Des carrés écossais et des points fixes dans le plan complexe,
Denise Vella-Chemla,
7 août 2025

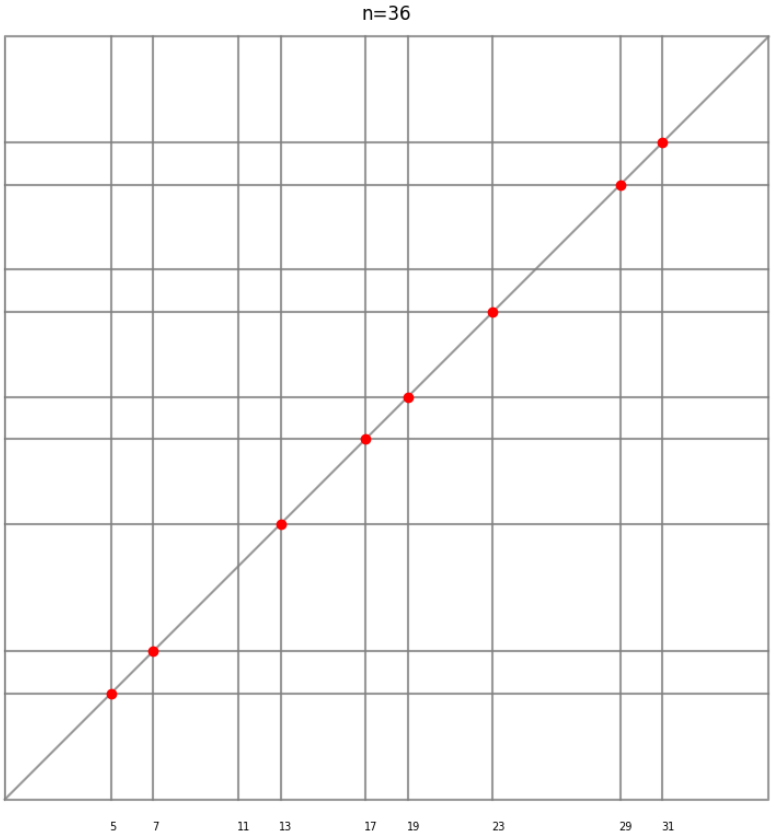
1. Présentation du problème

Toujours à la recherche d'une justification de la conjecture de Goldbach, on réfléchit à la façon de visualiser les décomposants de Goldbach dans le plan complexe.

On utilise un carré de côté de longueur n. Sur l'axe des abscisses x sont positionnés les nombres premiers impairs p_k compris entre 3 et $n-3$ inclus et qui ne divisent pas n (on note l'ensemble de ces nombres premiers $\mathcal{P}_n$).

On trace un réseau de droites dont les droites verticales ont pour équations $x = p_k$ pour chacun des nombres premiers de $\mathcal{P}_n$. On trace également des droites horizontales $y = n - p_k$; ce choix a pour conséquence que si l'on souhaite énumérer les nombres premiers de $\mathcal{P}_n$ le long des axes des coordonnées, on doit le faire comme habituellement de gauche à droite sur l'axe des abscisses, mais de haut en bas sur l'axe des ordonnées y (en s'approchant de plus en plus de l'origine).

Donnons un exemple : pour $n = 36$, le seul nombre premier "manquant" dans $\mathcal{P}_n$, car il ne divise pas n est 3. Les autres nombres premiers compris entre 3 et 33 inclus sont ceux de la liste : $[5, 7, 11, 13, 17, 19, 23, 29, 31]$.



2. Décomposants de Goldbach de n et symétrie par rapport à la diagonale SO-NE (d'équation $x = y$)

Pour mémoire, on rappelle que dans le plan complexe, l'image du point d'affixe z par une symétrie par rapport à la droite déterminée par les points a et b est $z' = \alpha\overline{z}+\beta$ avec $\alpha = \dfrac{a-b}{\overline{a}-\overline{b}}$ et $\beta = \dfrac{b\overline{a}-a\overline{b}}{\overline{a}-\overline{b}}$.

La diagonale SO-NE est déterminée par les points $a = 0$ et $b = n + ni$.

On a pour cette diagonale $\alpha = \dfrac{-n-ni}{-n+ni}$, $\beta = 0$.

Calculons les images des points du réseau de points choisi par la symétrie centrale par rapport à la diagonale SO-NE (ascendante, d'équation $y = x$). Le programme de calcul est fourni en annexe 1. Tout point de la forme $p + qi$ est envoyé sur le point $q + pi$. Le fichier des dessins des carrés pour les nombres n compris entre 6 et 103 est consultable à l'adresse :
https://denisevellachemla.eu/touscarrespourdg.pdf.

