**Multiplication par $i$, images sur la diagonale et décomposants de Goldbach, Denise Vella-Chemla, 22 juillet 2025.**

Toujours à la recherche d'une explication de l'existence systématique d'un décomposant de Goldbach pour un nombre pair $n$ ($> 2$) , i.e. un nombre premier $p$ dont le complémentaire à $n$ (i.e. $n-p$) est premier aussi, on met au point les représentations graphiques dont celle associée à $n = 40$ ci-dessous.

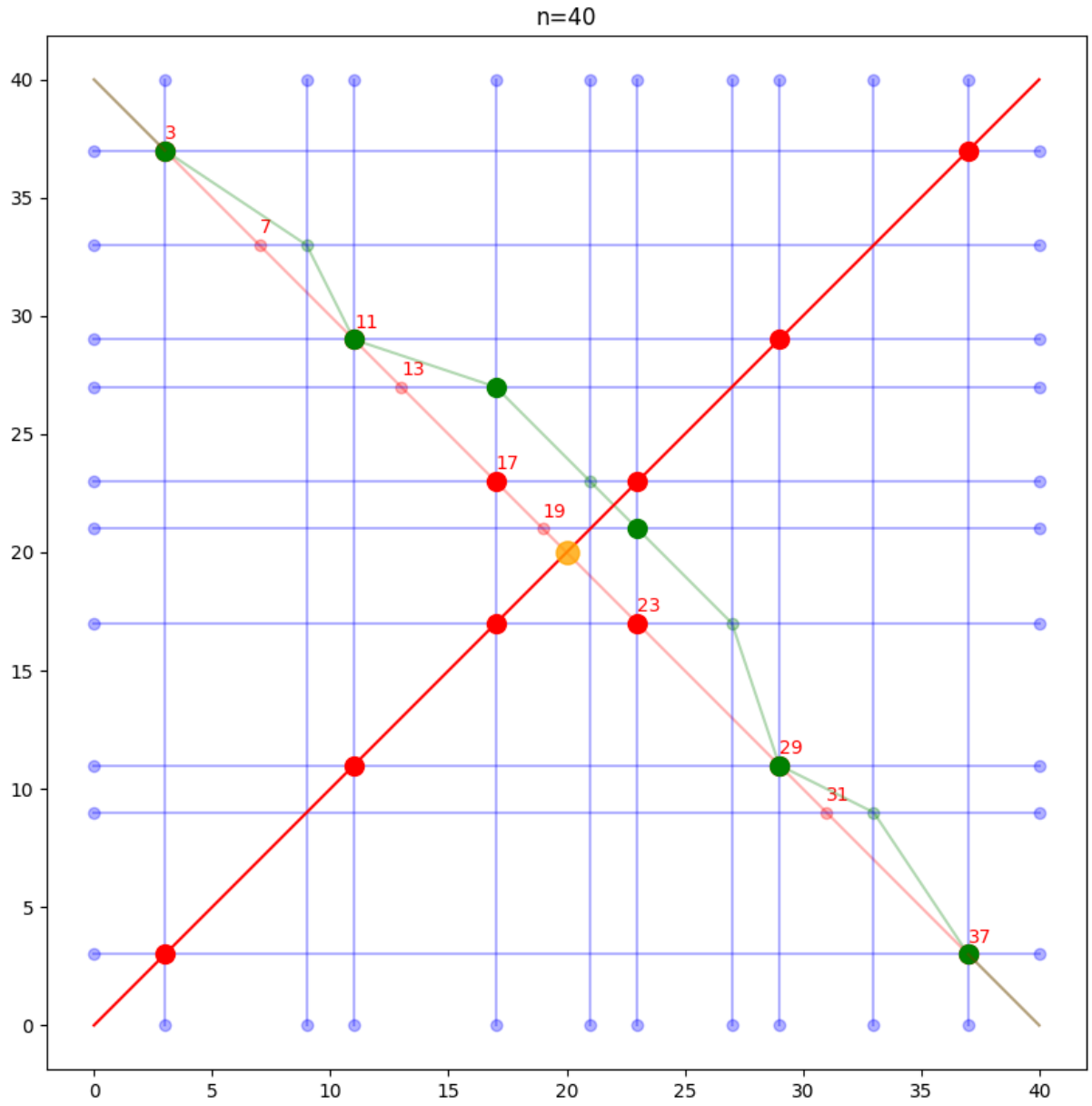


FIG. 1

Les graphiques associés aux nombres $n$ compris entre 8 et 102, sont stockés à l'adresse https://denisevellachemla.eu/dessins-foisi.pdf .

On utilise la liste $liste1$ des nombres premiers impairs de 3 à $n-3$ (dont ont été ôtés les diviseurs de $n$), la liste $liste2$ des complémentaires de ces nombres à $n$, et la liste $liste3$ des complémentaires en question, mais écrite dans l'ordre inverse, du dernier au premier d'entre eux.

On note par un rond le point de coordonnées $(n/2, n/2)$.

On note par deux lignes les diagonales $y = x$ (de $(0,0)$ à $(n,n)$) et $y = n - x$ (de $(0,n)$ à $(n,0)$).

Sur la diagonale descendante $x + y = n$, sont notés les nombres premiers, sauf ceux qui divisent $n$ (excepté $n/2$ quand il est premier), car les diviseurs de $n$ ne peuvent pas être des décomposants de Goldbach de $n$ (si ce n'est le décomposant trivial $n/2$, lorsqu'il est premier).

