Automorphismes intérieurs, involutions et points fixes, Denise Vella-Chemla pilotant gemini, juillet 2026

1. Introduction et contexte

Le document soumis, intitulé *"Qui utilise des automorphismes intérieurs sans en avoir conscienc"* par Denise Vella-Chemla (novembre 2024) https://denisevellachemla.eu/automorphismes-interieurs.pdf, revisite une note de 2018 sur les palindromes. L'approche explore l'utilisation des involutions et des points fixes pour caractériser des propriétés arithmétiques profondes. En s'inspirant explicitement de la célèbre preuve en une phrase de Don Zagier (1990) concernant la somme de deux carrés pour les nombres premiers de la forme $4k + 1$, l'autrice cherche à bâtir un cadre similaire pour aborder la conjecture de Goldbach.

L'objectif de ce document est de formaliser les définitions fonctionnelles de la note, d'analyser les résultats empiriques du programme Python fourni, et d'isoler les verrous théoriques clés afin de proposer des directions de recherche concrètes.

2. Formalisation mathématique de la structure fonctionnelle

2.1. Définition de l'ensemble de travail E_n

Pour un nombre pair n, on introduit l'ensemble E_n composé de couples ou de décompositions additives sous la forme :

$$E_n = \{a + b \mid a \text{ impair}, b \text{ impair}, a \in \mathcal{P}, 3 \leq a \leq \frac{n}{2}, 6 \leq a + b \leq n\}$$

où $\mathcal{P}$ désigne l'ensemble des nombres premiers. Cet ensemble rassemble les blocs fondamentaux des sommes d'impairs où le premier terme est premier et inférieur ou égal à la moitié de la borne supérieure.

2.2. L'involution de Goldbach f_n

On définit l'application $f_n : E_n \to E_n$ (qualifiée d'automorphisme intérieur dans la note) par l'action suivante sur les décompositions additives :

$$a + b_1 \mapsto a + b_2 \iff b_1 + b_2 = n$$

Sous forme de couple, cela correspond exactement à l'application programmée sous le nom `autom(n, d)` :

$$(a, b_1) \mapsto (a, n - b_1)$$

Cette application possède des propriétés structurelles remarquables :

- **Involution :** C'est une bijection qui est sa propre réciproque ($f_n \circ f_n = \mathrm{Id}$), car $n-(n-b_1) = b_1$.

- **Points fixes :** Un élément $a + b_1$ est un point fixe de f_n si et seulement si $b_1 = n - b_1$, ce qui impose $b_1 = \frac{n}{2}$.

3. Analyse des résultats du programme et observations arithmétiques

Les résultats numériques de l'annexe confirment une stricte dichotomie selon la structure arithmétique de n :

3.1. Classification selon la parité de $\frac{n}{2}$ (Formes $4k$ vs $4k+2$)

1. **Si** $n \equiv 0 \pmod 4$ **(forme** $4k$**) :** $\frac{n}{2} = 2k$ est un nombre pair. Or, par définition de E_n, le terme b_1 doit être un nombre impair. Par conséquent, il est géométriquement et arithmétiquement impossible d'avoir $b_1 = \frac{n}{2}$. La fonction possède donc rigoureusement **0 point fixe**.
2. **Si** $n \equiv 2 \pmod 4$ **(forme** $4k+2$**) :** $\frac{n}{2} = 2k + 1$ est un nombre impair. Si cet impair est compatible avec les contraintes de l'ensemble, des points fixes apparaissent.

3.2. Lien avec la primalité et croissance "à la Zagier"

L'observation la plus stimulante de la note indique que le nombre de points fixes augmente de façon marquante lorsque $\frac{n}{2}$ rencontre un nombre premier. En effet, un point fixe exige que $a + \frac{n}{2} \in E_n$. Pour que cette décomposition appartienne à E_n, il faut d'une part que $a \in \mathcal{P}$ et d'autre part que $\frac{n}{2}$ soit impair.

Si $\frac{n}{2}$ est lui-même un nombre premier, alors la décomposition $\frac{n}{2} + \frac{n}{2}$ devient un point fixe légitime (pour $a = \frac{n}{2}$), ce qui engendre un saut ou une progression dans le décompte combinatoire.

4. Verrous théoriques et pistes de prolongement

Pour transposer efficacement la stratégie combinatoire de Don Zagier à la conjecture de Goldbach, un certain nombre d'ajustements structurels sont à envisager :

4.1. Le principe de la preuve de Zagier

La force de la preuve de Zagier sur les formes $4k+1$ en somme de deux carrés réside dans l'existence d'une *seconde* involution sur le même ensemble fini, dont le comportement combinatoire force l'existence d'un point fixe partagé ou d'un nombre impair global d'éléments. Pour Goldbach, l'existence d'un point fixe pour f_n (quand $n = 4k+2$) ne fournit une solution que si $\frac{n}{2}$ est premier (cas trivial). Le véritable défi réside dans les cas où il n'y a pas de point fixe évident (comme les formes $4k$).

4.2. Proposition : construction d'une seconde involution croisée

Pour faire progresser cette approche, la piste cruciale consiste à définir une involution duale g_n : $E_n \to E_n$ qui n'agit pas sur le second membre de manière isolée, mais qui interconnecte la primalité

de a avec celle de b. Si l'on parvient à construire une involution g_n telle que :

1. Le nombre total d'éléments de E_n ou la parité des orbites soit invariant.
2. Les points fixes de g_n correspondent précisément aux paires de Goldbach (p, q) où p et q sont premiers.

Alors, un argument de parité topologique/combinatoire "à la Zagier" permettrait de garantir qu'un point fixe existe pour tout n, indépendamment du passage récursif de n à $n + 2$.

