la somme des diviseurs par des sommes de cosinus

Voir la figure 3.

Le signal. Dans cette version, le signal est $n \mapsto \sum_{b=2}^{99} \sum_{o=1}^b \cos\left(\frac{2\pi no}{b}\right)$, avec la valeur $\sum_b b = 4949$ en $n = 0$. Comme $\sum_{o=1}^b \cos\left(\frac{2\pi no}{b}\right)$ vaut b si $b \mid n$ et 0 sinon, le signal vaut exactement $\sigma(n) - 1$ pour $1 \leq n \leq 99$ (vérifié numériquement, écart $< 10^{-11}$; maximum 251 atteint en $n = 96$). La courbe du haut est donc celle de la somme des diviseurs, en dents de scie autour d'une tendance croissante.

Le spectre.

- Le plancher plat à environ 5000 sur la courbe du bas vient du terme $n = 0$: une impulsion de hauteur 4949 a une transformée de Fourier constante. Sans ce terme, le module médian tombe à 247 (contre 4865 avec lui).
- Une fois l'impulsion retirée, le spectre n'est pas en forme d'arcsinus : un plateau bas, des valeurs élevées aux deux bords (qui reflètent la tendance croissante de $\sigma(n)$) et des pics isolés aux fréquences $k/100$ avec $k = 33, 50, 67$, c'est-à-dire aux fractions $1/3, 1/2, 2/3$ (modules 792, 1877, 792).
- Ces pics aux fractions de petit dénominateur sont ceux que prévoit la formule de Ramanujan,

jan, $\sigma(n) = \zeta(2) n \sum_{q \geq 1} \frac{\sum_{\substack{a \bmod q \\ (a,q)=1}} \cos\left(\frac{2\pi an}{q}\right)}{q^2}$ dans laquelle intervient la somme de Ramanujan $\sum_{\substack{a \bmod q \\ (a,q)=1}} \cos\left(\frac{2\pi an}{q}\right)$. Ils viennent de la divisibilité, non d'une loi de Lévy. (Nous n'avons pas vérifié quantitativement la décroissance en $\frac{1}{q^2}$.)

Conclusion. La ressemblance avec la version en $\frac{\sin(x)}{x}$ (figure 4) est celle des deux bords relevés (tendance croissante du signal) et de l'impulsion en $n = 0$; la forme en arcsinus n'apparaît pas ici.

12. Lire tous les bits précédents

L'information est dans le préfixe. Si $p \mid n$, alors tous les nombres $n - p, n - 2p, \dots$ sont multiples de p , donc composés (sauf p lui-même). Si $p \nmid n$, la progression $n - kp$ contient des nombres premiers. Lire les bits aux décalages $p, 2p, 3p, \dots$ permet donc de savoir si p divise n , sans diviser n .

Règle “peigne” : n est composé s’il existe un premier $p \leq \sqrt{n}$ (reconnu par son propre bit) tel que la progression des $m < n$ congrus à n modulo p ne contienne aucun premier hormis éventuellement p . Sur des tirages au hasard : exactitude 0,9990 pour $100 \leq n < 1000$ (trois premiers ratés sur 3000 tirages), 1,0000 pour $10^3 \leq n < 10^5$. C’est le crible d’Ératosthène lu à l’envers ; la règle utilise cependant des décalages $p, 2p, \dots$ qui dépendent de n : elle n’est pas une matrice de poids unique.

Un modèle à poids fixes qui voit de plus en plus loin. Régression logistique, une par classe $n \bmod 60$, qui lit les W derniers bits ; nombres autour de 10^8 .

W	perte test (règle $60k + a : 0,5053$)
64	0,5056
256	0,4996
1024	0,4833
2048	0,4706
4096	0,4568
8192	0,4411

Références sur les mêmes nombres :

règle $60k + a$	0,5075
$n \bmod 7$ connu	0,4718
$n \bmod 7, 11, 13$ connus	0,4300

Lecture.

