ChatGPT

Synthèse sur nombres premiers, flottants, p-adiques, théorie de Galois et calcul parallèle

Introduction

Les systèmes de nombres et les structures algébriques jouent un rôle fondamental à la fois en mathématiques pures et en calcul numérique. Cette synthèse explore un éventail de thèmes interconnectés : la représentation informatique des très grands nombres premiers et la complexité formelle de leur reconnaissance, la structure des nombres flottants et son incidence sur l'approximation des fonctions utilisées en intelligence artificielle, la théorie et l'implémentation des nombres *p*-adiques – ces complétions « duales » des nombres réels –, la théorie de Galois et la classification des groupes finis (notamment via les groupes linéaires sur les corps finis), et enfin une perspective prospective sur la compilation dynamique orientée par le choix d'un corps de calcul (réel vs *p*-adique) pour optimiser le parallélisme dans les calculs massifs sur GPU. Chaque partie met en lumière les liens structurels entre ces notions, en s'appuyant sur des références récentes et des exemples concrets (par exemple, motifs digitaux, diagrammes d'actions de groupes). Nous conclurons sur les perspectives offertes par ces rapprochements interdisciplinaires, et proposons en annexe un rappel des notations principales.

Partie I : Représentation des très grands nombres premiers en machine

La manipulation de nombres premiers extrêmement grands est courante en cryptographie et en théorie des nombres computationnelle. En pratique, ces entiers sont souvent représentés en base 2 (binaire) ou base 16 (hexadécimal) pour s'aligner sur l'architecture binaire des ordinateurs. Par exemple, un entier de 1024 bits sera typiquement affiché comme une chaîne hexadécimale de 256 caractères. Cependant, même exprimés en base 16, les nombres premiers *n* semblent présenter des chiffres « aléatoires » sans motif simple. Des travaux de théorie des nombres ont confirmé l'absence de biais évident dans les motifs digitaux des nombres premiers : Mauduit et Rivat (2010) ont ainsi prouvé que, pour les nombres premiers en base 10, la somme des chiffres est paire aussi souvent qu'elle est impaire [1]. En d'autres termes, la parité de la somme des chiffres d'un nombre premier est essentiellement aléatoire, suggérant qu'aucune propriété simple des représentations décimales (ou hexadécimales) ne distingue les nombres premiers des composés.

Cette imprévisibilité s'accompagne d'une complexité structurelle du langage des nombres premiers, du point de vue des automates finis. Un résultat marquant de Hartmanis & Shank (1968) a montré que l'ensemble des nombres premiers (exprimés en base fixée, par ex. base 2 ou 10) **n'est pas un langage régulier** : tout automate fini déterministe reconnaissant la primalité devrait avoir un nombre d'états au moins exponentiel en la longueur de l'entrée [2] [3]. Intuitivement, il n'existe pas de « motif local » simple dans les chiffres qui caractérise un nombre premier – tester la primalité exige de la mémoire croissant avec la taille du nombre. Des travaux ultérieurs ont renforcé ce constat : Shallit (1996) a introduit la notion d'**automaticité** et montré que la taille minimale d'un automate approchant le langage des nombres premiers croît exponentiellement [4]. Plus récemment, Fijalkow (2019) a étendu ces résultats à des modèles plus puissants (automates alternants) en établissant qu'en base 2, tout automate alternant reconnaissant les nombres premiers requiert un nombre d'états au moins linéaire

en la longueur du nombre [5]. Cela confirme que le *langage* des nombres premiers possède une complexité intrinsèque élevée, échappant aux descriptions par automates finis de taille raisonnable.

Malgré l'absence de motifs réguliers simples, certaines **familles spéciales de nombres premiers** exhibent des représentations particulières qui ont été étudiées. Par exemple, les *nombres de Mersenne*, de la forme $2^p - 1$, s'écrivent en binaire comme une suite de 1 répétés (ex. $31_{10} = 11111_2$) et les rares cas où ils sont premiers permettent un test spécifique (test de Lucas-Lehmer) [6]. De même, les *répunits* (tous les chiffres égaux à 1 en base 10, comme 11111) ont donné lieu à des recherches : quelques-uns sont premiers et leur structure décimale induit des critères de divisibilité élégants. En base 16, un nombre composé uniquement du chiffre F (15 en décimal) a la forme $16^k - 1$; s'il est divisible par un premier p, cela implique $16^k \equiv 1 \pmod p$, reliant ainsi la représentation hexadécimale à l'ordre multiplicatif de 16 modulo p [7] [8]. On définit aussi des *premiers à période maximale* (ou **premiers full reptend**) par la longueur de la période de $1/p$ en base donnée. Un premier p est *full reptend* en base b si la période décimale de $1/p$ est $p-1$ (valeur maximale). Ceci équivaut à dire que b est une racine primitive modulo p. Par exemple, $p=7$ est full reptend en base 10 car $1/7 = 0.\overline{142857}$ a une période de 6 chiffres = $7-1$ [9]. En base 16 en revanche, aucun premier impair n'a de période $p-1$ car 16 est une puissance parfaite (2^4) : on peut montrer que la période hexadécimale de $1/p$ divise au plus $(p-1)/2$ [10]. Ainsi, *16 n'est pas racine primitive modulo p*, ce qui limite la longueur des répétends hexadécimaux. Ces propriétés soulignent que la base de représentation influence la structure des fractions $1/p$ et renvoie à des notions profondes d'algèbre modulaire (racines primitives, ordres multiplicatifs). En résumé, en machine, les très grands nombres premiers sont représentés en binaire/hexadécimal sans schéma visible simple ; et cette « pseudo-aléatoire » des digits est corroborée tant par des résultats statistiques (parité de la somme des chiffres aléatoire) que par des résultats de complexité montrant l'absence de pattern reconnaissable par un automate fini polynomial [2] [11].

