Positionner les nombres sur le disque-unité par leurs restes de divisions euclidiennes (Denise Vella-Chemla, 28.8.2020)

On voudrait fournir ici des résultats surprenants qui nous ont été inspirés par un travail tout l'été autour de la preuve par Alain Connes du théorème de Morley ([1]).

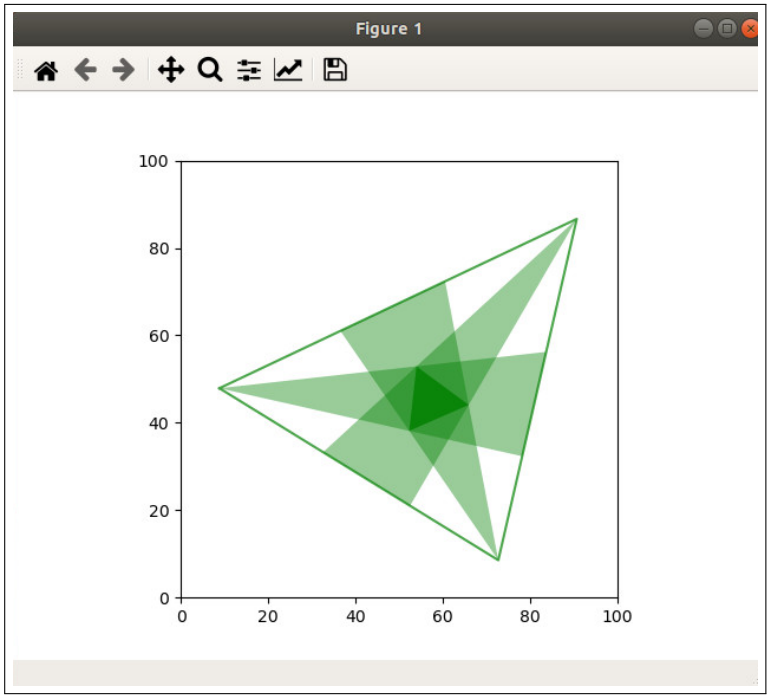
En effet, on souhaitait programmer la preuve pour essayer de vérifier que le théorème n'est pas applicable pour les triangles sphériques (voir notamment la conférence [2]).

Programme de vérification et visualisation du Théorème de Morley dans le plan

```
from math import *
from matplotlib import *
from matplotlib.pyplot import *

def vecteur(point1, point2):
  return [y - x for x, y in zip(point1, point2)]

def add(vecteur1, vecteur2):
  return [x + y for x, y in zip(vecteur1, vecteur2)]

def norme(vecteur):
  return sqrt(prodscal(vecteur, vecteur))

def prodscal(vecteur1, vecteur2):
  return sum([x*y for x, y in zip(vecteur1, vecteur2)])

def determ(vecteur1, vecteur2):
 return vecteur1[0]*vecteur2[1]-vecteur1[1]*vecteur2[0]

def angle(vecteur1, vecteur2):
  cosinus=prodscal(vecteur1, vecteur2)/(norme(vecteur1)*norme(vecteur2))
  sinus=determ(vecteur1, vecteur2)/(norme(vecteur1)*norme(vecteur2))
  return atan2(sinus,cosinus)

def rotation(u, theta):
  return [u[0]*cos(theta)-u[1]*sin(theta),u[0]*sin(theta)+u[1]*cos(theta)]

def intersecte(x, y, z, t):
  a1 = (y[1]-x[1])/(y[0]-x[0])
  b1 = x[1]-a1*x[0]
  a2 = (t[1]-z[1])/(t[0]-z[0])
  b2 = z[1]-a2*z[0]
  return [(b2-b1)/(a1-a2),((b2-b1)/(a1-a2))*a1+b1]

a=[8.83, 47.89]
b=[72.74, 8.55]
c=[90.72, 86.63]
ab = vecteur(a,b) ; ac = vecteur(a,c)
ba = vecteur(b,a) ; bc = vecteur(b,c)
ca = vecteur(c,a) ; cb = vecteur(c,b)
anglea=angle(ab,ac)  ; angleb=angle(bc,ba)  ; anglec=angle(ca,cb)
aprime = add(a, rotation(ab,anglea/3.0))  ; aseconde = add(a, rotation(ab,anglea*2.0/3.0))
bprime = add(b, rotation(bc,angleb/3.0)) ; bseconde = add(b, rotation(bc,angleb*2.0/3.0))
cprime = add(c, rotation(ca,anglec/3.0)) ; cseconde = add(c, rotation(ca,anglec*2.0/3.0))
p=intersecte(a,aseconde,b,c) ; q=intersecte(a,aprime,b,c)
r=intersecte(c,cseconde,a,b) ; s=intersecte(c,cprime,a,b)
t=intersecte(b,bseconde,c,a) ; u=intersecte(b,bprime,c,a)
n=intersecte(a,aseconde,c,cprime)
o=intersecte(c,cseconde,b,bprime)
x=intersecte(b,bseconde,a,aprime)
print('Normes des cotes %3.15f '% norme(vecteur(n,o)))
print('Normes des cotes %3.15f '% norme(vecteur(o,x)))
print('Normes des cotes %3.15f '% norme(vecteur(x,n)))

fig = matplotlib.pyplot.figure()
ax = fig.add_subplot(111)
matplotlib.pyplot.plot([a[0],b[0],c[0],a[0]],[a[1],b[1],c[1],a[1]], 'g', alpha=0.7)
matplotlib.pyplot.fill([a[0],p[0],q[0]],[a[1],p[1],q[1]], 'g', 2, alpha=0.4)
matplotlib.pyplot.fill([c[0],r[0],s[0]],[c[1],r[1],s[1]], 'g', 2, alpha=0.4)
matplotlib.pyplot.fill([b[0],t[0],u[0]],[b[1],t[1],u[1]], 'g', 2, alpha=0.4)
matplotlib.pyplot.fill([n[0],o[0],x[0]],[n[1],o[1],x[1]], 'g', 2, alpha=0.8)
matplotlib.pyplot.xlim(0,100)
matplotlib.pyplot.ylim(0,100)
ax.set\_aspect('equal')
matplotlib.pyplot.show()
```

