Les surprises des matrices, Denise Vella-Chemla, 15 juin 2025.

1. Introduction

Le contexte est la conjecture de Goldbach : on cherche à décomposer n un nombre pair $\geqslant 4$ en somme de 2 nombres premiers.

Cette note est destinée à rendre compte d'une découverte concernant des matrices qui se sont avérées surprenantes : on pensait que ces matrices fournissaient les décomposants de Goldbach d'un nombre pair n en étudiant leurs diagonales et il s'avère qu'elles les fournissent en étudiant plutôt leurs lignes. On fournira le programme qui a permis de découvrir cela.

2. Matrice initiale associée à un nombre pair n

La matrice initiale M est une matrice triangulaire inférieure de taille $(n-1) \times (n-1)$. Elle est définie par :

$$M[x, n-2-x] = 1 \ \ \forall x, \quad 0 \leq x \leq n-3$$

pour les booléens à vrai sur la diagonale descendante principale, et par

$$M[kp, n-2p-kp] = 1 \ \ \forall p, p \text{ premier et } 2 \leq p < n/2.$$

M contient les caractères de divisibilité des nombres de 2 à $n-1$ dans ses diagonales ascendantes, de la diagonale concernant 2 en bas à droite de la matrice, à la diagonale concernant $n-1$, en haut à gauche de la matrice.

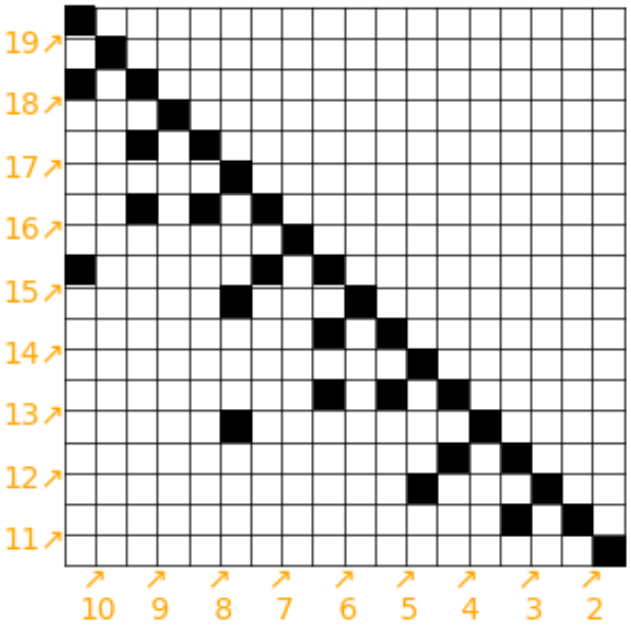

FIGURE 1 : Matrice M pour $n = 20$.

3. Symétrisation de la matrice initiale de n

La matrice M', symétrique de M par rapport à sa diagonale ascendante centrale, positionne dans la diagonale qui était associée à un nombre x dans M les caractères de divisibilité de son complémentaire. Là intervient une bizarrerie, du fait d'erreurs qu'on a commises dans les indices ; ce n'est pas le complémentaire de x à n, qui est positionné dans la diagonale de M' correspondant à la diagonale de x dans M, comme on souhaitait que ça soit le cas, mais c'est son complémentaire à $n+1$ qui est $n+1-x$. Ainsi, ci-dessous, dans la matrice M' associée à $n = 20$, les caractères de divisibilité de $9 = 21 - 12$ sont positionnés dans la même diagonale de M' que ceux de 12 dans la matrice M.

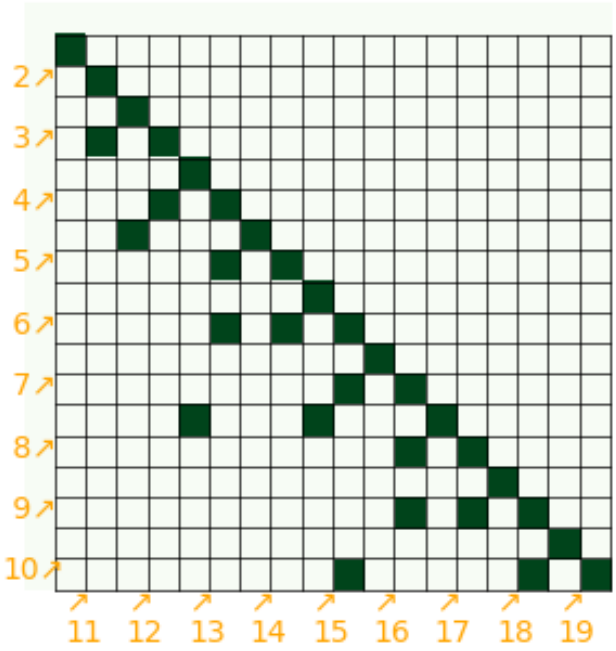

FIGURE 2 : Matrice M' pour $n = 20$.