Les décomposants de Goldbach de n sont colorés en rouge sur la diagonale SO-NE.

Remarque : les nombres pairs qui sont des doubles d'un nombre premier (comme $38 = 2 \times 19$, ou $94 = 2 \times 47$) vérifient trivialement la conjecture de Goldbach, ils sont la somme de deux nombres premiers identiques. Leur carré présente un point rouge à l'intersection de ses diagonales, i.e. en plein milieu du carré).

Par un raisonnement simple, on comprend que si un décomposant de Goldbach existe, il est un point fixe d'une double application de cette symétrie par rapport à la diagonale SO-NE (d'équation $y = x$) et qu'il a donc ses coordonnées qui sont égales.

Voyons maintenant le nombre de points que contient le réseau des points.

Premier cas : si le nombre de nombres premiers compris entre 3 et $n - 3$ (inclus tous les deux) est impair ; son carré, qui compte le nombre de points du réseau est impair également (en effet, $(2k + 1)^2 = 4k^2 + 4k + 1 = (4k^2 + 4k) + 1 = 2k' + 1$), la fonction "symétrie par rapport à la diagonale du carré" a pour domaine un nombre impair de points. Mais dans la mesure où les points fixes d'une symétrie élevée au carré (une symétrie par rapport à une droite est une application idempotente : son carré est l'identité) se trouvent sur la droite par rapport à laquelle s'effectue cette symétrie, si le réseau contient un nombre impair de points, et si les points en dehors de la diagonale vont par deux : $p + iq$ s'envoie sur $q + ip$ et retour, et inversement, $q + ip$ s'envoie sur $p + iq$ et retour, la diagonale doit contenir au moins un point fixe ; dit autrement, le point "en trop par rapport à la parité" (i.e. le "+1" du $2k + 1$) doit se trouver sur la diagonale. On a ainsi mis au jour au moins un décomposant de Goldbach de n.

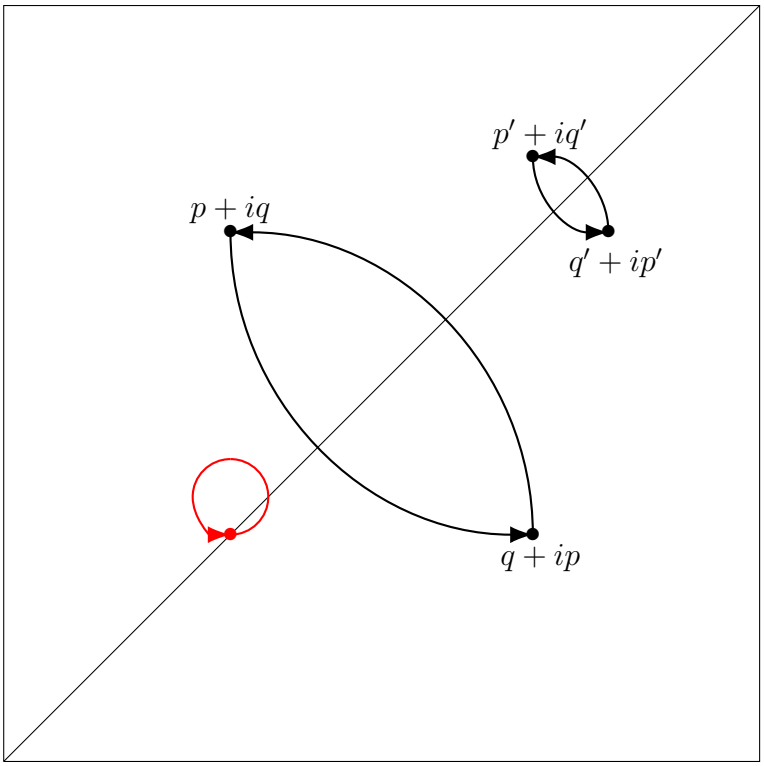
Second cas : le nombre de nombres premiers compris entre 3 et $n-3$ (inclus tous les deux) est pair; son carré, qui compte le nombre de points du réseau est pair : on n'est pas assuré d'avoir un point fixe "tout seul". On choisir alors d'ajouter à l'ensemble des nombres premiers $\mathcal{P}_n$ un seul diviseur de n. Cela a pour effet de rendre le cardinal de $\mathcal{P}_n$ impair, et par voie de conséquence son carré

impair également et on est ramené au premier cas, pourquoi ? Car un diviseur de n ne peut en aucun cas être un décomposant de Goldbach de n. Cela s'exprime par le langage des congruences de Gauss ainsi :

$$(p \text{ divise } n) \wedge (p \text{ premier}) \iff p \text{ divise } n - p.$$

Le diviseur de n en question se retrouve associé à son symétrique par rapport à la diagonale descendante par un double aller-retour entre eux (il est de la forme $p + (n - p)i$ avec $p \neq q$ et son "correspondant" est de la forme $(n - p) + pi$.