Le programme ci-dessus produit la visualisation ci-après, et imprime comme longueur des normes des côtés du triangle de Morley (le triangle équilatéral de sommets les intersections des trissectrices adjacentes du triangle quelconque "externe") la valeur 14.863746806168091 pour deux côtés et la valeur 14.863746806168082 pour

le troisième côté : la différence entre les valeurs est négligeable, les côtés sont de longueur égale, le théorème le prouve, même si l'ordinateur ne le constate qu'à un ε près (d'ailleurs, l'ordinateur détecte toujours l'égalité de flottants (les réels en langage informatique) à un ε près).



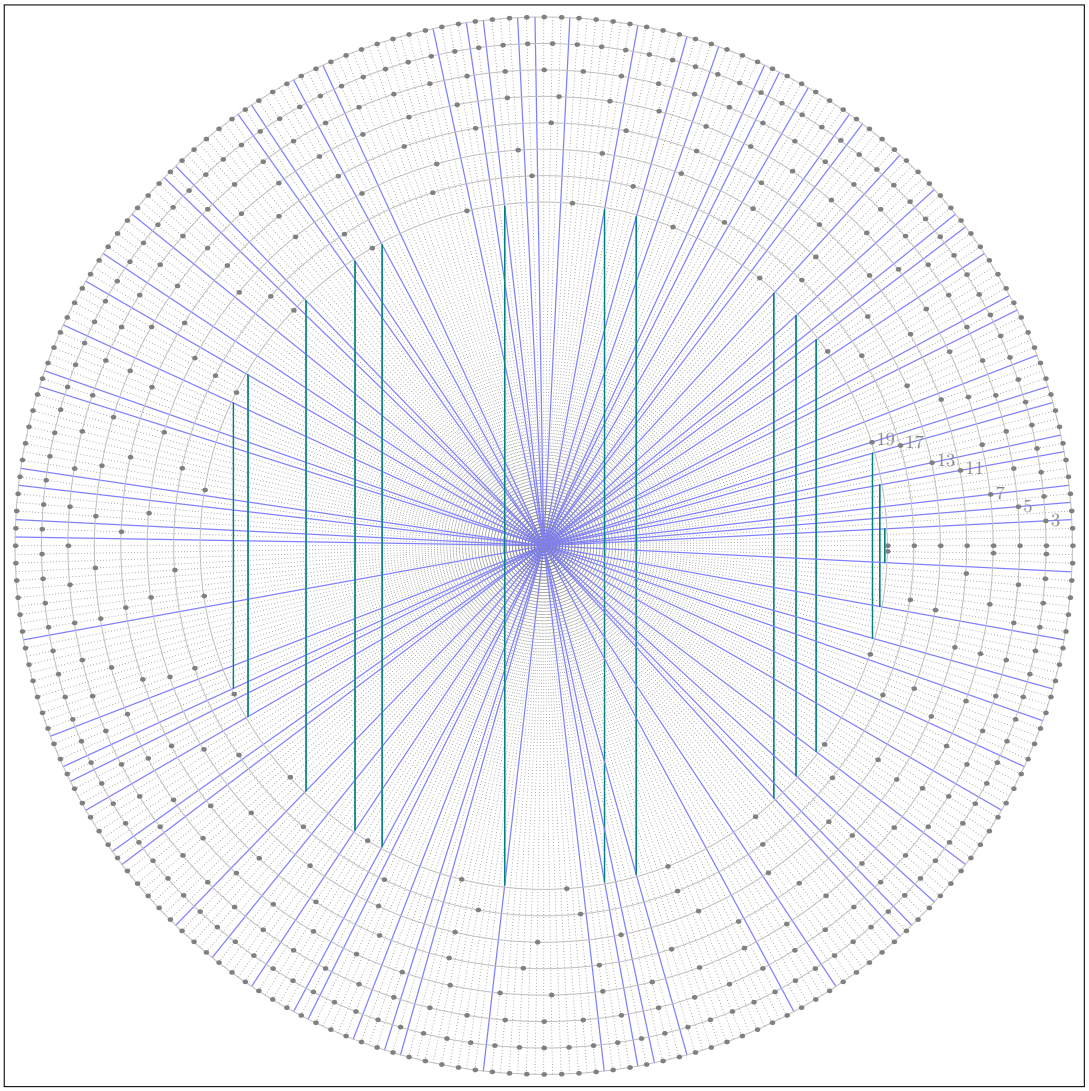
On avait également programmé des rotations dans le cercle unité en langage Asymptote, puis en langage python, pour visualiser les décomposants de Goldbach sur le cercle par les programmes suivants :

