

Programmes ou idées de juillet 2023 (Denise Vella-Chemla, 25 juillet 2023)

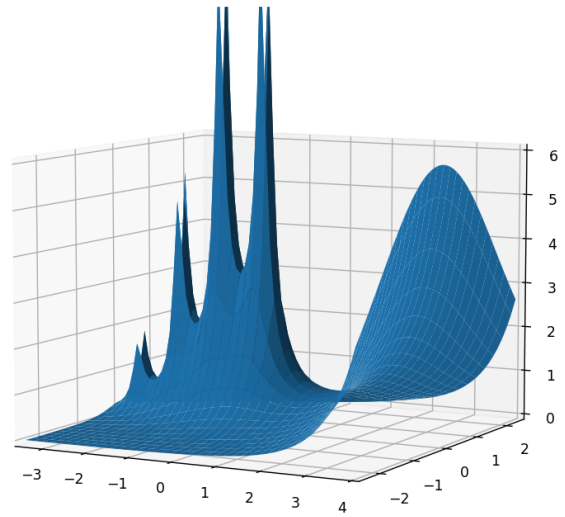
Gamma en python

```
youpi-gamma-python.py - GNU Emacs at DESKTOP-4PROGGL
File Edit Options Buffers Tools Python Help
[Icons]
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np

fig = plt.figure()
ax = Axes3D(fig)

N=70
theta = -np.pi + (2*np.pi/N) * (0.5 + np.arange(N))
c=-11
r=16
s=0.1
x = np.arange(-3.5, 4+s, s)
y = np.arange(-2.5, 2.5+s, s)
xx, yy = np.meshgrid(x, y)
zz = xx + 1j*yy
gaminv = 0*zz
for k in range(N):
    t = c + r*np.exp(1j*theta[k])
    gaminv += np.exp(t)*(t**(-zz))*(t-c)
gaminv = gaminv/N
gam = 1.0/gaminv
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(xx,yy, abs(gam))
ax.set_xlim(-3.5,4)
ax.set_ylim(-2.5,2.5)
ax.set_zlim(0,6)
plt.show()

-\\--- youpi-gamma-python.py All L1 (Python Ell
Wrote c:/Users/DENISE 2022/Desktop/youpi-gamma-python
```



Symétrie-miroir non vue jusque-là des nombres p q dont le produit vaut $1 \pmod{p}$, pour p premier, autour du milieu (les complémentaires de p et q que sont $n - p$ et $n - q$ ont aussi leur produit qui vaut $1 \pmod{p}$ et c'est chouette (même si c'est normal).

0.5 1.5 2.5 3.5

(7) 1 2 3 4 5 6

(11) 1 2 3 4 5 6 7 8 9 10

(19) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

(13) 1 2 3 4 5 6 7 8 9 10 11 12

$x \times y \equiv 1 \pmod{p}$

$\Leftrightarrow (p-x)(p-y) \equiv 1 \pmod{p}$

$p^2 - (x+y)p + xy \equiv 1 \pmod{p}$

$\left(\frac{p}{2} - \frac{1}{2}(2k+1)\right) \times \left(\frac{p}{2} + \frac{1}{2}(2k+1)\right)$ on voit les monnaies comme à gauche ou à droite de la moitié du module

$p \text{ premier} \Leftrightarrow p! \equiv -1 \pmod{p}$ ou Le miroir du Théorème de Wilson pour les p's

$3 \times 5 \equiv 1 \pmod{7}$

$\left(\frac{7}{2} - \frac{1}{2} \times (2 \times 0 + 1)\right) \times \left(\frac{7}{2} + \frac{1}{2} (2 \times 1 + 1)\right) \equiv 1 \pmod{7}$

$\frac{49}{4} - \frac{7}{4}(2 \times 0 + 1) + \frac{7}{4}(2 \times 1 + 1) - \frac{1}{4}(1 \times 3)$

