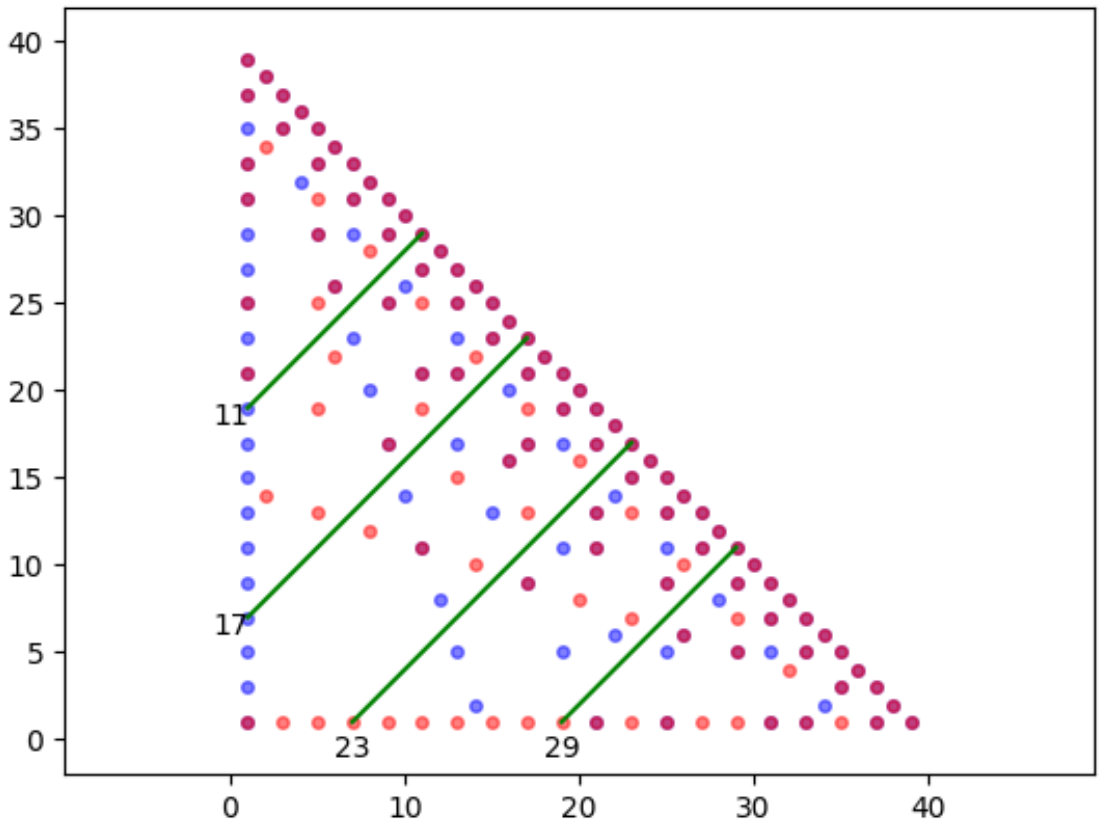
Les coordonnées de Goldbach ne commutent définitivement pas, Denise Vella-Chemla, juin 2025.

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 fournir le programme de représentation des décomposants de Goldbach dans le plan complexe.

La symétrie qu'on avait utilisée par rapport à la diagonale des matrices est ici un échange des coordonnées x et y.

Ci-dessous, sont représentés dans le plan complexe pour $n = 40$ les points représentant les caractères de divisibilité.



Fournissons ci-après le programme de dessin des points : on remarquera combien le programme est beaucoup plus facile à lire que les programmes précédents.

```
import matplotlib.pyplot as plt
import numpy as np
import math
from math import sqrt

for n in range(40,42,2):
     points = np.zeros((n-1)*(n-1),dtype='complex')
     nbde1 = 0
     for d in range(n//2):
          for l in range(((n-2*d-2)//(d+1))+1):
               z = 1+l*(d+1)+1j*(n-2*d-(1+l*(d+1)))
               if z.imag >= 0:
                    plt.scatter(z.real,z.imag,color='blue',alpha = 0.5, s=16)
                    points[nbde1] = z
                    nbde1 = nbde1+1
     indiceimage = 0
     pointsimage = np.zeros(nbde1,dtype='complex')
     for d in range(nbde1):
          point = points[d]
          zimage = point.imag+1j*point.real
          plt.scatter(zimage.real,zimage.imag,color='red',alpha = 0.5, s=16)
          pointsimage[indiceimage] = zimage.real+1j*zimage.imag
          indiceimage = indiceimage+1
     plt.axis('equal')
     touspoints = []
     for k in range(nbde1):
          if points[k] not in touspoints:
               touspoints.append(points[k])
          if pointsimage[k] not in touspoints:
               touspoints.append(pointsimage[k])
     for x in range(3,n//2+1,2):
          pointssurdiago = 0
          for p in touspoints:
               if x > sqrt(n) and p.imag-p.real == n-2*x:
                    pointssurdiago = pointssurdiago+1
          if pointssurdiago == 2:
               print(x,' decomposant ')
               plt.annotate(x,xy=(-1,n-2*x))
               plt.plot([1,x],[n+1-2*x,n-x],color='green')
               plt.annotate(n-x,xy=(n-2*x,-1))
               plt.plot([n+1-2*x,n-x],[1,x],color='green')
     plt.ylim(-2,n+2)
     plt.show()
```

On ne sait pas si cette représentation peut être utile pour trouver une démonstration de la conjecture de Goldbach binaire.

Un calcul fournit exactement le nombre de points de la figure associée à n (ou de la figure avant qu'on lui ait rajouté des points par interversion des coordonnées pour les points qui n'ont pas de symétrique par rapport à la droite $x = y$) : si on appelle D le nombre de points total et D_{init} le

nombre de points avant symétrisation on a :

$$D_{\text{init}} = n - 1 + \sum_{2 \leq p \leq \frac{n}{2}} \left(\left\lfloor \frac{n}{p} \right\rfloor - 1 \right)$$

et

$$D(n) = D_{\text{init}}(n) + \sum_{\substack{2 \leq p \leq \frac{n}{2} \\ p \nmid n}} \left(\left\lfloor \frac{n}{p} \right\rfloor - 1 \right)$$

On a deux définitions récursives de D et D_{init} qui sont :

$$D_{\text{init}}(n) = D_{\text{init}}(n-2) + \sum_{\substack{1 < d \leq n \\ d \mid n}} 1 + \sum_{\substack{1 < d \leq n-1 \\ d \mid n-1}} 1$$

et

$$D(n) = D_{\text{init}}(n) + \sum_{p \nmid n} \left(\left\lfloor \frac{n}{p} \right\rfloor - 1 \right)$$

Mais tant qu'on ne sait pas quels sont les booléens égaux à 1 qui appartiennent à la même diagonale, on ne peut être assuré qu'on ne va pas "bloquer" toutes les diagonales pour un certain n.