- La perte baisse à chaque doublement de W (d’environ 0,013 à 0,016 par doublement à partir de 1024) et n’a pas atteint de palier à 8192 ; l’erreur d’échantillonnage attendue sur ces pertes est de l’ordre de quelques 10^{-3} .
- À $W = 2048$ le modèle atteint le niveau de la référence “ $n \bmod 7$ connu” (0,4706 contre 0,4718) et le dépasse nettement à 4096 ; à $W = 8192$ il récupère environ 86 % de l’écart entre la règle $60k + a$ et la référence “ $n \bmod 7, 11, 13$ connus” (0,0664 sur 0,0775).
- Le gain vient du crible : le modèle apprend à utiliser les décalages multiples de 7, 11, 13, $\dots$, mais seulement de façon approchée, avec un jeu de poids par classe modulo 60.

Précautions. Sans régularisation adaptée, le même modèle se dégrade à 4096 et 8192 bits (sur-apprentissage : jusqu’à 0,52, pire que la règle $60k + a$) ; c’est pourquoi C est choisi sur des blocs de validation, et il diminue quand W augmente ($C = 0,01$ pour $W \leq 256$, 0,003 pour 1024 et 2048 et 4096, 0,001 pour 8192). Les résultats dépendent aussi de la quantité de données. Ce test ne montre pas qu’une structure existe au-delà du crible : il montre qu’avec assez de bits et de données, un modèle simple réapprend le crible par les petits nombres premiers.

13. Une marche $4k + 1$ versus $4k + 3$ sur les décompositions de Goldbach

Protocole (programme 7 en annexe). Pour chaque pair n de 6 à $N_{\max}$, on liste ses décompositions $n = p_1 + p_2$ ($p_1 \leq p_2$ premiers impairs) par p_1 croissant, et on construit une marche : pas

+1 si le premier considéré est $\equiv 1 \pmod{4}$, pas -1 s'il est $\equiv 3 \pmod{4}$.

Trois variantes : **V1**, un pas par décomposition donné par p_1 ; **V2**, un pas par décomposition donné par p_2 ; **V3**, deux pas par décomposition (p_1 puis p_2).

Pour chaque n ayant au moins 10 décompositions, on calcule la proportion de temps où la marche est positive (convention de Feller : le pas k compte si $S_k > 0$, ou si $S_k = 0$ et $S_{k-1} > 0$).

Les marches sont courtes (10 à 52 pas pour $n \leq 1000$, 10 à 329 pour $n \leq 10^4$), et même une marche à pile ou face de cette longueur s'écarte de la loi de l'arcsinus $F(x) = \frac{2}{\pi} \arcsin \sqrt{x}$: on compare donc aussi à un *témoin*, des marches de même longueur dont les pas sont des tirages indépendants ± 1 équiprobables.

Pourquoi V3 est dégénérée. Si $n \equiv 0 \pmod{4}$, les deux premiers p_1 et p_2 de chaque décomposition sont de classes opposées (1 et 3 modulo 4) : chaque décomposition donne $+1, -1$ ou $-1, +1$ et la marche reste entre -1 et 1 . Si $n \equiv 2 \pmod{4}$, ils sont de même classe. V3 mélange donc deux régimes, ce qui crée un pic autour de 0,5 sans rapport avec l'arcsinus.

Résultats.

$N_{\max}$	variante	moy. positive	dist. obs./témoin	chgts de signe
1000 (371 n)	V1	0,345	0,195	0,596
	V2	0,472	0,055	0,596
	V3	0,401	0,292	0,501
10^4 (4871 n)	V1	0,374	0,149	0,575
	V2	0,491	0,022	0,575
	V3	0,428	0,287	0,501

Entre parenthèses : nombre de pairs n retenus. “moy. positive” : proportion moyenne de temps positif. “dist. obs./témoin” : distance de Kolmogorov entre la répartition observée et celle du témoin. “chgts de signe” : proportion de changements de signe entre pas successifs (0,5 si indépendants).

Histogramme de V2 pour $N_{\max} = 10^4$ (10 classes, densités) :

arcsinus	2,05	0,90	0,74	0,67	0,64	0,64	0,67	0,74	0,90	2,05
témoin	2,12	0,87	0,74	0,63	0,62	0,67	0,64	0,70	0,88	2,12
V2 observé	2,12	0,90	0,72	0,63	0,66	0,69	0,73	0,74	0,81	1,99

Lecture.