Partie II : Nombres flottants et fonctions d'activation

Les **nombres flottants** constituent le format standard pour représenter les réels en machine, du moins approximativement. Un flottant (selon la norme IEEE 754 en double précision, par exemple) consiste en un signe, un exposant sur 11 bits et une mantisse sur 52 bits, représentant une valeur sous la forme $\pm 1.mantisse \times 2^{exposant}$ dans une certaine plage. Ce système produit une grille discrète de nombres rationnels dyadiques (dénominateurs puissances de 2) répartis de manière logarithmique sur la ligne réelle. La structure même des flottants a plusieurs implications pour la représentation des fonctions mathématiques en machine. D'une part, la précision est limitée à ~15–16 chiffres décimaux pour les doubles, ce qui introduit des erreurs d'arrondi et pose le problème de la *stabilité numérique* des algorithmes d'inférence numérique. D'autre part, la distribution non uniforme des flottants (plus dense autour de 0, plus lâche pour les grandes magnitudes) reflète la nature **logarithmique** de l'échelle de représentation : cela se connecte au fait que certaines fonctions analytiques peuvent être mieux ou moins bien approchées selon la métrique considérée (euclidienne vs logarithmique).

Pour calculer efficacement les fonctions couramment utilisées en **inférence et apprentissage automatique** (fonctions d'activation comme $\operatorname{ReLU}(x)=\max(0,x)$, sigmoïdes, $\tanh(x)$, etc.), on exploite le fait que beaucoup sont *holomorphes* (au moins par morceaux ou après une certaine transformation) et peuvent donc être approchées par des polynômes ou des séries. Par exemple, $\tanh(x)$ est infiniment dérivable et possède un développement en série de Taylor (ou de Laurent) convergent sur son domaine analytique. En pratique, les implémentations logicielles des fonctions mathématiques (exponentielle, trigonométriques, hyperboliques…) recourent à des **approximations polynomiales ou rationnelles** sur des intervalles réduits, combinées à des identités (par ex. formules d'angle double) pour couvrir $\mathbb{R}$. Ces polynômes approchent la fonction avec une précision de l'ordre de la machine. Un point clé d'analyse est la **densité des polynômes**

trigonométriques dans l'espace des fonctions utilisées. Un théorème classique d'approximation de Weierstrass stipule que tout polynôme trigonométrique (somme finie de termes $\cos(nx), \sin(nx)$) peut approcher uniformément toute fonction continue 2π-périodique [12]. Autrement dit, les polynômes trigonométriques sont denses dans $C([0,2\pi])$. Dans le contexte de l'**approximation des fonctions d'activation**, cela signifie par exemple que pour toute fonction continue convenable sur un intervalle fixé (et a fortiori toute fonction lisse sur $\mathbb{R}$ que l'on peut périodiser ou borner à un domaine d'intérêt), on peut trouver une combinaison de sinusoïdes qui l'approxime aussi finement que désiré. Ainsi, les fonctions $\tanh(x)$ ou même $\operatorname{ReLU}(x)$ (qui est continue par morceaux, bien qu'ayant un coin en 0) peuvent être approchées par des **séries de Fourier** tronquées ou par des polynômes (de Chebyshev, par exemple) sur un intervalle donnée. En pratique, cette densité justifie l'usage de polynômes trigonométriques pour approximer les fonctions d'activation dans certaines analyses théoriques ou même pour accélérer les calculs sur hardware spécialisé (approximations matérielles). Par ailleurs, la connexion entre séries trigonométriques et réseaux de neurones est un sujet d'étude en soi : une *couche de neurones* à activation sinus ou cosinus pourrait approximativement synthétiser n'importe quelle fonction cible – on rejoint ici l'idée des réseaux de Fourier. Bien que la plupart des réseaux utilisent des activations plus simples (ReLU, etc.), l'idée qu'une combinaison linéaire de fonctions « simples » (polynômes, ondes trigonométriques) peut approximer des fonctions complexes est au cœur de l'**universalité des réseaux de neurones**. En résumé, la structure des flottants impose de travailler avec des approximations polynomiales finies des fonctions continues, et les théorèmes d'approximation (Weierstrass, Stone) garantissent qu'avec suffisamment de degrés ou de termes trigonométriques on peut s'approcher arbitrairement bien des fonctions d'activation usuelles – moyennant un coût calculatoire potentiellement élevé, géré par la précision machine disponible.

Partie III : Nombres *p*-adiques – arithmétique et infrastructure logicielle

Les **nombres *p*-adiques** (pour un nombre premier fixé p) forment un système numérique alternatif aux nombres réels, fournissant une autre complétion du corps des rationnels $\mathbb{Q}$. Là où $\mathbb{R}$ complète $\mathbb{Q}$ selon la valeur absolue usuelle (distance euclidienne), $\mathbb{Q}*p$ est une complétion selon la distance p-adique, définie à partir de la valuation p-adique $v_p(x)$ (exposant de p dans la factorisation de x). Concrètement, tout nombre p-adique se représente par une série formelle de base p s'étendant à l'infini vers la gauche au lieu de vers la droite* [13]*. Par exemple en base 10 (donc $p=2$ ou 5 non respecté, mais pour l'illustration) un nombre 10-adique s'écrirait $\ldots a_3a_2a_1a_0$ (chiffres infiniment nombreux à gauche). Plus rigoureusement, on définit $\mathbb{Z}_p$, l'anneau des entiers p-adiques, comme la limite projective des anneaux $\mathbb{Z}/p^n\mathbb{Z}$ quand $n\to\infty$* [14]*. Intuitivement, connaître un nombre p-adique, c'est connaître son reste modulo p, modulo p^2, modulo p^3, et ainsi de suite – on affine indéfiniment sa précision comme un zoom vers les chiffres de poids faibles (multiples de grandes puissances de p)* [15] [16]*. Deux nombres sont très proches p-adiquement s'ils coïncident sur de nombreux chiffres de bas poids (c'est-à-dire si leur différence est divisible par une grande puissance de p)* [16]*. Cette notion de distance non-archimédienne entraîne des propriétés contre-intuitives : par exemple, en métrique 10-adique la suite de rationnels $1, 1.4, 1.41, 1.414, \ldots$ (approximations successives de $\sqrt{2}$) diverge car les différences finissent par ne plus être divisibles par des puissances croissantes de 10, tandis que la suite $1, 11, 111, 1111, \ldots$ (qui diverge dans $\mathbb{R}$) converge dans $\mathbb{Q}$* [17] [18]. Ces exemples illustrent que les }$ puisque $111\ldots1$ avec n chiffres 1 diffère de la limite $…111_{10}$ seulement de 10^n [17]. De même, un entier négatif comme -1 s'écrit en base 5 (p. ex.) comme $…44445$ car $-1 \equiv 4 \pmod 5$, $-1 \equiv 24 \pmod{25}$, $-1 \equiv 124 \pmod{125}$, etc., produisant une expansion $(-1) = \ldots 44444_{(5)}**rationnels ont des développements périodiques vers la gauche** en base p (là où en base réelle 10 ils ont une expansion périodique vers la droite pour les fractions). Les