Programme asymptote de la visualisation de Goldbach des décomposants de 400 :

```
unitsize(1cm);
real xmax = 12;

pen prem = lightblue;
pen couleurdot = gray;
pen comp = dotted+gray;
pen lieprem = deepcyan;

int n = 400;
int m = floor(sqrt(n));

int[] premier = {2};

bool[] est_premier;
for (int k = 0; k < n; ++k) {
  est_premier.push(false);
}

bool is_prime(int p) {
  bool flag = true;
  for (int k = 0; k < premier.length; ++k) {
    if (p % premier[k] == 0) {
      flag = false;
      break;
    }
  }
  return flag;
}

for (int p = 3; p <= n; p = p+2) {
  if (is_prime(p)) {
    premier.push(p);
    est_premier[p] = true;
  }
}
```

```
path cercle = scale(xmax)*unitcircle;
path cercle2 = scale(7.75)*unitcircle;
for (int k = 0; k < n; ++k) {
  real t = length(cercle)*k/n;
  real t2 = length(cercle)*(n-k)/n;
  pair P = point(cercle, t);
  pair P2 = point(cercle2, t);
  pair P3 = point(cercle2, t2);
  if (est_premier[k]) {
    draw((0,0)--P, prem+0.8bp);
    if (est_premier[n-k])
      draw(P2--P3, lieprem+0.8bp);
  } else {
    draw((0,0)--P, comp);
  }
}

for (int k = 0; premier[k] < m; ++k) {
  path cercle = scale(xmax*(1-k/m))*unitcircle;
  draw(cercle, mediumgray);
  for (int d = 0; d < n; d = d+premier[k]) {
    real t = length(cercle)*d/n;
    pair P = point(cercle, t);
    if (d <= m) {if (est_premier[d]) dot(format("%d",d), P, dir(P), couleurdot);
                 else dot(P,couleurdot);}
    else dot(P, couleurdot);
  }
}
```

qui produit la visualisation ci-dessous ($400 = 3 + 397 = 11 + 389 = 17 + 383 = 41 + 359 = 47 + 353 = 53 + 347 = 83 + 317 = 89 + 311 = 107 + 293 = 131 + 269 = 137 + 263 = 149 + 251 = 167 + 233 = 173 + 227$).



Le même programme en python / matplotlib fournit une visualisation moins fine.

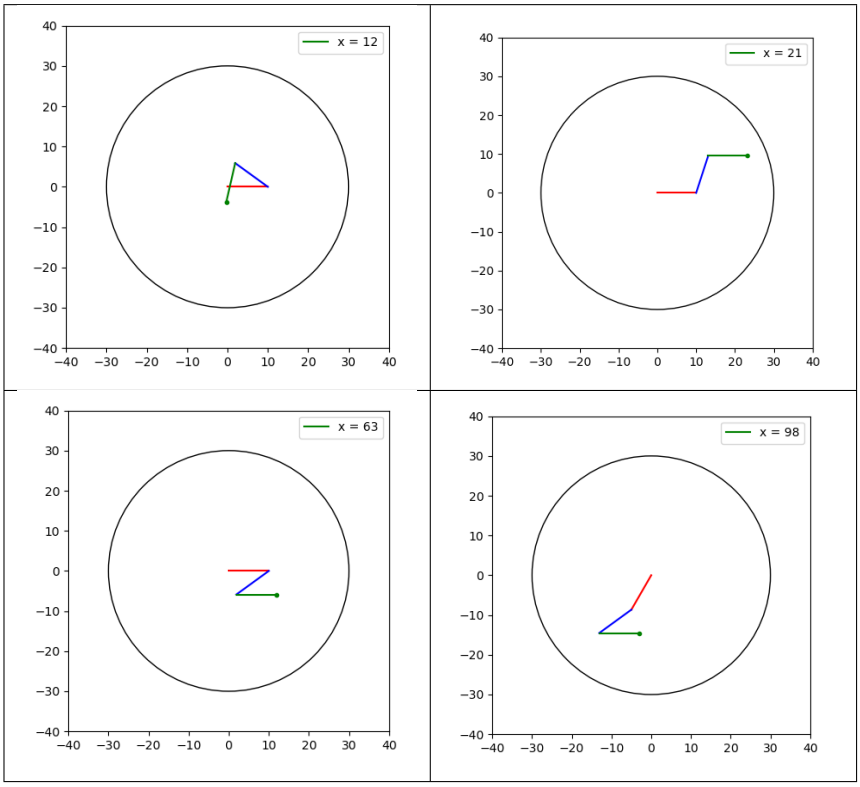
```
from math import *
from matplotlib import *
import matplotlib.pyplot as plt

def prime(atester):
  pastrouve = True
  k = 2
  if (atester == 0): return False
  if (atester == 1): return False
  if (atester == 2): return True
  if (atester == 3): return True
  if (atester == 5): return True
  if (atester == 7): return True
  while (pastrouve):
    if ((k * k) > atester):
      return True
    else:
      if ((atester % k) == 0):
        return False
      else: k=k+1

fig = plt.gcf()
ax = fig.gca()
xmax = 50
n = 98
m = floor(sqrt(n))
ax = plt.gca() ; ax.cla() ; plt.axis('scaled')
num = 0
for p in range(1,m):
  if (prime(p)):
    rayon = xmax*(1-num/m)
    cercle = plt.Circle((0,0), rayon, fill=False)
    num = num+1
    ax.add_artist(cercle)
num = 0
for p in range(1,m):
  if (prime(p)):
    for d in range(n):
      if ((d % p) == 0):
        t = 2*pi*d/n
        rayon = xmax*(1-num/m)
        ax.plot(rayon*cos(t), rayon*sin(t), 'bo', markersize=3)
        ax.text(rayon*cos(t), rayon*sin(t), str(d), color='blue',
                fontsize=9, va='bottom', ha='right', alpha=0.85)
    num = num+1
cercle = xmax
xmin = 33
cercle2 = 33
for q in range(n):
  t = 2*pi*q/n
  t2 = 2*pi*(n-q)/n
  xP, yP = xmax*cos(t), xmax*sin(t)
  xP2, yP2 = xmin*cos(t), xmin*sin(t)
  xP3, yP3 = xmin*cos(t2), xmin*sin(t2)
  xs = [0,xP]
  ys = [0,yP]
  if (prime(q)):
    plt.plot(xs, ys, color='fuchsia')
    xs = [xP2,xP3]
    ys = [yP2,yP3]
    if (prime(n-q)):
      plt.plot(xs, ys, color='chartreuse')
  else:
    plt.plot(xs, ys, color='gray', ls=':')
ax.set_xlim(((-1)*xmax-10, xmax+10))
ax.set_ylim(((-1)*xmax-10, xmax+10))
plt.show()
```

De fil en aiguille, on a eu l'idée d'un programme, qui utiliserait notre modélisation par les restes pour la conjecture de Goldbach (le Snurp∞, cf. [3]) ainsi que des rotations.