$\frac{49}{4} - \frac{7}{4} \times (1 + 3) - \frac{3}{4}$

$\frac{49}{4} - 7 - \frac{3}{4}$

$\frac{46}{4} - \frac{28}{4} = \frac{18}{4} \equiv 1 \pmod{7}$

$\Leftrightarrow 18 \equiv 4 \pmod{7}$

Enfin ma surface bosselée avec les décompositions de Goldbach qui “tombent au sol” ($z = 0$) en python, avec en deuxième graphique : Plaid Goldbach vu de dessus (on distingue en foncé 3+3, 3+5, 3+7), et en troisième graphique : les nombres premiers de 3 à 19 vus en coupe de profil de la surface bosselée.

```
Jacques-interpole-ma-surface.py - GNU Emacs at DESKTOP-4PROGGL
File Edit Options Buffers Tools Python Help
[Icons]
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import math
import numpy as np
from scipy.interpolate import RBFInterpolator

fig = plt.figure()
ax = Axes3D(fig)

def sd(n):
    return(np.sum([np.sum([np.cos(2*np.pi*n*l/k) for l in range(1,k+1)]) for k in
range(1,n+1)]))
print(sd(100))

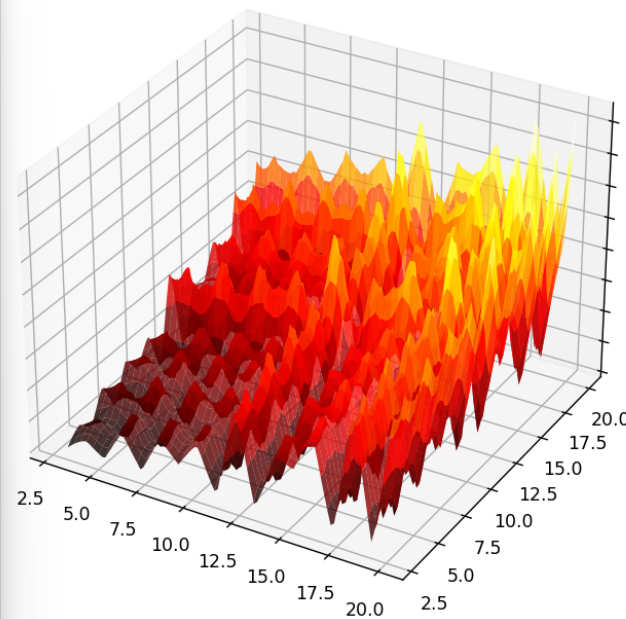