On résume le raisonnement par le schéma suivant :



On a ainsi, peut-être, trouvé une justification de la véracité de la conjecture de Goldbach.

Annexe 1 : programme python utilisé pour les expérimentations qui corrobore l'idée présentée dans la note

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

def prime_sieve(N):
     is_prime = np.full(N, True)
     is_prime[:2] = False
     for p in range(2, math.isqrt(N) + 1):
          if is_prime[p]:
               is_prime[p*p::p] = False
     return np.nonzero(is_prime)[0]

class Primes():
     def __init__(self, N):
          self.__primes = prime_sieve(N)
     def __str__(self):
          return str(self.__primes)
     def __iter__(self):
          return iter(self.__primes)
     def __len__(self):
          return self.__primes.size
     def __getitem__(self, k):
          return self.__primes[k]
     def __contains__(self, x):
          k = self.index(x)
          return k < len(self) and self.__primes[k] == x
     def index(self, x):
          return np.searchsorted(self.__primes, x, side='left')
     def count(self, x):
          return np.searchsorted(self.__primes, x, side='right')
     def range(self, start, stop, step=1):
          return self.__primes[self.index(start):self.index(stop):step]
     def factors(self, n):
          if n in self:
               return np.array([n])
          else:
               P = self.range(2, n//2 + 1)
               return P[n % P == 0]

def syme(z,a,b):
     abar = a.real-a.imag*1j
     bbar = b.real-b.imag*1j
     zbar = z.real-z.imag*1j
     aaalpha = (a-b)/(abar-bbar)
     bbbeta = (b*abar-a*bbar)/(abar-bbar)
     return(aaalpha*zbar+bbbeta)
```

```
N = 104
P = Primes(N+1)
for n in range(6, 104, 2):
     trouve = False
     _,ax = plt.subplots(figsize=(10,10))
     milieu = n//2
     lf = P.factors(n)
     l1 = P.range(3, n - 3 + 1)
     l1 = np.array([x for x in l1 if x not in lf or x == milieu])
     l2 = n - l1
     listecomplexes = []
     for x in range(len(l1)):
          for y in range(len(l1)):
               listecomplexes.append(l1[x]+1j*(n-l1[y]))
     print('liste complexes = ',listecomplexes)
     print('diago')
     for z in listecomplexes:
          print(z,' image :  ',syme(z,0,n+n*1j))
     lg = np.array([x in P for x in l2])
     print(f' n = n (facteurs = lf) '.center(80, '_') + f'1 = l1'+ f'2 = l2')
     print('lg = [',end='')
     for k in range(len(lg)):
          if lg[k]:
               print(l1[k],' ',end='')
     print(']')
     for k in range(len(l1)):
          plt.plot([l1[k],l1[k]],[0,n],color='gray',alpha=0.8,zorder=0)
          plt.plot([0,n],[n-l1[k],n-l1[k]],color='gray',alpha=0.8,zorder=0)
     plt.plot([n,l1[-1]],[n,l1[-1]],color='gray',alpha=0.8,zorder=0)
     plt.plot([0,l1[0]],[0,l1[0]],color='gray',alpha=0.8,zorder=0)
     plt.plot([0,n,n,0,0],[0,0,n,n,0],color='gray',alpha=0.8,zorder=0)
     plt.plot(l1,l1,color='gray',alpha=0.8,zorder=0)
     for x in range(len(l1)):
          plt.annotate(l1[x],xy=(l1[x]-1,-2),fontsize=7)
     plt.scatter(l1[lg],l1[lg],marker='o',s=36,color='red',zorder=0)
     plt.title('n='+str(n))
     plt.axis('equal')
     plt.xlim(-2,n+2)
     plt.ylim(-5,n+2)
     plt.show()
     nomfic ='carreRQ'+str(n)
     plt.savefig(nomfic)
     plt.close()
```