- **V2** se comporte comme une marche à pile ou face : proportion moyenne proche de 0,5, histogramme presque identique à celui du témoin (distance 0,022 à $N_{\max} = 10^4$), donc proche de l'arcsinus.
- **V1** est décalée vers le négatif (0,37 du temps positif) : les petits premiers, par lesquels commence la marche, sont plus souvent $\equiv 3$ que $\equiv 1 \pmod{4}$ (biais de Chebyshev). C'est un déséquilibre réel, mais il n'a pas la forme de l'arcsinus.

- Les signes successifs alternent plus souvent que des tirages indépendants (57,5 % de changements au lieu de 50 % ; 59,6 % jusqu'à 1000). Cet écart dépasse largement l'erreur d'échantillonnage naïve (les marches de paires différents réutilisent les mêmes premiers, donc cette erreur est sous-estimée) ; il n'a pas d'effet visible sur l'histogramme de V2.

Ce que cela montre, et ce que cela ne montre pas. La loi de l'arcsinus est celle d'une marche *équilibrée* : le temps passé au-dessus de 0 est alors proche de 0 ou de 1 plutôt que de $\frac{1}{2}$. La retrouver pour V2 est donc compatible avec l'idée d'un hasard équilibré sur les classes $4k + 1$ et $4k + 3$ des décompositions de Goldbach, sans en être une preuve ; deux écarts mesurables restent (le biais de V1 et l'alternance des signes).

Cela n'explique pas l'histogramme de p/n : celui-ci n'est pas un arcsinus (voir plus haut), et la marche est un autre objet.

14. Annexe : résultats des programmes

Résultats du programme 1 ici <https://denisevellachemla.eu/pgmarcsin1.pdf> :

```
N=10^2: 25 premiers (OK), 0.00s, 13 octets
N=10^3: 168 premiers (OK), 0.01s, 125 octets
N=10^4: 1229 premiers (OK), 0.00s, 1250 octets
N=10^5: 9592 premiers (OK), 0.00s, 12500 octets
N=10^6: 78498 premiers (OK), 0.00s, 125000 octets
N=10^7: 664579 premiers (OK), 0.06s, 1250000 octets
100 premiers bits :
01101010001010001010001000001010000010001010001000
00100000101000001000101000001000100000100000001000
total : 0.1s
```

Résultats du programme 2 ici <https://denisevellachemla.eu/pgmarcsin2.pdf> :

```
N=10^6, W=64
Ref. densite seule (1/ln n) : perte test = 0.2628
Ref. connaitre n mod 60 : perte test = 0.1575
Regression logistique (fenetre): perte test = 0.1946 (6.2s)
Decalages les plus influents (lag : poids) :
lag 41 : -1.478
lag 29 : -1.475
lag 23 : -1.471
lag 35 : -1.460
lag 59 : -1.455
lag 39 : -1.449
lag 47 : -1.447
lag 17 : -1.447
lag 37 : -1.443
lag 5 : -1.441
```

```
lag 53 : -1.438
lag 19 : -1.431
Graphique : coeffs_lags.png
```

Enfin, ci-dessous, les coefficients appris.