4. Ou exclusif booléen entre la matrice M de n et sa matrice symétrique M'

On rappelle que le *ou exclusif booléen* est défini sur l'ensemble $\mathbb{B}$ des booléens par :

$$\begin{aligned} 0 \ \texttt{XOR} \ 1 &= 1 \\ 1 \ \texttt{XOR} \ 0 &= 1 \\ 0 \ \texttt{XOR} \ 0 &= 0 \\ 1 \ \texttt{XOR} \ 1 &= 0 \end{aligned}$$

La matrice M'' obtenue en effectuant un `XOR` (*ou exclusif booléen*) entre les matrices M et M' est caractérisée ainsi : un élément de M'' est le booléen `VRAI` si et seulement si les éléments correspondant (i.e. aux mêmes indices) dans M et dans M' sont différents ; deux booléens `VRAI` à la même

position (d'indices de ligne et colonne identiques) se détruisent l'un l'autre par l'opération XOR.

Voici la matrice $M'' = M$ XOR M' associée au nombre pair $n = 20$.

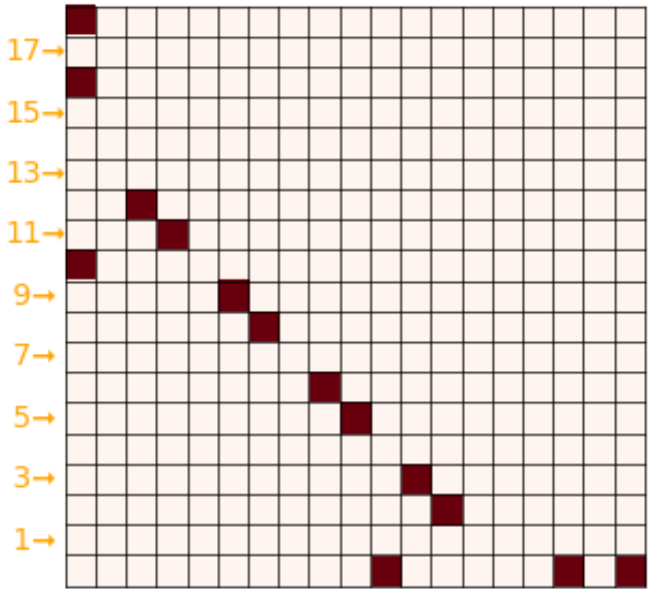

FIGURE 3 : Matrice XOR M'' pour $n = 20$.

On considère alors la matrice XOR comme étant la matrice d'adjacence d'un graphe et on cherche quels sont les sommets du graphe de degré nul, i.e. les sommets du graphe qui ne sont reliés à aucun autre sommet par une arête du graphe. Cela revient à effectuer des comptages non pas par diagonales comme on le faisait jusque-là, mais par lignes, en repérant les lignes vides. Il s'avère que les nombres correspondant aux lignes vides, lorsqu'on numérote les lignes comme on l'a fait dans la matrice XOR ci-dessus, s'ils ne sont pas composés, sont systématiquement des nombres premiers décomposants de Goldbach de n.

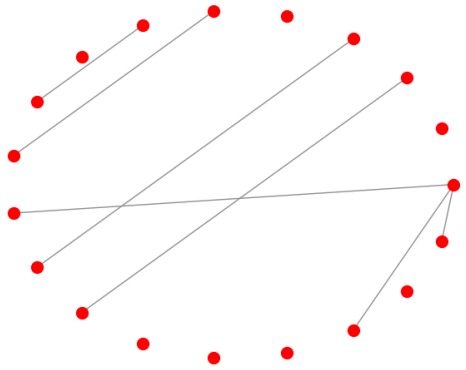

FIGURE 4 : Graphe de la matrice M'' considérée comme une matrice d'adjacence.

Une autre façon de considérer la recherche des lignes vides consiste à :

- placer les booléens égaux à `VRAI` de la matrice M aux points du plan complexe qui ont pour coordonnées leurs indices dans la matrice ;
- obtenir les points symétriques de ces points initiaux, qui ont les mêmes coordonnées qu'eux, mais interverties ;
- considérer la recherche des lignes vides comme une projection sur l'axe des ordonnées et comme le fait de quotienter $\mathbb{R}$ par cette projection (on quotiente pour ne garder que le complémentaire de la projection).

Quelle pourrait être la raison de ce phénomène, de retrouver les décomposants de Goldbach de n par les lignes vides ? On avait considéré précédemment les nombres non pas en soi, mais plutôt comme établissant, ou pas, des relations de divisibilité entre d'autres nombres. Par exemple, le nombre 9 est composé parce qu'il "contient" 3 et 6 qui le composent de manière additive, et qu'on a 3 divise 6 (on le note $3 \mid 6$). Ainsi, tout nombre composé peut se décomposer en une somme au moins de deux nombres, dont l'un divise l'autre. Les nombres premiers ne possèdent pas une telle propriété. Il y a un certain nombre de relations invariantes, que vérifient les éléments de la matrice M, sur chaque diagonale descendante associée à un nombre premier, qui est que tout booléen égal à `VRAI` est suivi d'un booléen égal à `FAUX` et que deux booléens égaux à `VRAI` consécutifs sont toujours écartés du même nombre de cases (p) verticalement et horizontalement l'un de l'autre, comme des tours sur un échiquier. Ces relations invariantes par diagonales doivent avoir pour conséquences des relations entre les éléments d'une même ligne, qu'il faut trouver.