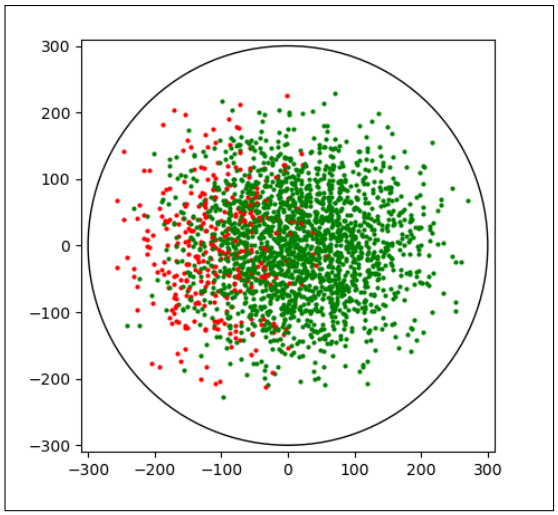
Présentons l'idée sur quelques exemples. On se place sur le disque unité. On représente chaque nombre comme un petit chemin, depuis le centre du cercle unité, jusqu'à un point du disque à déterminer. Le chemin est une ligne brisée dont les segments sont obtenus comme des rotations d'angles correspondant aux restes du nombre dans les différents corps premiers $\mathbb{Z}/p_i\mathbb{Z}$.

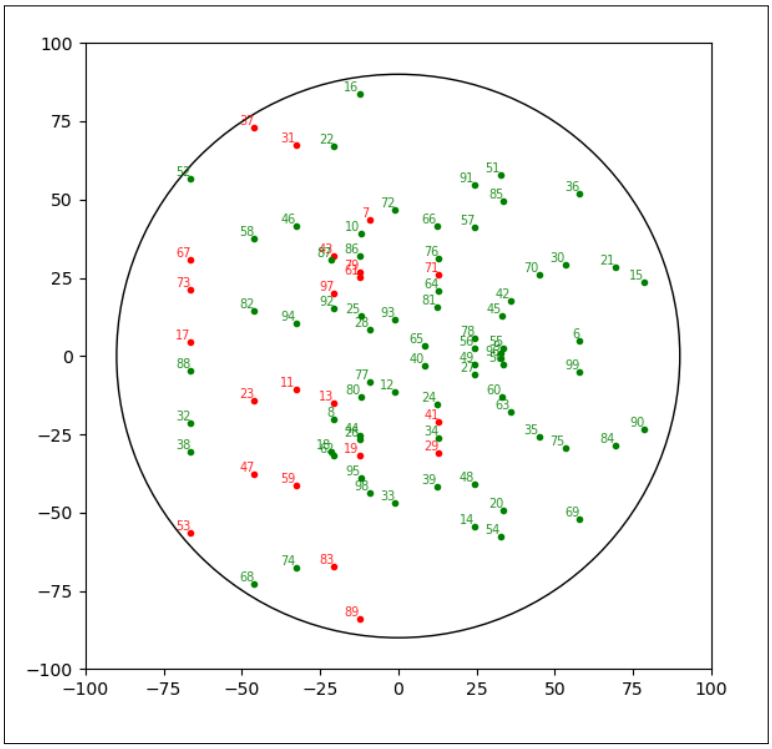
Ci-dessous, sont fournis les petits chemins pour les nombres 12, 21, 63 et 98, en ne gérant que 3 restes modulaires, le reste modulo 3 partant du centre et en rouge, suivi du reste modulo 5 en bleu, suivi du reste modulo

7 en vert. Le dernier point atteint est visualisé par un petit cercle plein vert. Par exemple, la “manivelle” pour 21 représente le fait que 21 est congru à 0 modulo 3, à 1 modulo 5 (angle orienté vers la première racine $5^{\text{ème}}$, à $e^{\frac{2i\pi}{5}}$) de l'unité, et congru à 0 modulo 7 (on voit également $12 \equiv 0 \pmod 3; 12 \equiv 2 \pmod 5; 12 \equiv 5 \pmod 7$ ou $21 \equiv 0 \pmod 3; 21 \equiv 1 \pmod 5; 21 \equiv 0 \pmod 7$ ou enfin $98 \equiv 2 \pmod 3; 98 \equiv 3 \pmod 5; 98 \equiv 0 \pmod 7$).



On a dans un premier temps utilisé nos petits chemins dans les corps premiers pour les nombres premiers considérés “dans l'ordre”, de 2, 3, 5,... à 13, du plus petit au plus grand, et on était un peu découragée car les “clusters” de points ne semblaient pas être aisément circonscriptibles, comme on le voit sur les 2 visualisations ci-dessous.

