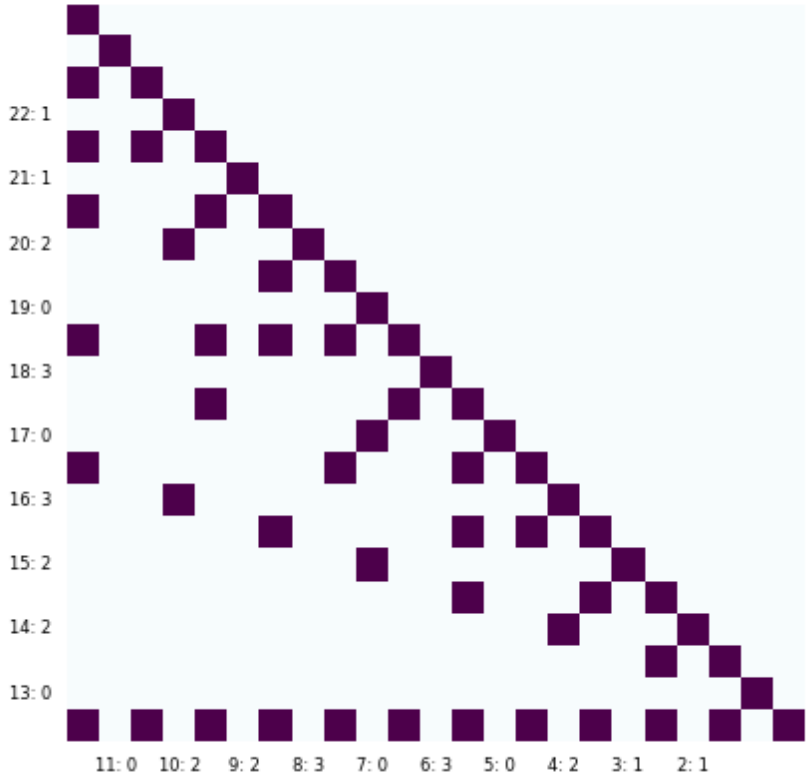
Conjecture de Goldbach, matrices booléennes, symétries
Denise Vella-Chemla, 5.2.25

On étudie ici la conjecture de Goldbach en utilisant des matrices de booléens, auxquelles on applique certaines transformations.

La matrice de base utilisée, appelons-la M_1 est une matrice de divisibilité, de la forme suivante :



On dit d'une telle matrice qu'elle est triangulaire inférieure. Elle est ici de taille 23, à la recherche des décomposants de Goldbach du nombre pair $n = 24$.

Les divisibilités se lisent sur les diagonales descendantes, en remontant du bas de la matrice vers le haut, et en lisant de la droite vers la gauche : la diagonale principale, dont tous les pixels sont colorés, "dit" que tout nombre est divisible par 1 ; sur la première diagonale "de divisibilité", c'est-à-dire la première diagonale contenant des pixels colorés et se trouvant sous la diagonale principale (i.e. la diagonale sous la diagonale qui est directement sous la diagonale principale), un pixel sur 2 est coloré ; puis un pixel sur 3, sur la quatrième diagonale sous la diagonale principale ; puis un pixel sur 4, etc., et ce sur chaque diagonale dite de divisibilité (*note* : pour se repérer en lisant M_1, on prendra les pixels colorés de la dernière ligne comme premiers pixels des diagonales de divisilité).

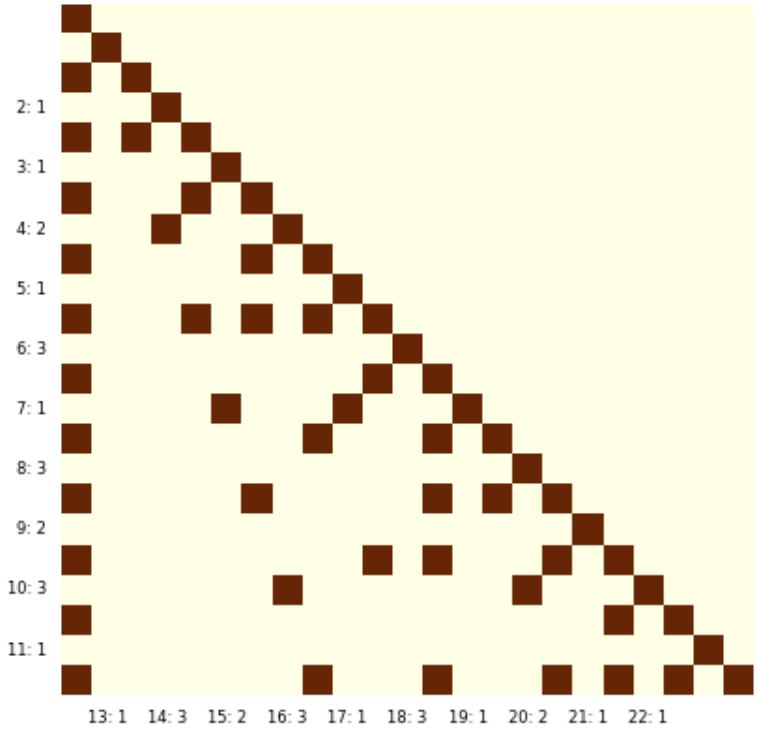
L'invariance qui caractérise une diagonale de divisibilité, comme celle de la divisibilité par p, est une invariance par translation : on peut la décrire de différentes manières :