5. Première conclusion

La piste 2 de Denise Vella-Chemla introduit un changement de paradigme fécond en cherchant une validation topologico-combinatoire via les involutions. La classification des points fixes selon la structure $4k$ / $4k + 2$ est élégamment mise en lumière par le code Python. Pour franchir l'obstacle du passage récursif évoqué par l'autrice, la formalisation d'une seconde involution agissant sur les paires de diviseurs ou de restes s'impose comme la suite logique et prometteuse de ce travail.

6. Objectif topologique et combinatoire

En s'inspirant de la structure de preuve de Don Zagier, l'objectif est d'identifier une seconde involution $g_n : E_n \to E_n$ agissant en synergie avec votre première involution $f_n(a + b_1) = a + (n - b_1)$. Pour inverser ou bloquer la situation sur les configurations de type $4k$ (qui n'ont structurellement aucun point fixe par f_n), cette fonction g_n doit briser la rigidité du premier terme a en permutant les nombres premiers entre eux selon des règles déterministes.

7. Modélisation informatique de la recherche

Le script Python associé construit systématiquement l'ensemble de vos partitions :

$$E_n = \{a + b \mid a \in \mathcal{P}, a \leq n/2, a, b \text{ impairs}, 6 \leq a + b \leq n\}$$

La première stratégie testée dans le script est le **couplage par classe de restes** modulo 3. Comme les nombres premiers impairs (hors 3) sont confinés dans les classes $1 \pmod 3$ et $2 \pmod 3$, l'application tente d'établir un pont symétrique entre ces deux sous-ensembles :

$$(a, b) \mapsto (a \pm 2, b \mp 2)$$

Si les conditions aux bords forcent cette application à se refermer correctement sur E_n, l'analyse arithmétique de ses points fixes (les éléments isolés qui ne peuvent pas commuter) fournira la clé combinatoire manquante.

8. Structure du code fourni

Le fichier `recherche_involution.py` autonome généré localement comprend :

1. Un prédicat efficace de primalité (`is_prime`).

2. Un générateur de l'ensemble strict E_n défini dans votre note (`generate_E`).
3. Un banc d'essai évaluant la validité de l'involution candidate et dénombrant ses points fixes pour plusieurs ordres de grandeur de n.

Ces outils locaux vous permettront d'expérimenter et de modifier directement les critères géométriques et arithmétiques pour observer en temps réel la distribution des orbites.

Le programme d'expérimentation

```
# -*- coding: utf-8 -*-
\"\"\"
Script de recherche brute-force d'une seconde involution (g_n) sur l'ensemble E_n.
Inspir  de l'approche de Don Zagier pour la conjecture de Goldbach.
\"\"\"

def is_prime(n):
    if n < 2: return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0: return False
    return True

def generate_E(n):
    \"\"\"G n re l'ensemble E_n des couples (a, b)\"\"\"
    E = []
    # a impair, premier, 3 <= a <= n//2
    for a in range(3, n // 2 + 1, 2):
        if is_prime(a):
            # b impair tel que 6 <= a+b <= n
            for b in range(3, n - a + 1, 2):
                if a + b >= 6:
                    E.append((a, b))
    return E

def test_involution_piste1(E, n):
    \"\"\"
    Piste 1 : Involution par inversion des restes modulo 3
    On cherche    coupler un premier a = 3k+1 avec un premier a' = 3k+2 (ou inversement)
    en ajustant b pour pr server des propri t s structurelles.
    \"\"\"
    g = {}
    elements = set(E)

    # Trouver les nombres premiers disponibles dans l'intervalle
    primes_in_E = sorted(list(set(a for a, b in E)))

    for (a, b) in E:
        # Exemple de transformation sur le reste de a mod 3
        # On d cale de +/- 2 pour changer de classe de reste tout en restant impair
        if a % 3 == 1 and (a + 2) in primes_in_E and (a + 2, b - 2) in elements:
            g[(a, b)] = (a + 2, b - 2)
        elif a % 3 == 2 and (a - 2) in primes_in_E and (a - 2, b + 2) in elements:
            g[(a, b)] = (a - 2, b + 2)
        else:
            # Cas par d faut : point fixe de cette tentative
            g[(a, b)] = (a, b)
```

```
    # V rification si c'est une involution parfaite sur E
    is_inv = True
    for pair in E:
        if g.get(g[pair]) != pair:
            is_inv = False
            break

    fixed_points = [k for k, v in g.items() if k == v]
    return is_inv, fixed_points

def main():
    print("--- Recherche Brute-Force d'Involutions Duales ---")
    for n in [12, 16, 20, 22, 24, 28, 30]:
        E = generate_E(n)
        is_inv, f_pts = test_involution_piste1(E, n)
        print(f"n = {n:2d} | Taille E_n = {len(E):2d} | Involution modulo 3 valide : {is_inv}

if __name__ == "__main__":
    main()
```

Son résultat d'exécution

```
--- Recherche Brute-Force d'Involutions Duales ---
n = 12 | Taille E_n =  7 | Involution modulo 3 valide : False | Pts fixes : 4
n = 16 | Taille E_n = 15 | Involution modulo 3 valide : False | Pts fixes : 10
n = 20 | Taille E_n = 21 | Involution modulo 3 valide : False | Pts fixes : 14
n = 22 | Taille E_n = 29 | Involution modulo 3 valide : False | Pts fixes : 21
n = 24 | Taille E_n = 33 | Involution modulo 3 valide : False | Pts fixes : 24
n = 28 | Taille E_n = 48 | Involution modulo 3 valide : False | Pts fixes : 37
n = 30 | Taille E_n = 53 | Involution modulo 3 valide : False | Pts fixes : 41
```