n = 20 # taille de la grille d'entiers
m = 5*n # taille de sa discretisation

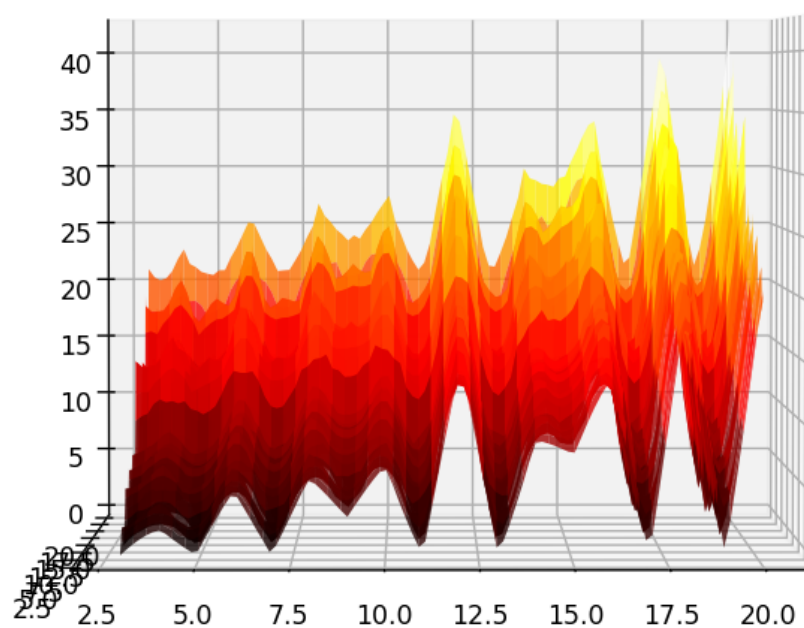
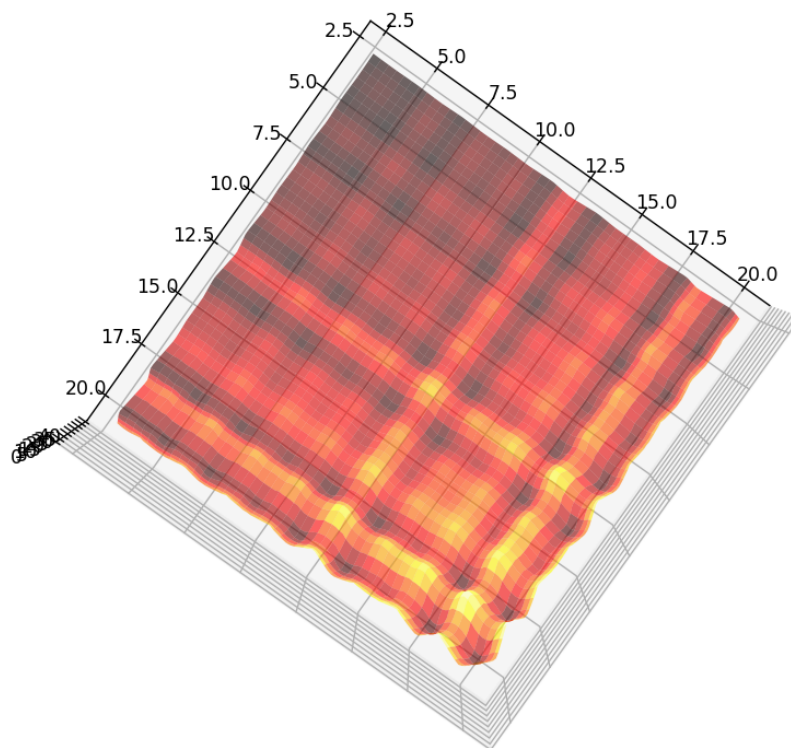
XY = np.array([[i, j] for j in range(3, n+1) for i in range(3, n+1)])
Z = np.array([sd(i) + sd(j)-i-j-2 for j in range(3, n+1) for i in range(3, n+1)])
print(Z)

Xd = Yd = np.linspace(3, n, num=m)
Xgrid, Ygrid = np.meshgrid(Xd, Yd)
XYd = np.array(list(zip(Xgrid.flat, Ygrid.flat)))
#print(f'\nXgrid =\n{Xgrid}\nYgrid =\n{Ygrid}\nXYd =\n{XYd}')

Zd = RBFInterpolator(XY, Z, kernel='linear', epsilon=7, neighbors=8)(XYd)
Zgrid = Zd.reshape(m, m)
#print(f'\nZd =\n{Zd}\nZgrid =\n{Zgrid}')

ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(Xgrid, Ygrid, Zgrid, cmap=plt.cm.hot, linewidth=0, antialiased=True, alpha=0.8)
#ax.plot_wireframe(Xgrid, Ygrid, Zgrid, alpha=0.3)
#ax = fig.add_subplot(111)
#CS = ax.contour(Xgrid, Ygrid, Zgrid)
#ax.clabel(CS, inline=True, fontsize=10)
plt.show()
```