entiers relatifs, eux, apparaissent comme des *p*-adiques dont les chiffres finissent par devenir nuls (pour les positifs) ou une constante (pour les négatifs), ce qui garantit une représentation unique sans ambigüité de signe (pas de « -0 » distinct) [19] [20] .

D'un point de vue informatique, manipuler un élément de $\mathbb{Q}_p$ nécessite de tronquer son développement à une certaine précision finie N (nombre de chiffres *p*-adiques conservés). Typiquement, on représentera un nombre *p*-adique par son résidu modulo p^N (mantisse) et un exposant de valuation. Les bibliothèques modernes adoptent cette approche pour efficacité : par exemple FLINT (Fast Library for Number Theory, en C) définit un type `padic_t` stockant un entier u (la mantisse, non divisible par p) et un exposant v (la valuation), avec une précision maximale N fixée [21] . Ainsi x est représenté sous la forme $x = p^v \cdot u$, avec u stocké modulo p^N (typiquement sous forme d'un entier machine ou multiprécision) [22] [23] . La bibliothèque Nemo.jl (Julia) bâtie sur FLINT suit le même schéma : on peut construire $\mathbb{Q}_p$ pour n'importe quel p et une précision désirée, et chaque élément est manipulé comme un objet comportant mantisse (entier modulo p^N) + valuation + méta-données de précision [24] [25] . Les opérations ($+$, $\times$, inverses, etc.) sont alors implémentées via de l'arithmétique modulaire rapide sur ces représentants entiers modulaires, en ajustant dynamiquement la précision en cas de besoin (par exemple lors d'inversions ou d'opérations sensibles à la précision) [26] . Cette stratégie « zélée » (eager) garantit un coût mémoire constant dépendant seulement de N, et non de la magnitude de l'opérande *p*-adique, par opposition à une implémentation naïve par tableau de chiffres [27] . À l'inverse, on peut imaginer des algorithmes *paresseux* (lazy) où la précision *p*-adique est augmentée à la volée seulement si nécessaire au cours d'un calcul – on parle aussi d'approches *relaxées*. Des recherches existent sur de tels algorithmes *p*-adiques paresseux pour optimiser les calculs sans sacrifier la exactitude [28] . Par exemple, un calcul symbolique *p*-adique pourrait déterminer qu'un résultat final est exact à certaines puissances de p près sans avoir à pousser toutes les opérations à la précision maximale a priori [29] . Dans tous les cas, l'objectif est de préserver l'exactitude arithmétique (pas d'erreur d'arrondi, puisque l'arithmétique *p*-adique est exacte tant qu'on reste dans la précision tronquée choisie) en exploitant la robustesse des opérations mod p^N.

Les nombres *p*-adiques commencent à trouver des applications en calcul formel et même en apprentissage. En calcul formel, ils sont utilisés via le *le lifting de Hensel* pour résoudre des équations polynomiales modulo des puissances croissantes de p : cela permet par exemple de factoriser des polynômes à coefficients entiers ou de trouver des solutions congruentes modulo p^n. Ils ont été formalisés dans des assistants de preuve comme Coq ou Lean (y compris une preuve formelle du lemme de Hensel en 2019) [30] [31] . Du côté de l'**apprentissage automatique**, les nombres *p*-adiques offrent un cadre non-archimédien potentiellement intéressant pour modéliser des hiérarchies ou des distributions ultramétriques dans les données. Des travaux récents (Zúñiga-Galindo et al., 2021) ont proposé des réseaux de neurones cellulaires *p*-adiques, où les neurones sont organisés selon une arborescence infinie reflétant l'arbre des expansions *p*-adiques [32] . L'idée est que les réseaux neuronaux *p*-adiques peuvent approcher le comportement de réseaux hiérarchiques de profondeur et largeur croissantes, en exploitant la structure ultramétrique pour coder efficacement l'information. Bien que ces recherches en soient au stade théorique, elles montrent la **pertinence potentielle des *p*-adiques pour des calculs d'inférence** non conventionnels, en particulier lorsque les données ou les structures à apprendre présentent des relations hiérarchiques (arbres, clusters emboîtés). Par ailleurs, dans le domaine du calcul haute performance, l'arithmétique *p*-adique modulaire est étudiée pour améliorer la fiabilité et le parallélisme des algorithmes. En effet, effectuer des calculs en module p^N peut permettre d'éviter les erreurs d'arrondi liées aux flottants en gardant la **partie entière exacte** (pas de perte de carrés ou annulations numériques) [33] . Une approche dite *Hensel lifting* ou *Hensel code* consiste à réaliser des calculs numériques en utilisant des expansions *p*-adiques modulo plusieurs puissances de p et à recoller les résultats. Des comparatifs ont montré que ces méthodes non-archimédiennes peuvent être très prometteuses en termes de stabilité et parallélisation : on peut

appliquer indépendamment plusieurs opérations modulo p^k en parallèle sur des architectures modernes, puis reconstruire le résultat exact [33] [34]. Ainsi, les *p*-adiques offrent un terrain fertile pour innover tant du point de vue algorithmique (calcul exact, parallélisme) que pour d'éventuelles nouvelles architectures de réseaux (où la notion de proximité n'est plus Euclidienne mais ultramétrique).

