Matrices de divisibilité et conjecture de Goldbach : vérification du critère de diagonale vide, et introduction aux valeurs singulières
Denise Vella-Chemla pilotant l'ia claude, juillet 2026

J'ai fourni à l'ia claude ces notes écrites en février 2025 [1] https://denisevellachemla.eu/CG-matinv.pdf , [2] https://denisevellachemla.eu/DG-matsymgrandit.pdf et [3] https://denisevellachemla.eu/Conj-Goldbach-flip-mat-booleennes.pdf [1] : claude m'avait dit que toute modélisation (et notamment celle que j'ai choisie d'utiliser dans le cas présent) qui codait de façon trop triviale la divisibilité (c'est le cas ici, les caractères de divisibilité étant codés sur les diagonales descendantes des matrices), était vouée à échouer à cause de l'obstruction de parité. Claude m'a alors conseillé d'étudier les valeurs singulières plutôt que les valeurs propres, c'est ce qui devrait faire l'objet de la présente note.

1. Contexte

Lors d'une session précédente, l'argument suivant avait été établi et démontré : la divisibilité ($d \mid n \Rightarrow d \leq n$) impose à toute matrice construite sur ce principe une structure triangulaire, et les valeurs propres d'une matrice triangulaire sont, par un résultat d'algèbre linéaire élémentaire, exactement égales à sa diagonale. Les trois matrices `trianglinf`, `miroir3` et `dg` de Denise, construites à partir d'un critère de divisibilité, sont concernées : leurs valeurs propres sont donc structurellement aveugles à tout ce qui se passe hors de la diagonale, c'est-à-dire à l'information combinatoire intéressante.

Les *valeurs singulières* échappent à cet argument, et constituent donc une piste non explorée. Ce document fait d'abord le point sur la vérification des matrices elles-mêmes (nécessaire avant tout calcul), puis introduit les valeurs singulières de façon autonome.

2. Vérification des matrices

Cette section est justifiée par le fait que je n'avais pas "sous la main" le code complet à fournir à claude, et il l'a donc recréé, peut-être en fouillant la page https://denisevellachemla.eu/notesnp.html de mon site ou bien en étudiant précisément, comme indiqué, les images fournies.

2.1. Construction

Le programme `autrecode.py` de Denise construit, pour un entier pair n, trois matrices de taille $(n-1) \times (n-1)$:

- `trianglinf` : matrice triangulaire inférieure encodant une structure de divisibilité ;
- `miroir3` : la même information, repliée par transposition puis double symétrie (`fliplr`, `flipud`) ;
- `dg = max(trianglinf, miroir3)` : le "rassemblement" des deux, qui réalise le repli $x \leftrightarrow n - x$ à l'intérieur du même triangle.

1. On peut ajouter ces deux notes de mai et juin 2025 : https://denisevellachemla.eu/matricealendroit.pdf et https://denisevellachemla.eu/sommesmatrices.pdf .

2.2. Calibrage à partir des images

Denise a fourni trois images pour $n = 40$ (`fig40-1/2/3.png`), la troisième portant des traits rouges tracés à la main sur les décompositions de Goldbach $(3, 37)$, $(11, 29)$ et $(17, 23)$. Une extraction automatique des pixels de ces images (identification des couleurs de remplissage, échantillon au centre de chaque cellule d'une grille 39×39) a permis de :

1. confirmer que la matrice `trianglinf` régénérée à partir du code correspond **exactement**, pixel pour pixel, à l'image fournie ;
2. confirmer de même pour `miroir3` ;
3. localiser précisément, grâce aux pixels touchés par les traits rouges, la famille de diagonales que Denise appelle "diagonale ascendante".

Ce calibrage a révélé que la diagonale associée à un candidat p n'est pas la diagonale "\" (ligne − colonne constante) mais l'**anti-diagonale** (ligne + colonne constante) :

$$K(p) \;=\; 2p - 2,$$

avec sa réfléchie par le repli à $K'(p) = 2(n-2) - K(p)$. Par exemple pour $n = 40$: $p = 3 \Rightarrow K = 4$ (et $K' = 72$), $p = 11 \Rightarrow K = 20$ (et $K' = 56$), $p = 17 \Rightarrow K = 32$ (et $K' = 44$) - exactement les trois diagonales suivies par les traits rouges de Denise.

2.3. Le critère de vacuité

Une anti-diagonale porte structurellement **trois** pixels toujours actifs, quelle que soit la primalité de p ou $n - p$:

- les deux pixels d'extrémité (les deux coins de la diagonale) ;
- le pixel central, où ligne = colonne, qui traduit un divisieur trivial (d divise d) et n'a rien à voir avec Goldbach.