Vieux souvenirs : programme d'été d'une courbe presque aussi jolie que celle de Hilbert (Mandelbrot la dénomme Minkowski sausage, en l'honneur de son ami mort jeune). Je la programme d'abord "langagièrement" : A est pour Avance (en logo de Marvin Minsky ou en Scratch du MIT), G pour "Tourne à gauche", D pour "Tourne à droite" (les changements de direction s'effectuent en restant sur place). Pour passer d'un niveau n au niveau supplémentaire $n + 1$, on itère 3 fois le programme de niveau n avec une toute petite modification : on change la dernière lettre de la deuxième itération de D en G, c'est tout, les appels récursifs font le reste (j'ai programmé rapidement, il faudrait améliorer le programme, ça c'est sûr).



```

courbe-croix-grecque-2.py - GNU Emacs at DESKTOP-4PROGGL
File Edit Options Buffers Tools Python Help

str2='AGADADAGAGADAGADAD'
longueur = 25
setheading(90)
up()
x = x+taille
turtle.setposition(x,y)
down()
dessine(str2)

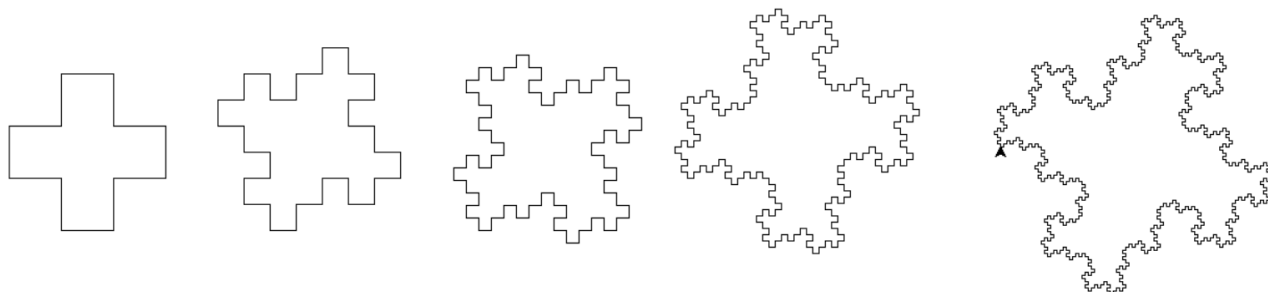
str3='AGADADAGAGADAGADADAGADADAGADADAGADADAGADADAGADADAGADAD'
setheading(90)
up()
longueur = 12
x = x+taille
turtle.setposition(x,y)
down()
dessine(str3)

str4 = str3+str3
str4b = str4[0:2*len(str3)-1]
str4b += 'G'+str3
setheading(0)
up()
longueur = 6
x = x+1.3*taille
y = y+taille
turtle.setposition(x,y)
down()
dessine(str4b)

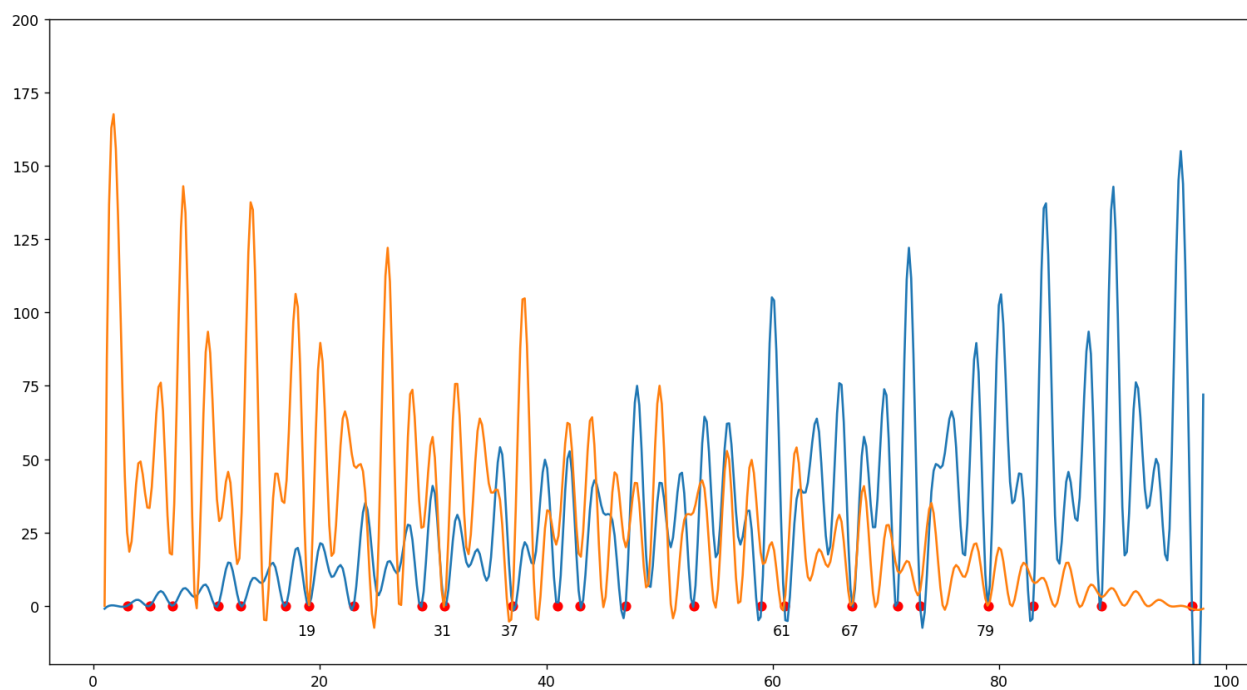
str5 = str4b+str4b
str5b = str5[0:2*len(str4b)-1]
str5b += 'G'+str4b
setheading(90)
up()
longueur = 3
x = x+1.2*taille
y = y-0.6*taille
turtle.setposition(x,y)
down()
dessine(str5b)
exitonclick()

--\--- courbe-croix-grecque-2.py Bot L42 (Python ElDoc)