Partie IV : Théorie de Galois : extensions finies, corps de rupture et groupes de Galois

La **théorie de Galois** étudie les symétries des solutions d'équations polynomiales au travers des *extensions de corps*. Une *extension finie* E/K est une extension de corps où E est un espace vectoriel de dimension finie sur K. Lorsqu'un polynôme à coefficients dans K est *séparable* et que l'on adjoint à K toutes ses racines, on obtient une extension particulière appelée **corps de rupture** (ou corps de décomposition) du polynôme. Une extension E/K est dite **galoisienne** si elle est à la fois séparable et normale, ce qui équivaut à ce que E soit le corps de rupture d'un certain polynôme de $K[X]$. Dans ce cas, l'ensemble des automorphismes de E qui fixent K (les bijections de E préservant toutes les éléments de K) forme un groupe fini, appelé **groupe de Galois** de l'extension, noté $\mathrm{Gal}(E/K)$. Ce groupe de Galois capture l'essence des symétries entre les différentes racines des polynômes dont E est le corps de décomposition.

Un exemple canonique est celui des **corps finis**. Pour tout nombre premier p et tout entier $n\ge 1$, il existe un unique (à isomorphisme près) corps fini à p^n éléments, noté $\mathbb{F}*{p^n}$ ou $\mathrm{GF}(p^n)$* [35] *. Ce corps peut s'obtenir comme corps de rupture d'un polynôme irréductible de degré n sur $\mathbb{F}_p$ (par exemple un polynôme primitif de degré n dans $\mathbb{F}_p[X]$). L'extension $\mathbb{F}$* / \mathbb{F*p$ est alors galoisienne de degré n. Son groupe de Galois est engendré par l'application **Frobenius** $F:x\mapsto x^p$, et est isomorphe au groupe cyclique d'ordre n* [36] *. En effet, appliquer Frobenius n fois revient à élever à p^n, or pour tout $x\in \mathbb{F}$*$ on a $x^{p^n} = x$ (par le petit théorème de Fermat généralisé), donc $F^n = \mathrm{Id}$ et l'ordre de F divise n. Inversement, F est d'ordre exactement n car $\mathbb{F*{p^n}$ est le plus petit corps contenant $\mathbb{F}_p$ dans lequel $x^{p^n}=x$ pour tous x. Ainsi $\mathrm{Gal}(\mathbb{F}*p)\cong C_n$ (cyclique). Ce fait illustre un lien structurel fort : les }/\mathbb{F**corps finis** et plus généralement les extensions finies fournissent un vaste réservoir de **groupes finis** via leurs groupes de Galois, souvent des groupes cycliques ou plus complexes selon la nature de l'extension.

Dans le cas des extensions de corps de nombres (extensions finies de $\mathbb{Q}$, comme $\mathbb{Q}(\sqrt[d]{a})$, corps cyclotomiques, etc.), les groupes de Galois ainsi obtenus sont des sous-groupes du groupe symétrique S_n (s'ils agissent sur n racines). La théorie de Galois établit une correspondance biunivoque entre les sous-corps intermédiaires de E/K et les sous-groupes du groupe de Galois, dévoilant une profonde connexion entre la structure algébrique du corps et celle du groupe [37]. Historiquement, Galois a démontré que les groupes de Galois des polynômes généraux de degré n (pour $n\ge 5$) sont le groupe symétrique S_n ou alterné A_n, ce qui implique l'**insolubilité par radicaux** des équations générales de degré ≥ 5 (puisque A_n et S_n pour $n\ge5$ sont non résolubles). Ainsi, dès le XIX^{e} siècle, apparaissent comme groupes de Galois des **groupes finis classiques** tels que A_n. Une question majeure, l'**inverse du problème de Galois**, est de savoir si tout groupe fini est (ou non) réalisable comme groupe de Galois d'une certaine extension de $\mathbb{Q}$. De nombreux groupes ont été réalisés (via des constructions ingénieuses de polynômes), y compris certains groupes sporadiques dans des extensions sur $\mathbb{Q}$ ou des corps de fonctions. Cependant la question dans toute sa généralité reste ouverte, bien qu'en pratique on dispose d'une grande variété de groupes de Galois connus.

On voit donc se tisser un lien entre la théorie de Galois et les **groupes finis classiques** étudiés en algèbre. Par exemple, le groupe de Galois d'une extension cyclotomique $\mathbb{Q}(e^{2i\pi/m})$ (où l'on adjoint une racine m-ième de l'unité) est isomorphe à $(\mathbb{Z}/m\mathbb{Z})^{\times}$, un groupe abélien fini. Le groupe de Galois de $\mathbb{Q}(\sqrt[d]{a})$ est isomorphe à un sous-groupe du groupe de permutations des d racines, etc. Les **corps finis** apportent un éclairage additionnel via la notion de **groupe de Frobenius**. Étant donné le corps $\mathbb{F}*{p^n}$, son groupe multiplicatif $\mathbb{F}*}^\times$ est un groupe cyclique d'ordre p^n-1 [36], isomorphe par exemple à $\mathbb{Z}/(p^n-1)\mathbb{Z}$. Ce groupe apparaît comme un cas particulier d'un groupe de Galois : celui de $\mathbb{F*{p^n}$ sur lui-même est trivial, mais si l'on considère une extension de degré n d'un corps fini $\mathbb{F}$* (avec q éléments), alors le groupe de Galois est cyclique d'ordre divisant n. En somme, la théorie de Galois jette un pont entre l'algèbre des **extensions de corps** et la **théorie des groupes finis**, permettant d'étudier des problèmes d'équations polynomiales à travers les propriétés de groupes bien connus (diédraux, alternés, linéaires, etc.) – et réciproquement, de construire des exemples concrets de ces groupes comme groupes d'automorphismes de structures algébriques.