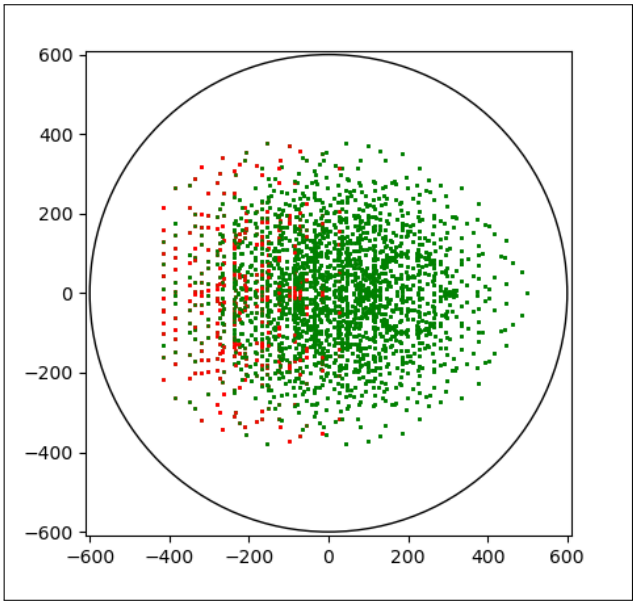




On utilise un programme sur le modèle de celui ci-dessous :

```
import math
from math import *
import matplotlib.pyplot as plt

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

n = 30 ; x = 1 ; xmax = 100 ; fig = plt.figure() ; ax = fig.gca() ;
Ens = [2,3,5]
nbprime = len(Ens) + 1 ; print(str(nbprime))
ax.set_xlim((-1)*nbprime*xmax-10,nbprime*xmax+10) ;
ax.set_ylim((-1)*nbprime*xmax-10,nbprime*xmax+10)
cercle = plt.Circle((0,0), nbprime*xmax, fill=False)
ax.add_artist(cercle)

#plt.plot(0, 0, 'black', marker='*', markersize=8)
xprec,yprec = 0, 0
while x <= n:
  x0, y0 = 0, 0
  for element in Ens:
    t = 2*pi*(x % element)/element
    x0 = x0 + xmax*math.cos(t)
    y0 = y0 + xmax*math.sin(t)
  #print(str(x0)+'   '+str(y0))
  if (prime(x)):
    plt.plot(x0, y0, 'r', marker='o', markersize=1)
    ax.text(x0, y0, str(x), color='r', fontsize=8, ha='right', alpha=0.8)
  else:
    plt.plot(x0, y0, 'g', marker='o', markersize=1)
    ax.text(x0, y0, str(x), color='g', fontsize=8, ha='right', alpha=0.8)
  if (x != 1):
    ax.plot([xprec,x0],[yprec,y0],'gray', alpha=0.25)
  xprec = x0
  yprec = y0
  ax.set_aspect('equal')
  x = x+1
plt.show()
```

Voici le résultat de ce programme pour les entiers inférieurs à 10000.



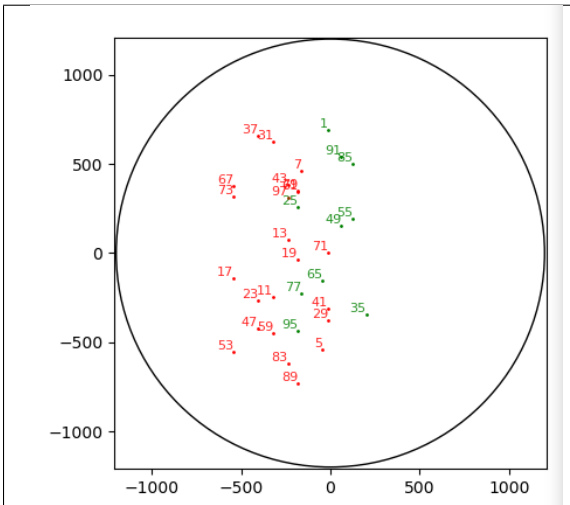
Le positionnement symétrique des points (pas forcément de la même couleur) par rapport à l'axe des abscisses se justifie par le fait qu'il y a probabilistiquement autant de points de reste x dans une division par p qu'il y en a de reste $p - x$ dans une division par p.

Des expérimentations informatiques avec des programmes similaires s'avèrent surprenantes, même si très difficiles à comprendre a priori. On fournit une ébauche d'explication mais voyons les résultats.

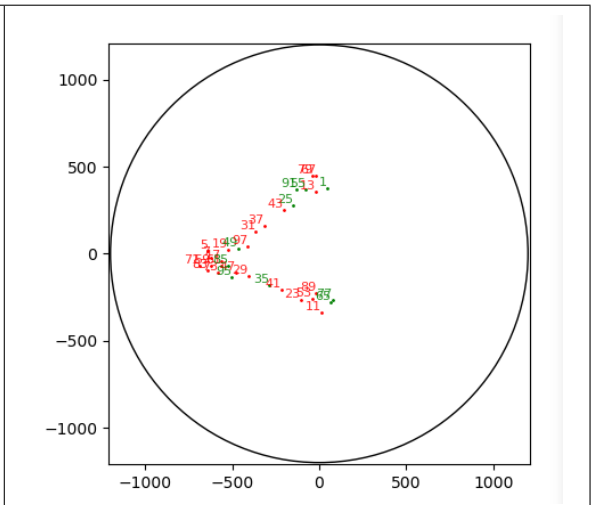
On réalise qu'on obtient des résultats qui semblent intéressants, ou du moins des alignements (même si on n'arrive pas à mettre les nombres premiers et les nombres composés sur des droites distinctes), en "isolant" les restes dans 4 corps premiers, dont les corps $\mathbb{Z}/3\mathbb{Z}$ et $\mathbb{Z}/2\mathbb{Z}$.