Notre première tentative de vérification ne retirait que les deux extrémités, ce qui produisait un désaccord systématique sur *tous* les couples de Goldbach testés. Une fois le point central également exclu, la définition de "diagonale vide" devient :

> **Une anti-diagonale $K(p)$ est vide si tous ses pixels, hors les deux extrémités et le point central, valent 0.**

2.4. Résultat de la vérification numérique

Avec cette définition corrigée, la propriété

$$\text{anti-diagonale } K(p) \text{ vide} \quad \Longleftrightarrow \quad p \text{ et } n - p \text{ premiers}$$

a été testée sur `dg` pour $n \in \{16, 18, 20, 30, 40, 50, 60, 80, 100, 150, 200, 300, 500\}$ et tout $p \geq 11$ (les p plus petits sont écartés : leur anti-diagonale est trop courte pour contenir un intérieur non trivial une fois les trois points structurels retirés, ce qui produit des "vides" vacueusement vrais sans rapport avec Goldbach).

Résultat : une seule exception a été trouvée sur l'ensemble des cas testés, à $n = 300$, $p = 11$ ($n - p = 289 = 17^2$, non premier), où la diagonale est prédite vide alors qu'elle ne devrait pas l'être. Ce point isolé mérite d'être noté honnêtement plutôt que caché : soit il s'agit d'un artefact de bord comparable à ceux des petits p, soit il indique que le critère doit encore être affiné. Il n'invalide pas la tendance très nette observée par ailleurs, mais empêche de parler de correspondance *prouvée*.

Le code correspondant est fourni en annexe [2].

3. Introduction aux valeurs singulières

Cette section est écrite pour quelqu'un qui n'a jamais manipulé les valeurs singulières : l'objectif est de donner l'intuition avant tout calcul.

3.1. Une matrice comme transformation géométrique

Une matrice carrée A peut être vue comme une transformation qui déplace les points de l'espace : elle prend un vecteur v et produit Av. Si l'on part d'un cercle (en dimension 2) ou d'une sphère (en dimension n) de vecteurs de longueur 1, l'image de cette sphère par A est en général une *ellipse* (ou un ellipsoïde) : la transformation étire l'espace plus fort dans certaines directions que dans d'autres.

3.2. Valeurs propres : une notion adaptée seulement à certaines matrices

Les valeurs propres λ de A répondent à la question : existe-t-il des directions v que A ne fait que *dilater*, sans les faire tourner ($Av = \lambda v$) ? Pour une matrice triangulaire, ces directions privilégiées sont liées à la diagonale, ce qui explique pourquoi les valeurs propres de `trianglinf`, `miroir3` et `dg` sont triviales (égales à 1 partout où la diagonale vaut 1) : elles ne voient pas la disposition des autres pixels.

3.3. Valeurs singulières : la déformation de l'ellipsoïde

Les valeurs singulières répondent à une question différente, qui a toujours un sens, pour *n'importe quelle* matrice (même non carrée, même non triangulaire) : quelles sont les longueurs des demi-axes de l'ellipsoïde image de la sphère unité par A ? Ce sont des nombres positifs $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$, obtenus par la *décomposition en valeurs singulières* (SVD) :

$$A = U \, \Sigma \, V^T,$$

où U et V sont des matrices orthogonales (rotations/réflexions pures, sans déformation) et Σ est diagonale, avec les σ_i sur la diagonale. Concrètement :

2. J'ai écrit sur la page de l'onglet Notes du site le 1er janvier 2026 cette petite note : "Je viens d'apprendre quelque chose : j'avais utilisé en février 2025 des matrices que je pensais symétriques pour représenter les décompositions de Goldbach. En fait, c'étaient des matrices qu'on dit "persymétriques", elles sont symétriques par rapport à "l'autre diagonale". Si je veux en faire des matrices auto-adjointes, il faut que j'inverse les lignes et les colonnes ; ça se fait facilement en python en prenant à la place de la matrice M la matrice `M[::-1,::-1]` (pour une matrice 2×2).

- V indique les directions (orthogonales entre elles) que A envoie sur les axes de l'ellipsoïde ;
- les σ_i indiquent de combien A étire l'espace le long de chacun de ces axes ;
- U indique l'orientation finale de ces axes après transformation.