Partie V : Classification des groupes finis simples et dynamiques des orbites linéaires

L'un des plus grands accomplissements de l'algèbre au XXe siècle est la **classification des groupes finis simples**. Un groupe *simple* est un groupe non trivial qui n'a aucun sous-groupe normal autre que l'identité et lui-même – il constitue en quelque sorte un « atome » de la théorie des groupes (puisque tout groupe fini peut être décomposé en une suite de sous-groupes normaux, ses quotients successifs étant simples). La classification, achevée vers 1980 après l'effort combiné de dizaines de mathématiciens et des milliers de pages, affirme que tout groupe fini simple appartient à l'une des familles suivantes : **(1)** les groupes cycliques d'ordre premier (cas abélien trivial) ; **(2)** les **groupes alternés** A_n pour $n\ge 5$ (groupes des permutations paires sur n objets) ; **(3)** les **groupes de type de Lie finis**, qui comprennent les groupes classiques (groupes linéaires, orthogonaux, symplectiques, unitaires définis sur des corps finis) et certains groupes exceptionnels dérivés de la théorie des algèbres de Lie ; **(4)** les **26 groupes sporadiques**, qui ne rentrent dans aucune famille infinie [38]. Dans la famille (3), on retrouve notamment des groupes comme $PSL_{n}(q)$ (groupe projectif spécial linéaire sur $\mathbb{F}*q$), $PSp*(q)$ (groupe projectif symplectique), $PGL_n(q)$, et toute une panoplie d'autres groupes définis à partir de groupes de Lie sur les corps finis (y compris le mystérieux groupe de Tits de type $^2F_4$). Les **groupes sporadiques** incluent par exemple les groupes de Mathieu, de Janko, de Conway, le groupe de Fischer, et le plus grand d'entre eux : le **groupe Monstre**, de taille astronomique 8×10^{53} [39]. Ce dernier agit naturellement sur un espace de dimension 196883, ce nombre correspondant à la plus petite représentation fidèle du Monstre (hors la triviale) [40]. Fait intrigant, $196883 + 1 = 196884$ apparaît comme coefficient dans le développement de la fonction modulaire $j(q)$, ce qui a mené à la théorie du *Moonshine* montrant une connexion profonde entre le Monstre et les fonctions modulaires [41]. Ces anecdotes illustrent la richesse des structures en présence. En résumé, la classification nous dit que tout groupe fini simple est soit cyclique (très petit), soit alterné, soit lié à une famille de matrices sur corps finis, soit l'un des 26 « ovnis » sporadiques [42] [39].

Parmi ces familles, les **groupes linéaires sur les corps finis** occupent une place centrale, et il est éclairant de considérer leurs actions sur des ensembles de base combinatoire ou géométrique. Prenons par exemple $SL_n(\mathbb{F}_p)$, le groupe des matrices $n\times n$ à coefficients dans $\mathbb{F}_p$ et de déterminant 1. Ce groupe (pour $n\ge 2$) est simple dans la plupart des cas (en fait $PSL_n(\mathbb{F}_p) = SL_n(\mathbb{F}_p)/{\pm I}$ l'est pour $n\ge 2$, $(n,p)\neq (2,2),(2,3)$). Il agit naturellement sur l'espace vectoriel $V = \mathbb{F}_p^n$ par multiplication linéaire. Considérons l'**action** de $SL_n(\mathbb{F}_p)$ sur l'ensemble des vecteurs non nuls de V (c'est-à-dire $V \setminus

{0}$). Deux vecteurs v, w de V sont dans la même orbite sous SL_n si et seulement si l'on peut passer de v à w par une transformation linéaire de determinant 1. Or, pour p donné et n fixé, *toutes* les directions dans V sont équivalentes sous l'action de SL_n : plus formellement, $SL_n(\mathbb{F}_p)$ agit transitivement sur $\mathbb{P}^{n-1}(\mathbb{F}_p)$, l'ensemble des directions (points de l'espace projectif). En effet, toute base de V peut être envoyée sur n'importe quelle autre base par un élément de $GL_n(\mathbb{F}_p)$, et moyennant un ajustement par un scalaire (de déterminant $=1$) on peut restreindre à SL_n. Ainsi, il n'y a qu'une seule orbite de SL_n sur les vecteurs non nuls ; géométriquement, cela reflète l'homogénéité de l'espace projectif fini. Si l'on raffine et qu'on considère l'action sur des **paires** de vecteurs, ou sur des sous-espaces de dimension k, on obtient des orbites classifiées par des invariants (par ex. l'angle entre deux vecteurs, ou la dimension de l'intersection de deux sous-espaces). L'étude des orbites de tels groupes permet de comprendre la structure des espaces quotients X/G et de dénombrer des objets à isomorphisme près. Un exemple classique est le comptage des sous-espaces de dimension k de $\mathbb{F}_q^n$: le nombre de tels sous-espaces est donné par le coefficient binomial gaussien, et correspond au nombre d'orbites de GL_n agissant sur les k-uplets de vecteurs linéairement indépendants. De même, SL_n agissant sur $V=(\mathbb{F}_2)^n$ (on peut identifier $(\mathbb{F}_2)^n$ à l'ensemble des sommets d'un hypercube ${0,1}^n$) aura une action transitive sur les sommets de l'hypercube ayant une **poids de Hamming** donné. En effet, toute combinaison linéaire invertible sur $\mathbb{F}_2$ conserve le nombre de 1 mod 2 (la parité du poids), mais pour SL_n qui est inclus dans GL_n, l'orbite d'un vecteur de poids w sera l'ensemble des vecteurs de poids w si w est impair, ou de poids w si w est pair (car $\det=1$ impose une contrainte de parité pour $p=2$). L'analyse des orbites peut ainsi se raffiner selon les invariants disponibles.