- la diagonale contient un pixel coloré tous les p pixels ;
- la somme de p pixels successifs sur la diagonale vaut toujours 1 [1];
- tout pixel de la diagonale est égal au $p^{\text{ième}}$ pixel après (ou avant) lui.

Les nombres premiers supérieurs à $\sqrt{n}$, la racine de la taille de la matrice carrée qui vaut dans le cas présent $\sqrt{24} = 4, ...$, sont clairement apparents dans la matrice M_1 : ils correspondent aux diagonales ascendantes ne contenant pas de pixel coloré, hormis celui de la diagonale descendante principale.

On peut faire subir à cette matrice des symétries : symétrie horizontale, symétrie verticale, ou symétrie par rapport aux diagonales (principale ascendante, principale descendante, par exemple), etc [2].

Voici la matrice M_2 obtenue à partir de M_1 par une symétrie par rapport à la diagonale ascendante principale.



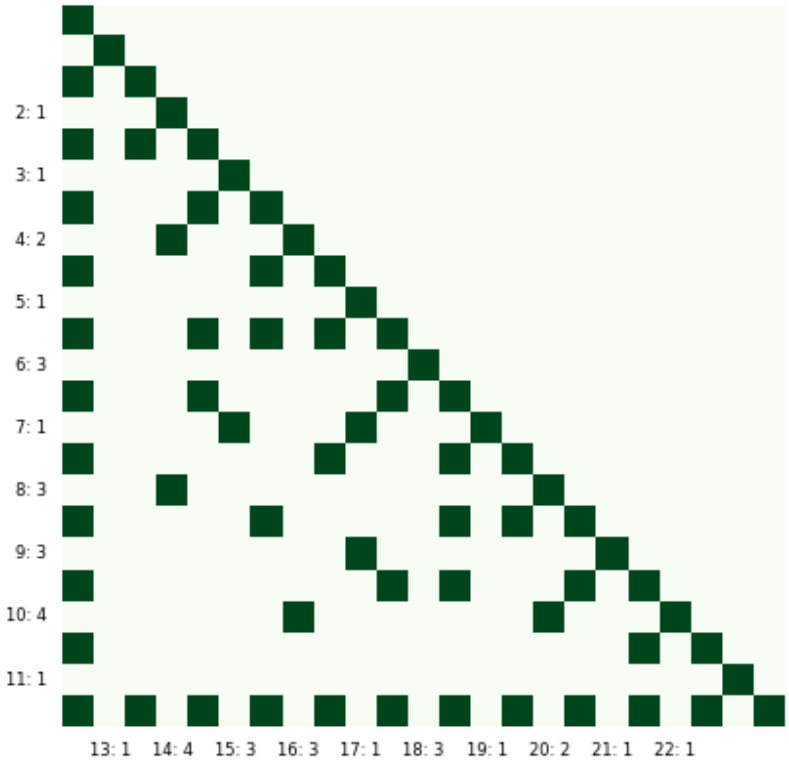
[1] Il faudrait réfléchir à la distinction entre de telles suites de pixels et les pavages de Penrose, caractérisés par des suites de booléens telles que tout 1 est toujours suivi d'un 0.

[2] Ces opérateurs appartiennent au groupe de Klein K_4, dit groupe du matelas.