On a aussi l'impression que les visualisations présentent de meilleures propriétés si les deux autres corps premiers sont des corps de nombres premiers jumeaux. Enthousiasmée par les visualisations, on a même tenté 6 corps premiers au lieu de 4 (les nombres premiers 2 et 3, et 2 paires de nombres premiers jumeaux).

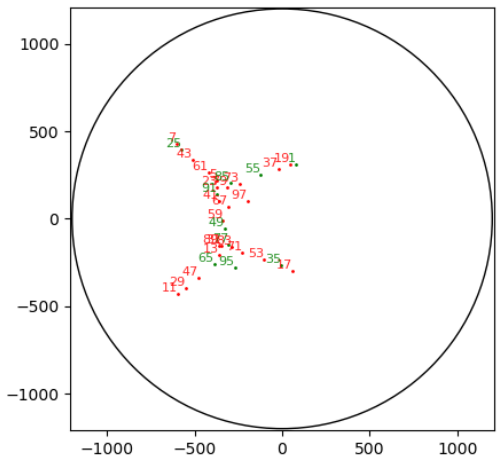
Les alignements sont provoqués par le fait d'avoir des restes simultanément complémentaires dans les différents corps premiers. Par exemple, 67 et 73 sont alignés selon les restes 7 et 5 parce qu'on a à la fois $67 \bmod 7 + 73 \bmod 7 = 7$ et $67 \bmod 5 + 73 \bmod 5 = 5$. Les cercles contiennent parfois des nombres de même reste dans des divisions euclidiennes par un certain nombre.



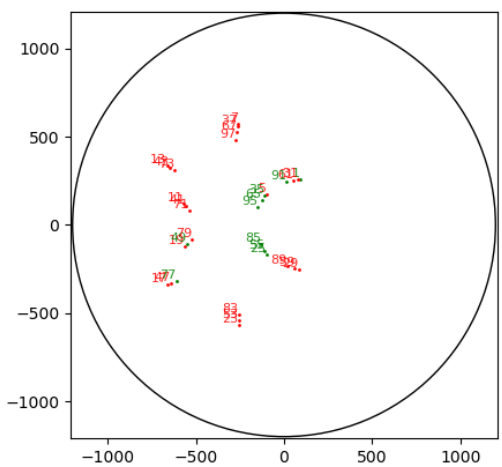
Premiers 7, 5, 3, 2.



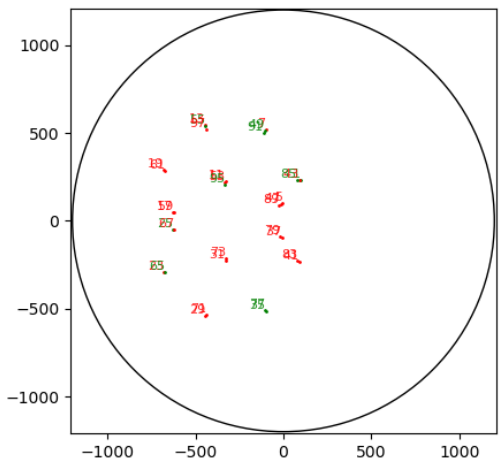
Premiers 13, 11, 3, 2.



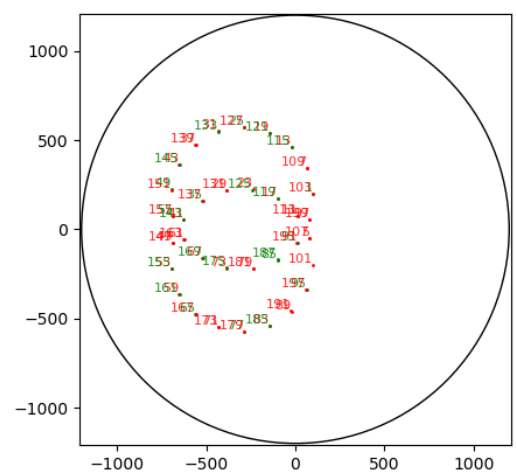
Premiers 19, 17, 3, 2.



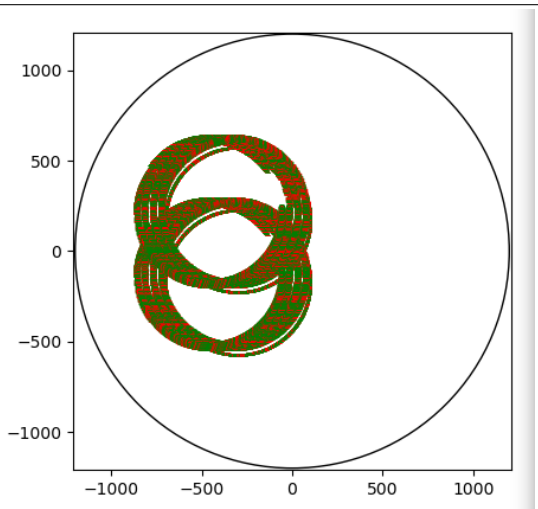
Premiers 31, 29, 3, 2.



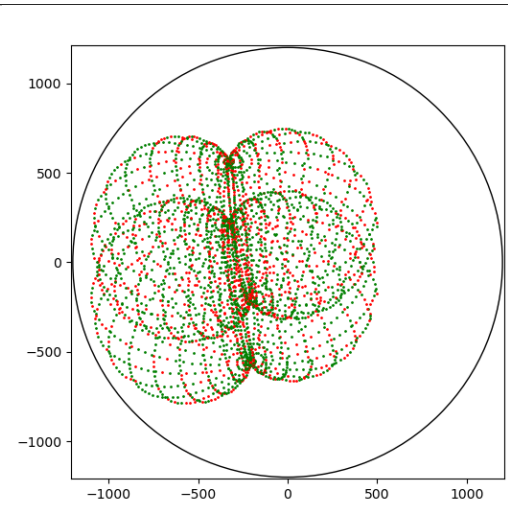
Premiers 43, 41, 3, 2.