La **dynamique des actions de groupes finis sur des ensembles combinatoires** offre une perspective utile en géométrie et en algorithmique. En *géométrie computationnelle*, exploiter les symétries via les actions de groupe peut drastiquement réduire la complexité en évitant de traiter des configurations isomorphes répétées. Par exemple, si l'on veut étudier toutes les configurations possibles d'un certain problème sur n points dans le plan, le fait que le groupe des permutations S_n agit sur ces configurations signifie qu'on peut restreindre l'analyse à un représentant par orbite (via un **quotient par l'action du groupe**), divisant ainsi le travail par $|S_n|=n!$ en théorie. De même, l'action de $SL_n(\mathbb{F}_p)$ sur ${0,1}^n$ mentionnée ci-dessus peut servir à classer des objets binaires (codes, ensembles) à une transformation linéaire près, ce qui est pertinent en théorie des codes ou pour étudier des propriétés invariantes par transformations linéaires. En **géométrie des polytopes**, le groupe $GL_n(\mathbb{Z})$ (ou $GL_n(\mathbb{F}_p)$) agit sur les points entiers (ou modulo p) d'espaces vectoriels, et comprendre les orbites peut aider à résoudre des problèmes de pavages, de réseaux ou d'optimisation linéaire en exploitant la symétrie. Plus généralement, la classification des groupes finis simples, en identifiant les « briques fondamentales » de symétrie, fournit un catalogue d'outils pour la géométrie et l'informatique. Par exemple, les groupes de Lie finis (tels que $PSL_n(q)$, $SO_n(q)$, etc.) agissent sur des géométries finies (bâtisses, variétés projectives sur $\mathbb{F}_q$) et ces actions possèdent des propriétés de régularité et de transitivité remarquables qui peuvent être utilisées dans des algorithmes (construction de graphes fortement réguliers, générateurs pseudo-aléatoires basés sur des marches aléatoires sur ces groupes, etc.). La perspective offerte par l'action de $SL_n(\mathbb{F}_p)$ sur des espaces vectoriels finis est donc double : **théorique**, en reliant combinatoire et algèbre linéaire, et **algorithmique**, en exploitant la symétrie pour optimiser des calculs ou classifier des objets de manière canonique.

Partie VI : Compilation dynamique orientée corps de calcul (réel vs *p*-adique) et parallélisation

Enfin, considérons une perspective exploratoire au croisement de l'architecture logicielle et des mathématiques : la possibilité de **compiler dynamiquement des instructions GPU** en paramétrant le « type de calcul » par le choix d'un corps de base – par exemple les nombres réels (flottants) classiques ou les nombres *p*-adiques. L'idée serait de développer des compilateurs ou bibliothèques capables de générer du code de calcul optimisé pour différentes arithmétiques sous-jacentes. Dans le contexte des modèles d'inférence massifs (réseaux de neurones à trillions de paramètres, etc.), cela pourrait signifier qu'une même opération linéaire (par ex. une multiplication matrice-vecteur) puisse être réalisée soit en virgule flottante standard, soit en utilisant une arithmétique modulaire sur plusieurs corps en parallèle, selon le besoin.

Une motivation pour cette approche provient de la recherche en **calcul parallèle et arithmétique de haute précision**. Des travaux suggèrent qu'en remplaçant certaines opérations en virgule flottante par des opérations en arithmétique modulaire (par exemple, travailler modulo plusieurs nombres premiers en parallèle), on peut obtenir des gains de fiabilité et de vitesse sur des architectures parallèles [33] [34]. En effet, les unités de calcul modernes (GPU, FPGA) sont très performantes pour les opérations entières et logiques, souvent plus que nécessaire pour l'arithmétique en précision simple. Exploiter ces capacités pour effectuer des calculs *exactement* modulo p^k (ou modulo plusieurs p_i en parallèle, à la manière du *Residue Number System*) peut permettre de **désengorger** certaines étapes critiques du calcul en virgule flottante. Par exemple, l'accumulation de très nombreuses additions de nombres réels entraîne une perte d'information (erreurs d'arrondi, dépassement de capacité d'exposant). Si l'on pouvait en parallèle accumuler ces sommes dans un (ou plusieurs) moduli entiers assez larges, on obtiendrait la partie entière exacte (au moins modulo 2^{64}, par ex.) du résultat [33], détectant les dépassements ou fournissant un contrôle sur l'erreur. Cette approche utilise les *p*-adiques de façon indirecte : calculer modulo 2^{N} revient à travailler dans $\mathbb{Z}*{2}$ tronqué, ce qui est l'analogue 2-adique d'une précision N bits. Plus généralement, calculer des résultats en parallèle dans $ \mathbb{Q}$* permettrait de }$ (avec troncature) et dans $\mathbb{R**comparer les deux résultats** pour détecter des anomalies : une discordance entre le calcul réel (soumis à l'arrondi) et le calcul *p*-adique (exact modulo p^N) signalerait une instabilité numérique ou un dépassement. Un compilateur orienté corps pourrait ainsi, à la volée, choisir de réaliser certaines additions ou multiplications dans un corps fini (ou anneau p-adique tronqué) en plus du calcul en flottant, pour vérifier ou affiner la précision.

Concernant la **parallélisation brute** des calculs dans les modèles de langage massifs, paramétrer le type de calcul offre aussi la possibilité d'utiliser au mieux les ressources du GPU. Par exemple, un GPU possède des unités SIMD pouvant réaliser 1024 opérations entières 32-bit en parallèle très efficacement. Si l'on paramètre une multiplication de matrices pour qu'elle opère sur des entiers modulo 2^{32} au lieu de flottants 32 bits, on pourrait potentiellement tirer parti de ces unités pour effectuer des multiplications *sans arrondi* en masse, puis recombiner les résultats (via l'isomorphisme entre $\mathbb{Z}$ et le produit de ses quotients chinois, etc.). Bien entendu, cela nécessite ensuite de "relever" le résultat entier en un nombre réel exploitable, mais des algorithmes comme la **transformée de NTT (Number Theoretic Transform)** ou le **CRT (Théorème des Restes Chinois)** peuvent reconstruire des résultats exacts à partir de composantes modulaire. On peut imaginer qu'une multiplication haute précision soit effectuée en scindant les nombres en plusieurs résidus modulo différentes bases, calculés en parallèle, atteignant ainsi une précision effective bien supérieure à la double précision, ce qui peut être utile pour, disons, accumuler des gradients de manière exacte lors de l'entraînement d'un réseau (éviter l'explosion/destruction de gradients par annulation).