Contrairement aux valeurs propres, les valeurs singulières **voient toute la matrice**, pas seulement sa diagonale : deux matrices triangulaires ayant la même diagonale mais des pixels hors-diagonale différents auront, en général, des valeurs singulières différentes. C'est précisément cette sensibilité à la disposition hors-diagonale qui rend la piste intéressante pour `dg` : le spectre singulier de `dg` pourrait, en principe, encoder quelque chose sur le nombre ou la disposition des décompositions de Goldbach de n - reste à vérifier si c'est réellement le cas, et ce sera l'objet de la prochaine étape.

3.4. Comment les calculer

Numériquement, on ne calcule jamais U, Σ, V à la main : on appelle une routine numérique (ici `numpy.linalg.svd`), qui renvoie directement les σ_i triés par ordre décroissant. C'est fait dans la fonction `singular_values` du fichier `goldbach_matrices.py` joint. À titre d'exemple, pour `dg` avec $n = 40$, les 10 plus grandes valeurs singulières sont approximativement :

$$7.34,\ 3.45,\ 2.85,\ 2.75,\ 2.69,\ 2.69,\ 2.38,\ 2.36,\ 2.30,\ 1.97,\ \ldots$$

à comparer aux "valeurs propres" (la diagonale), qui valent toutes 1 : une différence immédiate de richesse.

4. Prochaine étape

La question à explorer, une fois ce document validé par Denise, est de savoir si des quantités dérivées du spectre singulier de `dg` (la plus grande valeur σ_1, le rang numérique, la décroissance du spectre, etc.) sont corrélées au nombre de décompositions de Goldbach de n, ou à d'autres invariants arithmétiques de n (nombre de diviseurs, etc.), sur une plage de n suffisamment large pour écarter une coïncidence numérique. Cette vérification n'a pas encore été menée : ce document pose seulement les bases (matrices vérifiées, critère de diagonale vide corrigé, notion de valeur singulière introduite) nécessaires pour l'entamer proprement.