Premiers 103, 101, 3, 2.



Premiers 9931, 9929, 3, 2.

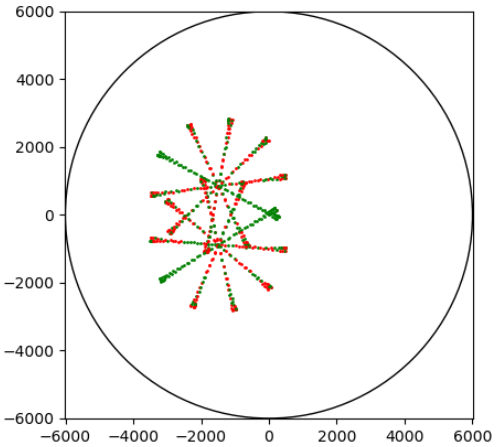


Premiers 9931, 9929, 103, 101, 3, 2.

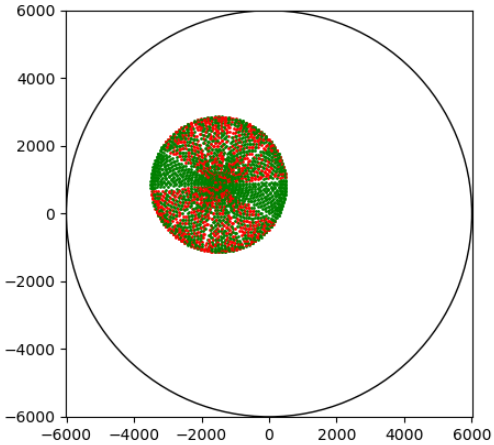
□ □

Dans la visualisation 13-11-3-2 (sorte de bec d'oiseau), les deux branches correspondent à nouveau aux $6x+1$, $6x-1$ (6 divise 12 entre les jumeaux 11 et 13). Dans la visualisation suivante 19-17-3-2, il y a un embranchement supplémentaire sur chacune des branches haute et basse. Le haut et le bas sont bien séparés comme d'habitude en $6x+1$ et $6x-1$, puis par exemple en haut, ce sont les $9x-1$ et les $9x+1$ qui sont dans les branches droite et gauche (car 9 divise 18 entre les jumeaux). Pour la visualisation 31-29, ce sont comme attendu des séparations selon deux pentagones, 5 divisant 30 entre les jumeaux 29 et 31. La visualisation suivante 43-41 séparant en heptagones. Pour ces deux séparations pentagonale et heptagonale, on constate avec intérêt que certains sommets ne semblent contenir que des premiers, ça reste à vérifier. Pour le 103-101, comme attendu, 102=2.3.17, on a deux séparations en deux cercles à 17 sommets, l'un en haut l'autre en bas. L'avant-dernière visualisation montre aussi les deux cercles et la dernière visualisation, qui fait penser à un schéma d'embryogenèse, reste difficile à interpréter avec ses 2 couples de jumeaux (il faut dire que 9930=2.3.5.331).

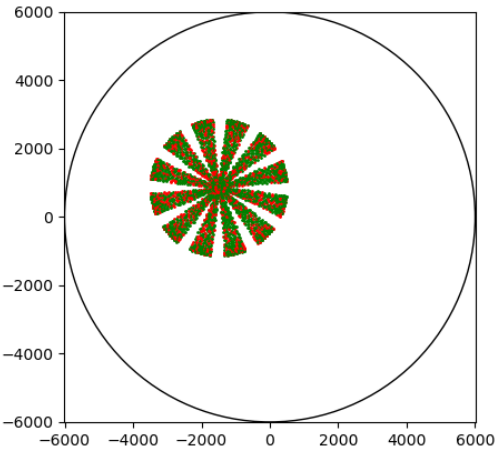
Ci-après, quelques visualisations supplémentaires qu'on pourrait appeler "fleurs d'enfants".