```



Une idée qui a fait son temps : on interpole la somme de cos (qui s'annule pour les premiers) par une (smoothie !) cubique, on prend sa symétrique par rapport au milieu et on voit les points nuls communs qui sont les décomposants de Goldbach.





```

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import math
import numpy as np
from scipy.interpolate import interp1d

fig, ax = plt.subplots(figsize=(15, 10))

def sd(n):
    # somme des diviseurs de n >= 1
    return sum([sum([math.cos(2*math.pi*n*l/k) for l in range(1, k+1)]) for k in
    range(1, n+1)])
#for n in 1, 2, 3, 4, 5, 6, 12, 100: print(f'sd({n:3}) = {sd(n):6.2f}')

n = 98 # taille de la grille d'entiers
m = 5*n # taille de sa discretisation

X = np.array([i for i in range(1, n+1)])
Z = np.array([sd(i) - i - 1 for i in range(1, n+1)])
#print(f'\nX =\n{X}\nZ =\n{Z}')
Zp = np.array([sd(n-i) - n + i - 1 for i in range(1, n+1)])

Xd = np.linspace(1, n, num=m)
#print(f'\nXd =\n{Xd}')

Zd = interp1d(X, Z, kind='cubic')(Xd)
Zdp = interp1d(X, Zp, kind='cubic')(Xd)
#print(f'\nZd =\n{Zd}')

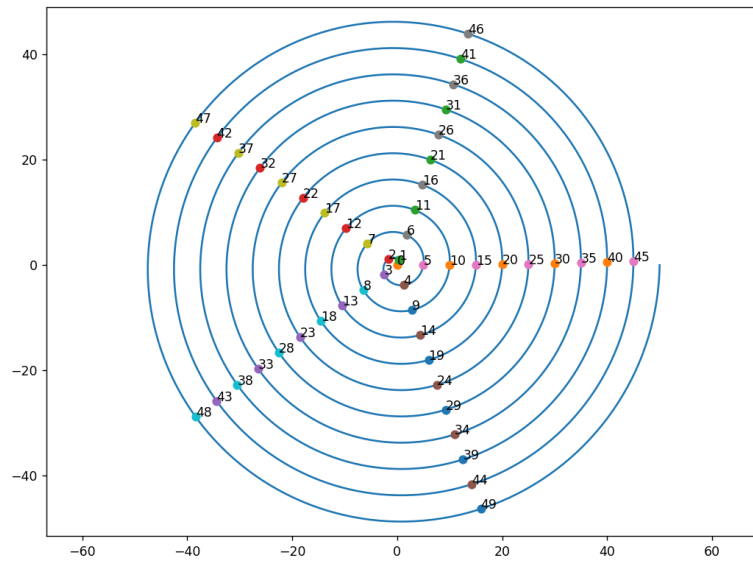
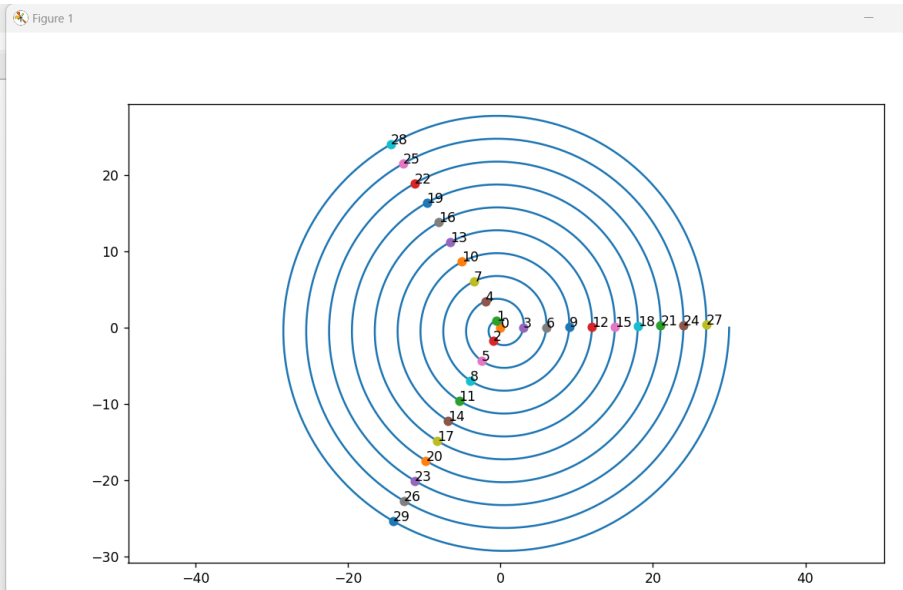
ax.plot(Xd, Zd)
ax.plot(Xd, Zdp)
ax.set_ylim(-20,200)
#ax.scatter(X, Z, c='r', marker='o')
ax.scatter([3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0], c='r', marker='o')
plt.text(18,-10,'19')
plt.text(30,-10,'31')
plt.text(36,-10,'37')
plt.text(78,-10,'79')
plt.text(66,-10,'67')
plt.text(60,-10,'61')
plt.show()