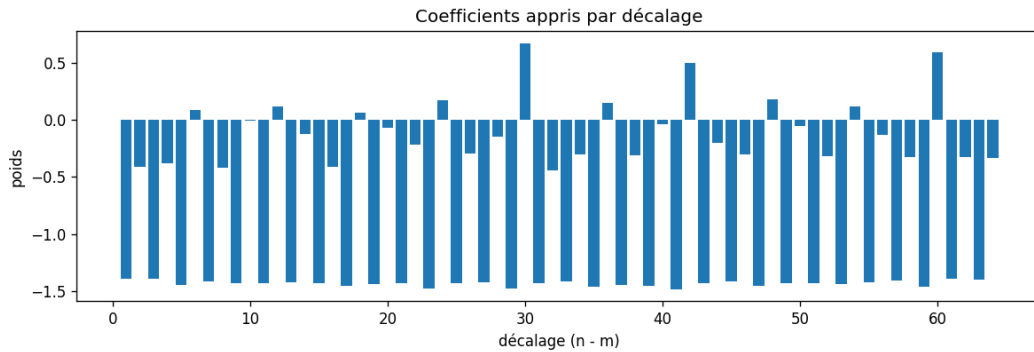


FIGURE 6

Résultats du programme 3 ici <https://denisevellachemla.eu/pgmarcsin3.pdf> :

```
N=107, W=64, train=300000, test=150000, densite test=0.2355
```

Oracles de crible (connaissent $n \bmod q$ pour $q \leq P$) :

```
P= 5 : perte test = 0.5459
P= 7 : perte test = 0.5039
P=11 : perte test = 0.4773
P=13 : perte test = 0.4547
P=17 : perte test = 0.4372
P=19 : perte test = 0.4217
P=23 : perte test = 0.4084
P=29 : perte test = 0.3977
P=31 : perte test = 0.3879
```

Regression logistique par classe : perte test = 0.5439 (1s)

Decalages les plus influents (somme $|\text{poids}|$ sur les 16 classes) :

```
42, 30, 60, 12, 18, 48, 36, 54, 6, 24, 28, 14
```

```
part du poids sur les decalages multiples de 7 : 0.201 (uniforme : 0.141)
part du poids sur les decalages multiples de 11 : 0.034 (uniforme : 0.078)
part du poids sur les decalages multiples de 13 : 0.026 (uniforme : 0.062)
part du poids sur les decalages multiples de 17 : 0.006 (uniforme : 0.047)
part du poids sur les decalages multiples de 19 : 0.010 (uniforme : 0.047)
part du poids sur les decalages multiples de 23 : 0.008 (uniforme : 0.031)
part du poids sur les decalages multiples de 29 : 0.016 (uniforme : 0.031)
part du poids sur les decalages multiples de 31 : 0.005 (uniforme : 0.031)
```

Arbres de gradient : perte test = 0.5437 (9s)

Arbres + $n \bmod 7, 11, 13$ (controle) : perte test = 0.4563 (oracle $P=13$: 0.4547) (7s)

total 17s

Résultats du programme 4 ici <https://denisevellachemla.eu/pgmarcsin4.pdf> :

```
(1) ecart max |double somme de cos - (sigma(n)-n-1)|
    pour n<=200 : 2.73e-11
(2) N=200000, centre (x=0,5) : empirique 0.941 | arcsinus 0.637 |
    modele PNT 0.931
    erreur quadratique (hors bords) : arcsinus 0.230 | plat 0.052 |
    modele PNT 0.0025
    bord (x=0,005) : empirique 1.58 | arcsinus 4.51 |
    modele PNT 1.52
(3) FFT de flute3 : ecart de lissage par moindres carres      log 18.3 |
    arcsinus 277.3 (module moyen 4995)
```

Résultats du programme 5 ici <https://denisevellachemla.eu/pgmarcsin5.pdf> :

```
10^6: 320010 candidats , 12s
10^7: 320011 candidats , 4s
10^8: 319994 candidats , 3s
10^9: 320008 candidats , 4s
10^10: 319983 candidats , 5s

  e  densite   P=5    P=7    P=13    P=31    logist.  arbres    gain/P=5 (+-err typ)
gain/(P5-P7)
  6  0.2715   0.5848   0.5353   0.4768   0.3964   0.5815   0.5817   +0.0033   0.0004
+0.07
  7  0.2339   0.5440   0.5024   0.4537   0.3876   0.5417   0.5415   +0.0024   0.0003
+0.06
  8  0.2044   0.5064   0.4708   0.4292   0.3734   0.5049   0.5046   +0.0019   0.0002
+0.05
  9  0.1815   0.4736   0.4425   0.4063   0.3578   0.4727   0.4724   +0.0012   0.0002
+0.04
 10  0.1628   0.4442   0.4166   0.3846   0.3420   0.4436   0.4432   +0.0010   0.0001
+0.04

total 29s
```