La courbe verte relie les points de coordonnées $(x, y)$ avec $x$ parcourant la liste $liste2$ et $y$ parcourant la liste $liste3$.

On voit que les décomposants de Goldbach sont les points de la courbe verte qui sont envoyés par rotation d'un quart de tour dans le sens horaire (i.e. par multiplication par $i$) sur la diagonale descendante rouge.

Il faut cependant être attentif au fait que pour effecture la rotation de 90° par multiplication par $i$, on doit d'abord "ramener le point autour de l'origine" car le graphique associé à $n$ est centré en $(n/2, n/2)$. On doit donc translater le point $(x, y)$ en lui ajoutant le point $(-n/2, -n/2)$, puis le multiplier par $i$, puis le "ramener à sa nouvelle place" en effectuant la translation opposée en lui ajoutant le point $(n/2, n/2)$. Les seuls points qui sont des décomposants de Goldbach de $n$ s'envoient sur la diagonale ascendante par cette transformation $T^{-1}RT$ avec $T$ la translation de vecteur $(n/2, n/2)$.

Un tel point existe toujours mais on ne sait pas le justifier. Il nous semble que l'espace dans lequel on se situe ici est le 2-tore non-commutatif, dans la mesure où les écarts entre les lignes verticales (resp. horizontales) en haut et en bas (resp. à gauche et à droite) du carré sont égaux mais non réguliers. On aurait alors comme unité de longueur sur le 2-tore en question $\frac{1}{\sqrt{2}}$ la mesure de la diagonale du carré la plus petite possible : entre deux nombres (premiers) impairs (les seuls décomposants de Goldbach possible pour les nombres pairs $\geq 6$), l'écart minimum est 2.

Le programme de vérification de cette idée est fourni en fin de note.

Voici son résultat pour le nombre pair $n = 40$.

n = 40 (facteurs = [2 5])

l1 = [ 3 7 11 13 17 19 23 29 31 37]
l2 = [37 33 29 27 23 21 17 11 9 3]
l3 = [ 3 9 11 17 21 23 27 29 33 37]
listepoints = [(3+37j), (9+33j), (11+29j), (17+27j), (21+23j), (23+21j), (27+17j), (29+11j), (33+9j), (37+3j)]

p = (3+37j)
image de p = (32+32j)
p = (9+33j)
image de p = (36+38j)
p = (11+29j)
image de p = (40+40j)
p = (17+27j)
image de p = (42+46j)
p = (21+23j)
image de p = (46+50j)
p = (23+21j)
image de p = (48+52j)
p = (27+17j)
image de p = (52+56j)
p = (29+11j)
image de p = (58+58j)
p = (33+9j)
image de p = (60+62j)
p = (37+3j)
image de p = (66+66j)

Les images des décomposants de Goldbach 3 et 37, ou 11 et 29, correspondant aux points $3 + 37i$ et $11 + 29i$, qui sont $32 + 32i$ et $40 + 40i$ sont sur la diagonale ascendante.

```
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]

N = 104
P = Primes(N + 1)
for n in range(8, N, 2):
     _,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
     l3 = np.flip(l2)
     lg = np.array([x in P for x in l2])
     print(f' n = n (facteurs = lf) '.center(80, '_') + f'1 = l12 = l23 = l3')
     plt.plot(l1, l2, color='red', marker='o', zorder=0,alpha=0.3)
     for k in range(len(l1)):
          plt.annotate(str(l1[k]), xy=(l1[k], l2[k]+0.5), color='red', zorder=0)
     plt.plot(l2, l3, color='green', marker='o', zorder=0,alpha=0.3)
     plt.scatter(l1[lg], l1[lg], marker='o',color='red', s=100, zorder=1)
     plt.scatter(l1[lg], l2[lg], marker='o',color='red', s=100, zorder=1)
     plt.scatter(l2[lg], l3[lg], marker='o',color='green', s=100, zorder=1)
     listepoints = []
     for k in range(len(l2)):
          plt.plot([0,n], [l3[k],l3[k]], color='blue', marker='o',
zorder=0,alpha=0.3)
          plt.plot([l3[k],l3[k]],[0,n], color='blue', marker='o', zorder=0,alpha=0.3)
          listepoints.append(l3[k]+1j*l2[k])
     print('listepoints = ',listepoints)
     plt.scatter(milieu,milieu,marker='o',color='orange',s=150,zorder=3,alpha=0.8)
     plt.axis('equal') ; plt.plot([0,n],[0,n],color='red')
     nb = len(l1)
     for k in range(nb):
          x1 = l1[k] ; y1 = l2[k]
     plt.plot([0,l1[0]],[n,l2[0]],color='red',alpha=0.3)
     plt.plot([n,l1[-1]],[0,l2[-1]],color='red',alpha=0.3)
     plt.plot([0,l2[-1]],[n,l3[-1]],color='green',alpha=0.3)
     plt.plot([n,l2[0]],[0,l3[0]],color='green',alpha=0.3)
     plt.title('n='+str(n))
     plt.show() ; nomfic ='arc'+str(n) ; plt.savefig(nomfic) ; plt.close()
     for p in listepoints:
          print('p = ',p) ; a = p.real ; b = p.imag
          translate = (a-milieu)+(b-milieu)*1j
          translatetournedei = 1j*translate
          revient = (translatetournedei.real+49)+(translatetournedei.imag+49)*1j
print('image de p = ',revient)
```