5. Annexe : Programme de calcul matriciel pour la conjecture de Goldbach

```
# -*- coding: utf-8 -*-
"""
goldbach_matrices.py
====================
Reprend la construction du programme de Denise (autrecode.py) pour les
matrices trianglinf / miroir3 / dg, et ajoute :
  1. build_matrices(n)      : reconstruit les trois matrices, inchang
                              par rapport    autrecode.py (juste factoris
                              en fonction et sans les trac s matplotlib).
  2. anti_diagonale(dg, p)  : extrait l'anti-diagonale (ligne+colonne=2p-2)
```

```
                                associ e au candidat p.
  3. est_vide(dg, p)          : teste si cette anti-diagonale est "vide" au
                                sens de Denise : vide = aucun pixel allum
                                EN DEHORS de trois points structurellement
                                toujours actifs :
                                  - les deux pixels d'extr mit  de la
                                    diagonale (les deux coins),
                                  - le pixel central (ligne == colonne),
                                    qui correspond   un divisieur trivial
                                    (d divise d) et est lui aussi toujours
                                    actif, ind pendamment de Goldbach.
                                Ce troisi me point manquait dans notre
                                premi re tentative de v rification et
                                produisait de faux "non-vide" sur TOUS
                                les couples de Goldbach.
  4. verifie_conjecture_diagonales(n)
                               : compare, pour tout p de 2 a n//2, le
                                statut "vide" de l'anti-diagonale au
                                statut "p et n-p sont premiers".
  5. singular_values(n)       : calcule les valeurs singuli res de dg (et
                                de trianglinf) via numpy.linalg.svd, en
                                vue de la prochaine piste (cf. le .tex
                                joint pour les explications).
Convergence constat e (voir goldbach_matrices.tex) : la r gle est exacte
pour tout p >= 11 environ ; les rares exceptions se concentrent sur les
tout petits p (2, 3, 5, 7), dont l'anti-diagonale est trop courte pour
contenir une information non triviale (effet de bord, pas un contre-
exemple   la conjecture ni   la r gle elle-m me).
"""

import matplotlib.pyplot as plt
import numpy as np
from sympy import isprime

def ecrimat(m, couleur):
    plt.imshow(m, cmap=couleur, aspect='equal', origin='upper')

# ------------------------------------------------------------------
# 1. Construction des matrices (identique a autrecode.py, factorisee)
# ------------------------------------------------------------------
def build_matrices(n):
    """Reconstruit trianglinf, miroir3 et dg pour un entier pair n.
    Retourne (trianglinf, miroir3, dg), trois matrices carrees de
    taille (n-1, n-1).
    """
    M = np.zeros((n, n), dtype=int)
    for somme in range(n, 0, -2):
        x = somme - 1
        ligne = (n - somme + 2) // 2
        while x > 0:
            y = somme - x
            M[x, y] = 1
            x = x - ligne
    sudest = np.zeros((n - 1, n - 1), dtype=int)
    for x in range(1, n):
```

```
        for y in range(1, n):
            sudest[x - 1, y - 1] = M[x, y]
    trianglinf = np.rot90(sudest, 1)
    miroir = trianglinf.T.copy()          # symetrie / diagonale ascendante
    miroir2 = np.fliplr(miroir)
    miroir3 = np.flipud(miroir2)
    dg = np.fmax(trianglinf, miroir3)     # repli x <-> n-x
    return trianglinf, miroir3, dg

# ------------------------------------------------------------------
# 2-3. Anti-diagonale associee a un candidat p, et test de vacuite
# ------------------------------------------------------------------
def anti_diagonale(mat, p):
    """Renvoie la liste des cellules (ligne, colonne) de l'anti-diagonale
    ligne + colonne = 2p - 2 dans `mat` (matrice de taille (n-1, n-1))."""
    size = mat.shape[0]
    K = 2 * p - 2
    lignes = range(max(0, K - size + 1), min(size - 1, K) + 1)
    return [(r, K - r) for r in lignes]

def est_vide(mat, p):
    """True si l'anti-diagonale de p est vide au sens de Denise :
    aucun pixel actif hors des deux extremites et du point central
    (ligne == colonne), qui sont structurellement toujours a 1."""
    cellules = anti_diagonale(mat, p)
    if len(cellules) <= 2:
        # diagonale trop courte pour contenir un interieur : on ne
        # peut rien en conclure (voir note sur les petits p)
        return None
    interieur = cellules[1:-1]   # on retire les 2 extremites
    interieur = [(r, c) for (r, c) in interieur if r != c] # on retire le centre
    if not interieur:
        return None
    return all(mat[r, c] == 0 for (r, c) in interieur)

# ------------------------------------------------------------------
# 4. Verification systematique face a Goldbach
# ------------------------------------------------------------------
def goldbach_pairs(n):
    return [(p, n - p) for p in range(2, n // 2 + 1)
            if isprime(p) and isprime(n - p)]

def verifie_conjecture_diagonales(n, matrice='dg', p_min=11, verbose=True):
    """Compare, pour p de p_min a n//2, le statut "vide" de l'anti-
    diagonale au statut reel "p et n-p premiers".
    p_min=11 par defaut car les tout petits p (diagonales trop courtes)
    produisent des faux positifs triviaux, cf. docstring du module.
    """
    trianglinf, miroir3, dg = build_matrices(n)
    ecrimat(dg,'Reds')
    plt.show()
    mat = {'trianglinf': trianglinf, 'miroir3': miroir3, 'dg': dg}[matrice]
    gp = set(p for p, q in goldbach_pairs(n))
    mismatches = []
    for p in range(p_min, n // 2 + 1):
```

```
        vide = est_vide(mat, p)
        if vide is None:
            continue
        reel = p in gp
        if vide != reel:
            mismatches.append((p, n - p, vide, reel))
    if verbose:
        print(f"n={n:5d}  p testes dans [{p_min}, {n//2}]  "
              f"mismatches={mismatches if mismatches else 'AUCUN'}")
    return mismatches

# ------------------------------------------------------------------
# 5. Valeurs singulieres (prochaine piste, cf. le .tex pour le contexte)
# ------------------------------------------------------------------
def singular_values(n, matrice='dg'):
    """Calcule les valeurs singulieres de la matrice choisie pour n."""
    trianglinf, miroir3, dg = build_matrices(n)
    mat = {'trianglinf': trianglinf, 'miroir3': miroir3, 'dg': dg}[matrice]
    # svd renvoie U, s, Vt ; s = valeurs singulieres triees decroissant
    s = np.linalg.svd(mat.astype(float), compute_uv=False)
    return s

# ------------------------------------------------------------------
# Programme principal : verification sur une plage de n, puis apercu SVD
# ------------------------------------------------------------------
if __name__ == '__main__':
    print("=== Verification du critere 'diagonale vide <=> Goldbach' ===\n")
    total = 0
    #for n in [16, 18, 20, 30, 40, 50, 60, 80, 100, 150, 200, 300, 500]:
    for n in range(6,104,2):
        mm = verifie_conjecture_diagonales(n, matrice='dg', p_min=11)
        total += len(mm)
    print(f"\nTotal d'exceptions (p >= 11) sur tous les n testes : {total}")
    print("\n=== Apercu des valeurs singulieres (n=40) ===")
    s = singular_values(40, matrice='dg')
    print("Valeurs singulieres de dg (10 plus grandes) :")
    print(np.round(s[:10], 4))
    print("Valeurs propres (diagonale, pour comparaison) :")
    _, _, dg40 = build_matrices(40)
    print(np.round(np.diagonal(dg40)[:10], 4))
```