Au-delà de l'exactitude, la **compilation paramétrable par corps** ouvre la porte à des calculs dans des structures algébriques non traditionnelles pour l'IA. Par exemple, on pourrait vouloir exécuter une couche de réseau de neurones dans $\mathbb{F}_2$ (logique booléenne) ou $\mathbb{F}_3$ pour tester la robustesse d'un modèle vis-à-vis de la quantification extrême, ou effectuer un calcul dans $\mathbb{Q}_p$ pour exploiter la distance ultramétrique dans un module de clustering hiérarchique. Un compilateur flexible permettrait de générer rapidement le kernel GPU approprié exploitant l'arithmétique choisie. Cela serait une extension radicale des approches de **quantization** déjà utilisées (où l'on passe en 8-bit, 4-bit, etc., pour accélérer les réseaux) : on ne se contenterait plus de réduire la précision, mais on changerait de *monde arithmétique* tout en bénéficiant de la massive parallélisation du matériel. Naturellement, de nombreux défis techniques existent (coût de conversion entre corps, gestion des overflows modulaires, etc.), mais la littérature HPC récente indique un intérêt croissant pour ces idées. Comme le souligne de Dinechin *et al.* (2019), l'arithmétique non-archimédienne est « bien adaptée à la parallélisation » et offre une base théorique solide pour le calcul exact [33] [34]. Intégrer de telles approches dans les moteurs d'inférence pourrait, à terme, améliorer la fiabilité des calculs de réseaux géants (en détectant les instabilités numériques) et permettre de mieux exploiter chaque transistor du GPU en diversifiant les types de calcul en parallèle.

En conclusion de cette partie, la notion de compilation dynamique multi-corps s'inscrit dans la tendance à **l'arithmétique adaptable**. Tout comme on utilise déjà différentes précisions (FP32, FP16, bfloat16) pour optimiser les performances des modèles de langage sans trop sacrifier la précision, on peut imaginer un futur où l'on utilise $\mathbb{F}_p$ ou $\mathbb{Q}_p$ ponctuellement pour garantir une exactitude ou une propriété mathématique (comme préserver une intégrité modulaire) au sein des calculs d'inférence. Cela pourrait impacter fortement la parallélisation : par exemple, exécuter en parallèle une opération en virgule flottante et le même calcul en modulo pourrait fournir un mécanisme de **redondance et vérification** sans ralentir drastiquement le traitement, le tout géré au niveau du compilateur. Bien que spéculative, cette perspective mêlant théorie des nombres, théorie de Galois (via l'utilisation de différents corps) et architecture GPU, illustre le potentiel qu'il y a à sortir des sentiers battus du calcul numérique pour les modèles d'IA de prochaine génération.

Conclusion et pistes futures

De l'étude des motifs digitaux des nombres premiers à la parallélisation ultramétrique des calculs massifs, nous avons parcouru un spectre de thèmes unissant la théorie des nombres, l'algèbre et l'informatique. Il en ressort un fil directeur : **la structure algébrique sous-jacente** – bases de représentation, types de complétions (réelle vs *p*-adique), symétries de Galois ou de groupes finis – influence profondément la manière dont on peut calculer et raisonner efficacement. Les nombres premiers, malgré leur distribution en apparence arbitraire en base 2 ou 16, se trouvent liés à des propriétés automatiques et des invariants modulaires profonds, nous rappelant la richesse cachée derrière la simplicité de leur définition. Les nombres flottants, qui semblent loin de ces considérations, présentent en réalité un compromis ingénieux entre continuité analytique et discrétisation : comprendre leurs limites encourage le développement de nouvelles techniques d'approximation (polynômes trigonométriques, etc.) pour fiabiliser les calculs d'inférence. Les nombres *p*-adiques offrent une « contre-partie » aux nombres réels, avec une topologie inverse, et commencent à inspirer tant des algorithmes exacts parallèles en calcul scientifique que des constructions théoriques en apprentissage automatique (réseaux hiérarchiques infinis). La théorie de Galois et la classification des groupes finis, apanage de l'algèbre la plus abstraite, trouvent des échos en informatique via l'étude d'actions de groupes sur des structures de données et l'exploitation des symétries pour réduire la complexité ou améliorer la compréhension (comme en combinatoire des réseaux ou en cryptographie basée sur les corps finis). Enfin, l'idée d'une compilation paramétrable par choix du corps de calcul jette un pont inédit entre le logiciel de bas niveau et la théorie des nombres, suggérant que des gains substantiels de

puissance de calcul pourraient être atteints en incorporant directement des structures algébriques alternatives dans les pipelines de calcul.

Parmi les **pistes futures**, on peut envisager :

- **Détection de motifs dans les suites de nombres premiers par IA** : utiliser des réseaux de neurones ou des méthodes d'apprentissage pour distinguer nombres premiers et composés à partir de leur représentation, afin de tester expérimentalement la "non-régularité" du langage des nombres premiers sur de grandes tailles (en complément des résultats théoriques connus).

- **Arithmétique mixte adaptative** : poursuivre le développement d'algorithmes capables de passer de façon transparente des flottants aux entiers modulaires ou *p*-adiques en cours de calcul, selon les besoins de précision. Par exemple, intégrer dans les librairies de tenseurs pour réseaux de neurones des types *entiers mod 2^n* permettant des accumulations exactes des gradients ou des quantifications sans perte.

- **Applications des distances ultramétriques** : en apprentissage non-supervisé ou en analyse de données, explorer l'utilisation de distances *p*-adiques pour des données à structure arborescente (par ex. phylogénie, taxonomies) : un réseau de neurones dont l'espace latent serait $\mathbb{Q}_p$ pourrait naturellement refléter des relations hiérarchiques (proximité *p*-adique signifiant partage d'« ancêtres » communs dans une arborescence de clusters).

- **Groupes de symétrie et architecture de réseaux** : s'inspirer de la classification des groupes pour construire des *réseaux invariants* ou *équivariants* par rapport à l'action de certains groupes finis. Des succès récents ont été obtenus en IA géométrique (par exemple des réseaux invariants par rotations 3D continues). On pourrait imaginer des réseaux de convolution définis sur des objets finis symétriques (grille torique, réseau projectif fini, etc.) exploitant l'invariance par un groupe de Lie fini ou un groupe sporadique (ceci restant largement exploratoire).