Effectuons maintenant un *ou* logique entre les éléments des deux matrices M_1 et M_2, terme par terme ; l'opération logique *ou* est désignée en général par le symbole $\vee$; elle est calculable par la fonction maximum (notée $\max(x, y)$) si le chiffre 0 code le booléen *faux* et le chiffre 1 code le booléen *vrai* : $x \vee y = 1$ si $x = 1$ ou $y = 1$ ou s'ils sont égaux tous les 2 à 1 ; on parle ici du *ou* inclusif, ou disjonction booléenne. $x \vee y = 0$ sinon. On résume cela dans la table de vérité logique du $\vee$ ci-dessous :

$\vee$	0	1
0	0	1
1	1	1

Le résultat du calcul de $M_3 = \boxed{\max}(M_1, M_2)$, élément par élément[3] des deux matrices, est la matrice suivante :



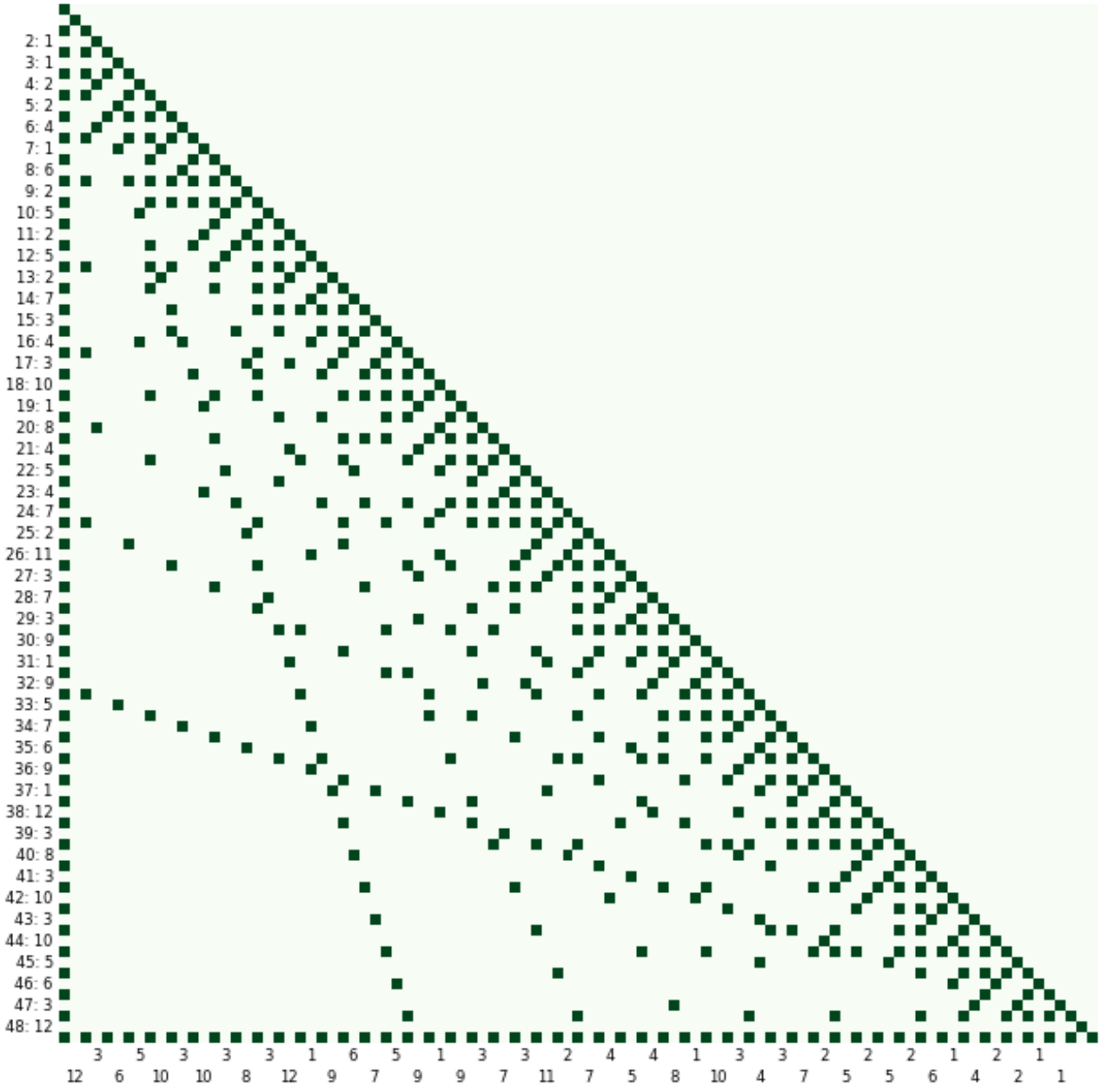
Voyons les nombres de booléens de chaque diagonale ascendante de la matrice résultante. Ces nombres indiquent très simplement les décomposants de Goldbach d'un nombre pair n qui sont supérieurs à $\sqrt{n}$: ce sont les nombres dont la diagonale correspondante contient un seul pixel coloré (hormis le pixel de la diagonale principale).