Le résultat de l'exécution du programme

```
=== Verification du critere 'diagonale vide <=> Goldbach' ===

n=    6  p testes dans [11, 3]  mismatches=AUCUN
n=    8  p testes dans [11, 4]  mismatches=AUCUN
n=   10  p testes dans [11, 5]  mismatches=AUCUN
n=   12  p testes dans [11, 6]  mismatches=AUCUN
n=   14  p testes dans [11, 7]  mismatches=AUCUN
n=   16  p testes dans [11, 8]  mismatches=AUCUN
n=   18  p testes dans [11, 9]  mismatches=AUCUN
n=   20  p testes dans [11, 10]  mismatches=AUCUN
n=   22  p testes dans [11, 11]  mismatches=AUCUN
```

```
n=   24  p testes dans [11, 12]  mismatches=AUCUN
n=   26  p testes dans [11, 13]  mismatches=AUCUN
n=   28  p testes dans [11, 14]  mismatches=AUCUN
n=   30  p testes dans [11, 15]  mismatches=AUCUN
n=   32  p testes dans [11, 16]  mismatches=AUCUN
n=   34  p testes dans [11, 17]  mismatches=AUCUN
n=   36  p testes dans [11, 18]  mismatches=AUCUN
n=   38  p testes dans [11, 19]  mismatches=AUCUN
n=   40  p testes dans [11, 20]  mismatches=AUCUN
n=   42  p testes dans [11, 21]  mismatches=AUCUN
n=   44  p testes dans [11, 22]  mismatches=AUCUN
n=   46  p testes dans [11, 23]  mismatches=AUCUN
n=   48  p testes dans [11, 24]  mismatches=AUCUN
n=   50  p testes dans [11, 25]  mismatches=AUCUN
n=   52  p testes dans [11, 26]  mismatches=AUCUN
n=   54  p testes dans [11, 27]  mismatches=AUCUN
n=   56  p testes dans [11, 28]  mismatches=AUCUN
n=   58  p testes dans [11, 29]  mismatches=AUCUN
n=   60  p testes dans [11, 30]  mismatches=AUCUN
n=   62  p testes dans [11, 31]  mismatches=AUCUN
n=   64  p testes dans [11, 32]  mismatches=AUCUN
n=   66  p testes dans [11, 33]  mismatches=AUCUN
n=   68  p testes dans [11, 34]  mismatches=AUCUN
n=   70  p testes dans [11, 35]  mismatches=AUCUN
n=   72  p testes dans [11, 36]  mismatches=AUCUN
n=   74  p testes dans [11, 37]  mismatches=AUCUN
n=   76  p testes dans [11, 38]  mismatches=AUCUN
n=   78  p testes dans [11, 39]  mismatches=AUCUN
n=   80  p testes dans [11, 40]  mismatches=AUCUN
n=   82  p testes dans [11, 41]  mismatches=AUCUN
n=   84  p testes dans [11, 42]  mismatches=AUCUN
n=   86  p testes dans [11, 43]  mismatches=AUCUN
n=   88  p testes dans [11, 44]  mismatches=AUCUN
n=   90  p testes dans [11, 45]  mismatches=AUCUN
n=   92  p testes dans [11, 46]  mismatches=AUCUN
n=   94  p testes dans [11, 47]  mismatches=AUCUN
n=   96  p testes dans [11, 48]  mismatches=AUCUN
n=   98  p testes dans [11, 49]  mismatches=AUCUN
n=  100  p testes dans [11, 50]  mismatches=AUCUN
n=  102  p testes dans [11, 51]  mismatches=AUCUN

Total d'exceptions (p >= 11) sur tous les n testes : 0

=== Apercu des valeurs singulieres (n=40) ===
Valeurs singulieres de dg (10 plus grandes) :
[7.34   3.4547 2.8512 2.7484 2.6926 2.6889 2.3777 2.3589 2.2951 1.9734]
Valeurs propres (diagonale, pour comparaison) :
[1 1 1 1 1 1 1 1 1 1]
```