La difficulté consiste maintenant à établir les propriétés que vérifient les indices des lignes, à partir des définitions de M et M' et du fait que $M'' = M$ `XOR` M', pour établir le fait que les nombres correspondant aux lignes vides, lorsque ce sont des nombres premiers, sont les décomposants de Goldbach de n d'une part, et d'autre part, pour établir le fait qu'il existe toujours une ligne vide associée à un nombre premier dans la matrice `XOR`.

Annexe : programme de calcul des différentes matrices et de leur affichage graphique

```
import numpy as np
from numpy import bitwise_xor,sqrt
import networkx as nx
import matplotlib.pyplot as plt

def premier(atester):
      k = 2
      if atester in [0, 1]:  return False
      if atester in [2, 3, 5, 7]:  return True
      while True:
            if k * k > atester:  return True
            else:
                  if atester % k == 0:  return False
                  else:  k = k + 1

for n in range(20,22,2):
      matrice1 = np.zeros((n-1,n-1),dtype='bool')
      matrice2 = np.zeros((n-1,n-1),dtype='bool')
      print('',n,':::::')
      for d in range(n//2):
            if d == 0 or premier(d+1):
                  for k in range(1,((n-2*d-2)//(d+1))+1):
                        if n-2*d-(1+k*(d+1)) >= 0:
                              matrice1[1+k*(d+1)-1,n-2*d-(1+k*(d+1))-1] = True
                              matrice2[n-2*d-(1+k*(d+1))-1,1+k*(d+1)-1] = True
      print('matrice 1')
      for x in range(len(matrice1)):
            for y in range(len(matrice1)):
                  if matrice1[x,y]:
                        print('(',x,',',y,'), ',end='')
                        print('0',end='')
            print('')
      print('matrice 2')
      plt.grid ( False )
      plt.axis ('off')
      for x in range(n-1):
            plt.plot([x-0.5,x-0.5],[0.5,n-1.5],color='black',linewidth=0.5)
      for y in range(n-1):
            plt.plot([-0.5,n-2.5],[y+0.5,y+0.5],color='black',linewidth=0.5)
      for x in range(1,n//2):
            plt.annotate(f'n-x:2d'+'',xy=(-2.5,n-2*x-1),color='orange')
            plt.annotate('',xy=(n-2*x-2,-0.2),color='orange')
            plt.annotate(f'x+1:2d',xy=(n-2*x-2,-1.2),color='orange')
      plt.imshow(matrice1, cmap='Grays', aspect='equal', origin='lower')
      plt.xlim(-3,n+3)
      plt.ylim(-3,n+3)
      plt.show()
      nomfic ='fig'+ str ( n ) +'-1'
      #plt.savefig ( nomfic ) ; plt.close ()
```

```
plt.grid ( False )
plt.axis ('off')
for x in range(n-1):
     plt.plot([x+0.5,x+0.5],[-0.5,n-2.5],color='black',linewidth=0.5)
for y in range(n-1):
     plt.plot([0.5,n-1.5],[y-0.5,y-0.5],color='black',linewidth=0.5)
for x in range(1,n//2):
     plt.annotate(f'x+1:2d'+'',xy=(-1.2,n-2*x-2.2),color='orange')
     plt.annotate('',xy=(n-2*x-1.2,-1.2),color='orange')
     plt.annotate(f'n-x:2d',xy=(n-2*x-1.2,-2.2),color='orange')
plt.imshow(matrice2, cmap='Greens', aspect='equal', origin='lower')
plt.xlim(-3,n+3)
plt.ylim(-3,n+3)
plt.show()
nomfic ='fig'+ str ( n ) +'-2'
plt.savefig ( nomfic ) ; plt.close ()
diffmat = bitwise_xor(matrice1,matrice2)
print('xor des deux matrices ')
plt.grid ( False )
plt.axis ('off')
for x in range(n):
     plt.plot([x-0.5,x-0.5],[-0.5,n-1.5],color='black',linewidth=0.5)
for y in range(n):
     plt.plot([-0.5,n-1.5],[y-0.5,y-0.5],color='black',linewidth=0.5)
for x in range(1,n-1,2):
     plt.annotate(f'x:2d'+'',xy=(-2.5,x-0.3),color='orange')
plt.imshow(diffmat, cmap='Reds', aspect='equal', origin='lower')
plt.xlim(-3,n+3)
plt.ylim(-3,n+3)
plt.show()
nomfic ='fig'+ str ( n ) +'-3'
plt.savefig ( nomfic ) ; plt.close ()
G = nx.from_numpy_array(diffmat)
nx.draw_circular(G,node_color='red',edge_color='909090',node_size=100)
for x in range(n-1):
     if G.degree[x] == 0 and x%2 != 0 and x > sqrt(n):
          print(x,' est de degre 0.',end='')
          if premier(x) and premier(n-x):
               print(' dg.')
          else:
               print('rate.')
```