On note que la matrice M_3 étant symétrique par rapport à la diagonale principale ascendante, les nombres de pixels des diagonales successives (notés autour des grilles de pixels présentées) concaténés dans une liste constituent une liste dite "palindrome" (i.e. qui se lit identiquement dans

[3]Ce calcul élément par élément amène à l'encadrement du mot "max" pour représenter l'opération.

les deux sens [4]).

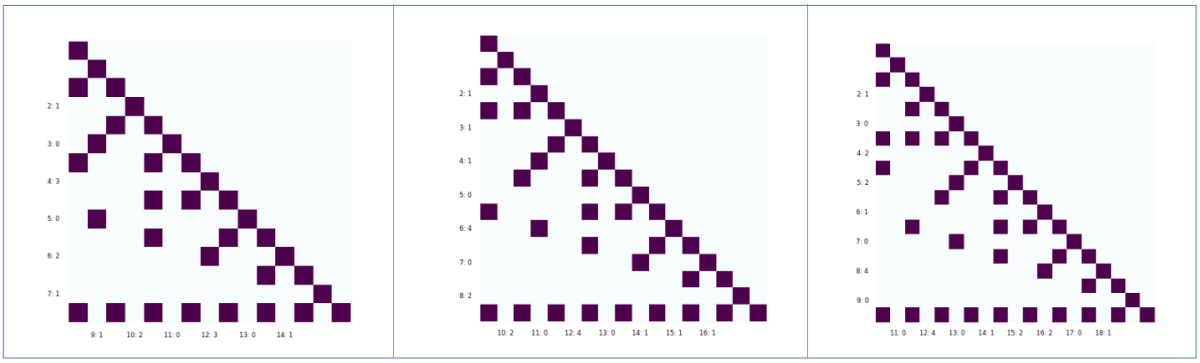
Ci-dessous la grille de pixels associée au nombre pair 98, sur laquelle on voit les décompositions de Goldbach $19+79, 31+67$ et $37+61$ (on n'a indiqué que les nombres de pixels, en bas du graphique, pour le rendre plus lisible).



Pour envisager une démonstration par récurrence, il faudrait bien étudier la manière dont la matrice

[4] comme la date à laquelle est écrite cette note, 5.2.25, de suite de chiffres 5225, avec l'année écrite en abrégé 25 pour 2025, alors que c'est en 1925, il y a un siècle, qu'a été écrit tout le formalisme matriciel de la mécanique quantique.

M_1 est modifiée par passage d'un nombre pair au suivant. À cette fin, on montre ci-dessous les grilles M_1 associées aux nombres pairs 16, 18 et 20.



On voit que la matrice doit être étendue dans le sens des diagonales decendantes principales, diagonale par diagonale en respectant l'invariance de divisibilité vue plus haut, et "vers le nord-ouest".

On a utilisé le programme ci-dessous [5] :

```
import matplotlib.pyplot as plt
import numpy as np

def ecrimat(m,couleur):
     plt.imshow(m, cmap=couleur, aspect='equal', origin='upper')
     taille, taille = m.shape
     for k in range(0,n-2,2):
          somme = 0
          x = 0
          y = k-x
          while y != 0:
               somme = somme+m[y,x]
               x = x+1
               y = y-1
          if somme>0:
               plt.text(-1,k+1,f'(k+2)//2:somme-1',fontsize=6,ha='right',va='center')
               if (k//2)%2 == 0:
                   plt.text(n-(k+1)-2,n-1,f'n-((k+2)//2):somme-1',fontsize=6,va='top',
                                    ha='center')
               else:
                   plt.text(n-(k+1)-2,n+1,f'n-((k+2)//2):somme-1',fontsize=6,va='top',
                                    ha='center')
     plt.grid(False)
     plt.axis('off')
     plt.show()
```

[5] Je remercie J. Chemla pour l'esthétique des résultats de ses programmes, qui facilite l'appréhension des problèmes.

```
def symparrapportdiagoasc(m):
     taille,taille = m.shape
     m2 = np.zeros((taille,taille),dtype='int')
     for x in range(taille):
          for y in range(taille):
               m2[x,y] = m[y,x]
     return(m2)

for n in range(16,22,2):
     M = np.zeros((n,n),dtype='int')
     sudest = np.zeros((n-1,n-1),dtype='int')
     trianglinf = np.zeros((n-1,n-1),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
     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)
     ecrimat(trianglinf,'BuPu')
     miroir = symparrapportdiagoasc(trianglinf)
     miroir2 = np.fliplr(miroir)
     miroir3 = np.flipud(miroir2)
     ecrimat(miroir3,'YlOrBr')
     dg = np.fmax(trianglinf,miroir3)
     ecrimat(dg,'Greens')
```