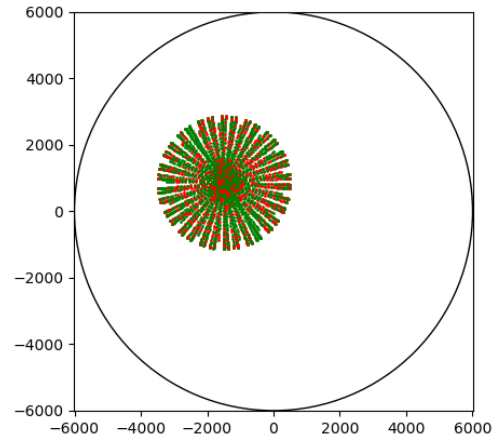
61, 59, haut et bas jusqu'à 2020



61, 59, $6x+1$ seuls, jusqu'à 20 020

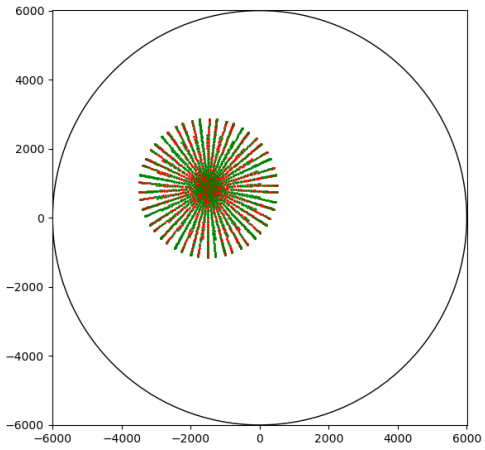


73, 71

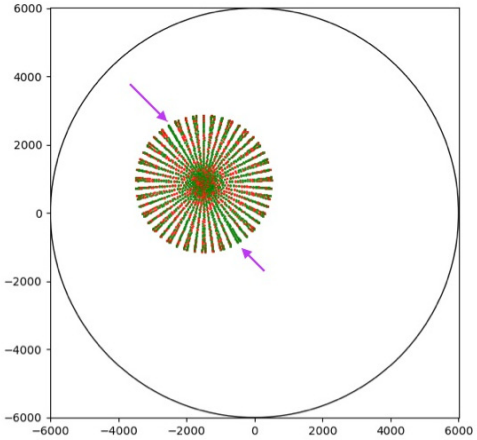


103, 101

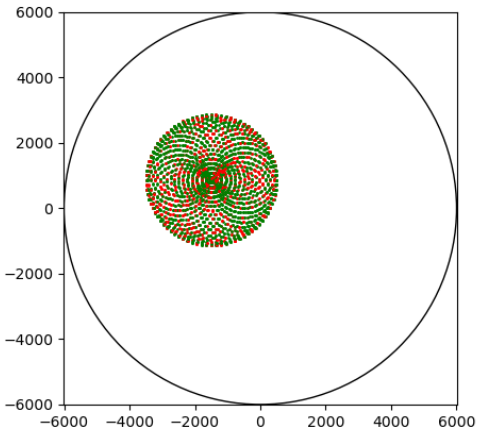
Dès le second dessin, on "oublie informatiquement" les $6x-1$ qui présente une structure symétrique de celle présentée par les $6x+1$, de l'autre côté (sous) l'axe des abscisses.



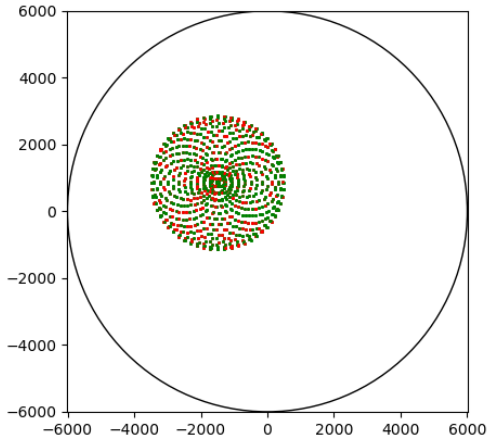
199, 197



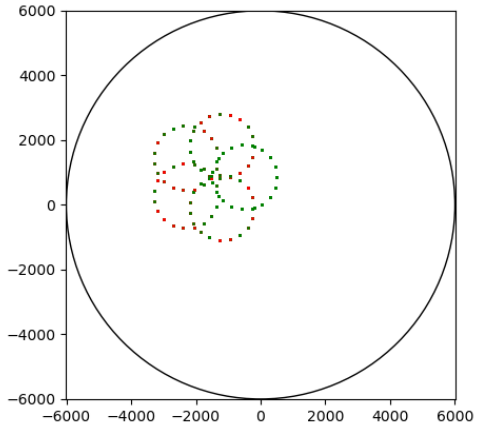
139, 137



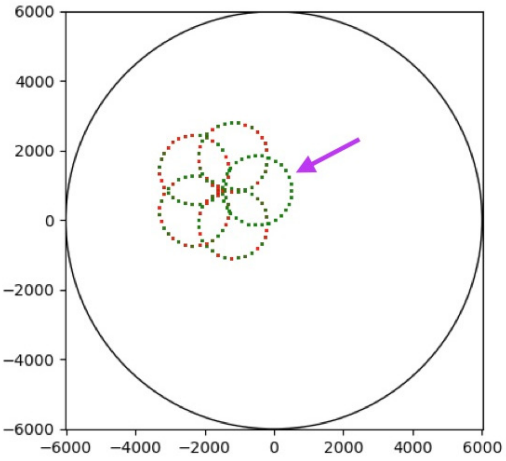
43, 41



31, 29



19 et 5, pas jumeaux



31 et 5, pas jumeaux

Les structures conditionnées par la factorisation de celui qu'on avait appelé un "père de premiers jumeaux" (nombre pair entre eux deux) apparaissent clairement dans la forme et le nombre des "pétales" (par exemple,

une succession rapprochée de deux pétales puis d'un, deux pétales longs, un pétale court, par exemple). Les pétales intégralement verts ou les cercles ne contenant que des points verts correspondent aux nombres divisibles par tel ou tel premier (on les a désignés par des flèches roses dans les $6^{\text{ème}}$ et dernière visualisations ci-dessus). Les "fleurs" (numérotés 1, 3, 4, 5, 6 sur 10 dessins) qui montrent des pétales droites depuis le centre correspondent à des pères de jumeaux divisibles par 4. Les autres dessins, montrant des cercles successifs ressemblant au coeur d'une fleur de tournesol, correspondent à un couple de jumeaux dont le père est un $4k + 2$ (n'est pas divisible par 4).

Bibliographie

[1] Alain Connes, "A new proof of Morley's theorem", *Publications Mathématiques de l'IHÉS*, **S88** : 43-46, 1998.

[2] Transcription d'une vidéo d'Alain Connes au Collège de France, visionnable ici :https://www.college-de-france.fr/site/colloque-2018/symposium-2018-10-18-10h00.htm, http://denisevellachemla.eu/transc-AC-langage.pdf.

[3] Denise Vella-Chemla, Snurpf, exemple, 2019,http://denisevellachemla.eu/snurpf-exemple.pdf, démonstration de la caractérisation :http://denisevellachemla.eu/demo-caracterisation-DG.pdf.