Résultats du programme 6 ici <https://denisevellachemla.eu/pgmarcsin6.pdf> :

```
Partie A : regle ‘‘peigne’’ sur le prefixe complet
n dans [100,1000) : exactitude 0.9990 ; premiers rates 3,
  faux premiers 0 (sur 3000 tirages , 463 premiers)
n dans [1000,10000) : exactitude 1.0000 ; premiers rates 0,
  faux premiers 0 (sur 3000 tirages , 352 premiers)
n dans [10000,100000) : exactitude 1.0000 ; premiers rates 0,
  faux premiers 0 (sur 3000 tirages , 287 premiers)
(1s)

Partie B : regression logistique a poids fixes , nombres autour de 10^8
```

(C, la force de regularisation, est choisi sur des blocs de validation separees du test)

```
W= 64 : perte test = 0.5056 (C choisi = 0.01)
[regle 60k+a 0.5075 |
 n mod 7 connu 0.4718 | n mod 7,11,13 connus 0.4300] (3s)
W= 256 : perte test = 0.4996 (C choisi = 0.01)
[regle 60k+a 0.5075 |
 n mod 7 connu 0.4718 | n mod 7,11,13 connus 0.4300] (5s)
W= 1024 : perte test = 0.4833 (C choisi = 0.003)
[regle 60k+a 0.5075 |
 n mod 7 connu 0.4718 | n mod 7,11,13 connus 0.4300] (17s)
W= 2048 : perte test = 0.4706 (C choisi = 0.003)
[regle 60k+a 0.5075 |
 n mod 7 connu 0.4718 | n mod 7,11,13 connus 0.4300] (29s)
W= 4096 : perte test = 0.4568 (C choisi = 0.003)
[regle 60k+a 0.5075 |
 n mod 7 connu 0.4718 | n mod 7,11,13 connus 0.4300] (47s)
W= 8192 : perte test = 0.4411 (C choisi = 0.001)
[regle 60k+a 0.5075 |
 n mod 7 connu 0.4718 | n mod 7,11,13 connus 0.4300] (109s)
```

Résultats du programme 7 ici <https://denisevellachemla.eu/pgmarcsin7.pdf> :

n pairs de 6 a 1000, au moins 10 decompositions :
 371 valeurs de n (longueur de 10 a 52 decompositions)
 Histogramme (10 classes, densite). Arcsinus exact :
 2.05 0.90 0.74 0.67 0.64 0.64 0.67 0.74 0.90 2.05

V1 : marche positive en moyenne 0.345 du temps
 observe : 3.45 1.21 0.92 0.62 0.70 0.57 0.40 0.38 0.67 1.08
 temoin : 2.20 0.91 0.72 0.56 0.59 0.72 0.57 0.57 0.96 2.20
 distance de Kolmogorov : observe/arcsinus 0.307 |
 temoin/arcsinus 0.187 | observe/temoin 0.195
 changements de signe entre pas successifs : 0.596
 (0,500 si independants)

V2 : marche positive en moyenne 0.472 du temps
 observe : 2.48 1.11 0.46 0.62 0.70 0.57 0.40 0.81 0.81 2.05
 temoin : 2.18 0.88 0.72 0.55 0.61 0.72 0.58 0.57 0.97 2.22
 distance de Kolmogorov : observe/arcsinus 0.213 |
 temoin/arcsinus 0.185 | observe/temoin 0.055
 changements de signe entre pas successifs : 0.596
 (0,500 si independants)

V3 : marche positive en moyenne 0.401 du temps
 observe : 1.99 0.67 0.43 0.70 2.59 2.16 0.40 0.22 0.24 0.59
 temoin : 2.16 0.81 0.76 0.59 0.58 0.73 0.63 0.64 0.83 2.27
 distance de Kolmogorov : observe/arcsinus 0.290 |
 temoin/arcsinus 0.133 | observe/temoin 0.292
 changements de signe entre pas successifs :
 0.501 (0,500 si independants)