En conclusion, l'examen conjoint de ces domaines révèle un foisonnement d'idées pour améliorer notre compréhension et nos capacités de calcul. La collaboration entre mathématiques pures et informatique – par exemple, entre un théoricien des nombres *p*-adiques et un concepteur d'accélérateurs de calcul – peut sembler inattendue, mais c'est précisément de telles synergies que naissent les innovations majeures. La prochaine génération de modèles de calcul massifs pourrait bien tirer parti de ces notions avancées pour repousser les limites actuelles, en fiabilité comme en performance, tout en ouvrant de nouvelles voies de recherche à l'interface de plusieurs disciplines.

Annexe : Notations principales

- $\mathbf{N}, \mathbf{Z}, \mathbf{Q}, \mathbf{R}, \mathbf{C}$: ensembles des entiers naturels, entiers relatifs, rationnels, réels, complexes.

- $\mathbb{F}_p$ *(ou $\mathbf{Z}/p\mathbf{Z}$) :* ***corps fini*** *à p éléments (où p est un nombre premier). Plus généralement $\mathbb{F}_$* désigne le corps fini à p^n éléments.

- $\mathbb{Z}_p$: **entiers *p*-adiques**, anneau des suites cohérentes de résidus modulo p^n. Un élément de $\mathbb{Z}_p$ peut s'écrire de façon unique sous forme $a_0 + a_1 p + a_2 p^2 + \cdots$ avec $a_i \in \{0,\ldots,p-1\}$ (développement en base p infini vers la gauche).

- $\mathbb{Q}_p$: **nombres *p*-adiques**, corps des fractions de $\mathbb{Z}_p$. Chaque $x\in\mathbb{Q}_p$ s'écrit $x = p^v \cdot u$ avec $v\in \mathbf{Z}$ (valuation) et $u\in \mathbb{Z}_p^\times$ (unité *p*-adique, de valuation 0). On note $|x|_p = p^{-v}$ la norme *p*-adique associée.

- $v_p(x)$: **valuation *p*-adique** d'un rationnel (exposant de p dans la décomposition de x). Par convention $v_p(0)=+\infty$.

- $SL_n(F)$: groupe spécial linéaire de degré n sur un corps F – ensemble des matrices $n\times n$ à coefficients dans F et de déterminant 1. Par ex. $SL_n(\mathbb{F}_p)$ est un groupe fini simple (pour $n\ge 2$, exceptions notées dans le texte).

- $GL_n(F)$: groupe général linéaire de degré n sur F, composé des matrices inversibles de taille n (déterminant non nul).

- A_n, S_n : A_n est le **groupe alterné** sur n éléments (permutations paires, ordre $n!/2$), S_n le **groupe symétrique** sur n éléments (toutes les permutations, ordre $n!$).

- *Groupe de Galois* $\mathrm{Gal}(E/K)$: groupe des automorphismes d'une extension de corps E qui fixent chaque élément de K. Exemple : $\mathrm{Gal}(\mathbb{F}_{p^n}/\mathbb{F}_p) \cong \mathbf{Z}/n\mathbf{Z}$ engendré par $x\mapsto x^p$.

- *Corps de rupture* (de décomposition) : plus petit corps contenant K dans lequel un polynôme $f \in K[X]$ se factorise en produit de facteurs linéaires (contient donc toutes les racines de f).

- $\operatorname{ReLU}, \tanh$: fonctions d'activation usuelles en apprentissage, $\operatorname{ReLU}(x)=\max(0,x)$ (linéaire par morceaux, non dérivable en 0), $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (fonction sigmoïde bornée entre -1 et 1).

- *NTT* : **Number Theoretic Transform**, analogue de la FFT (Transformée de Fourier rapide) opérant dans un corps fini (souvent $\mathbb{F}_p$) en utilisant une racine primitive de l'unité d'ordre N mod p (utile pour la multiplication rapide d'entiers ou de polynômes).

- *CRT* : **Chinese Remainder Theorem** (Théorème des restes chinois), qui assure que pour des moduli entiers copremiers $n_1,\dots,n_r$, l'isomorphisme $\mathbf{Z}/(n_1\cdots n_r)\mathbf{Z} \;\cong\; \mathbf{Z}/n_1\mathbf{Z} \times \cdots \times \mathbf{Z}/n_r\mathbf{Z}$ permet de reconstruire un entier à partir de ses résidus modulo chacun des n_i.

- *Automate fini* : machine à états finis lisant un mot (suite de symboles) et acceptant ou non ce mot. Un langage est **régulier** s'il existe un automate fini le reconnaissant. Le résultat de Hartmanis-Shank dit que ${ representations(p) \mid p \text{ est premier}}$ n'est pas un langage régulier (et même nécessite exponentiellement d'états).

---

1 2 3 4 5 6 7 8 9 10 11 on_prime_syntaxical_base16.pdf
file://file-W8UVPWQpEvcWWKQ6Ey4WLq

12 [PDF] Stone-Weierstrass Theorem
https://web.math.utk.edu/~freire/teaching/m467f19/Stone-Weierstrass-Notes.pdf

13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 padic_overview.pdf
file://file-5RUubezYP8TjKgXeB35bQ6

32 [2107.07980] p-adic Cellular Neural Networks
https://arxiv.org/abs/2107.07980

33 34 Parallelism in Computer Arithmetic: A Historical Perspective (Invited Paper)
https://www.researchgate.net/publication/331347570_Parallelism_in_Computer_Arithmetic_A_Historical_Perspective_Invited_Paper

35 36 37 39 42 260720250243_ChatGPT.txt
file://file-HDDTj38EGKZompmzgYX5so

38 Liste des groupes finis simples - Définition et Explications
https://www.techno-science.net/glossaire-definition/Liste-des-groupes-finis-simples.html

40 41 196,883 and the Monster – Mark Ronan
http://www.markronan.com/mathematics/symmetry-corner/196-883/