```

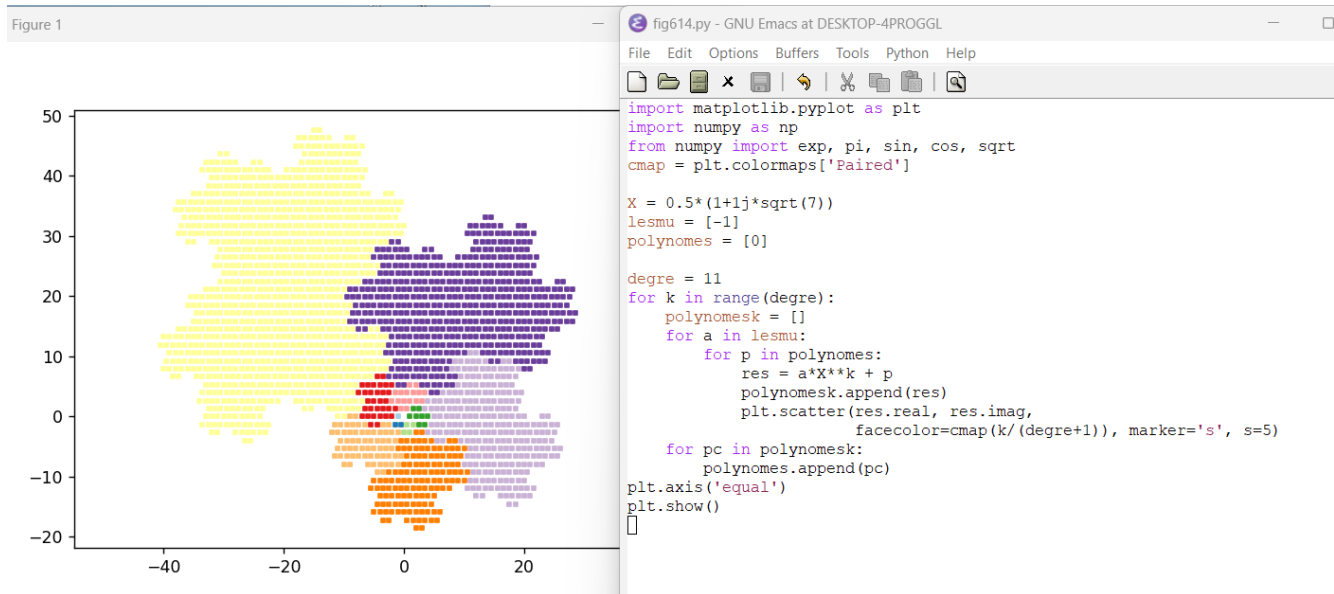
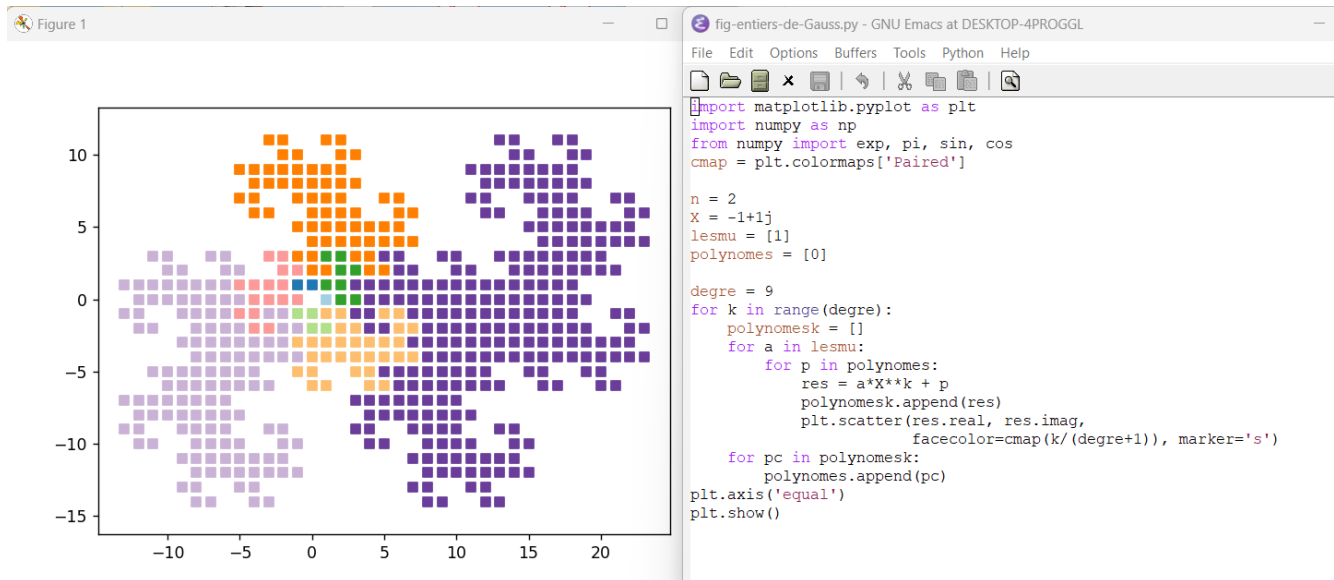
Spirale d'Archimède sur laquelle on positionne les nombres entiers successifs soit sur l'axe des abscisses s'ils sont multiples d'un nombre entier, soit sur les angles correspondant aux racines de l'unité suivant qu'ils sont congrus à $1, \dots, k-1$ selon le nombre k choisi (on montre le résultat pour $k=3$, ou $k=5$).

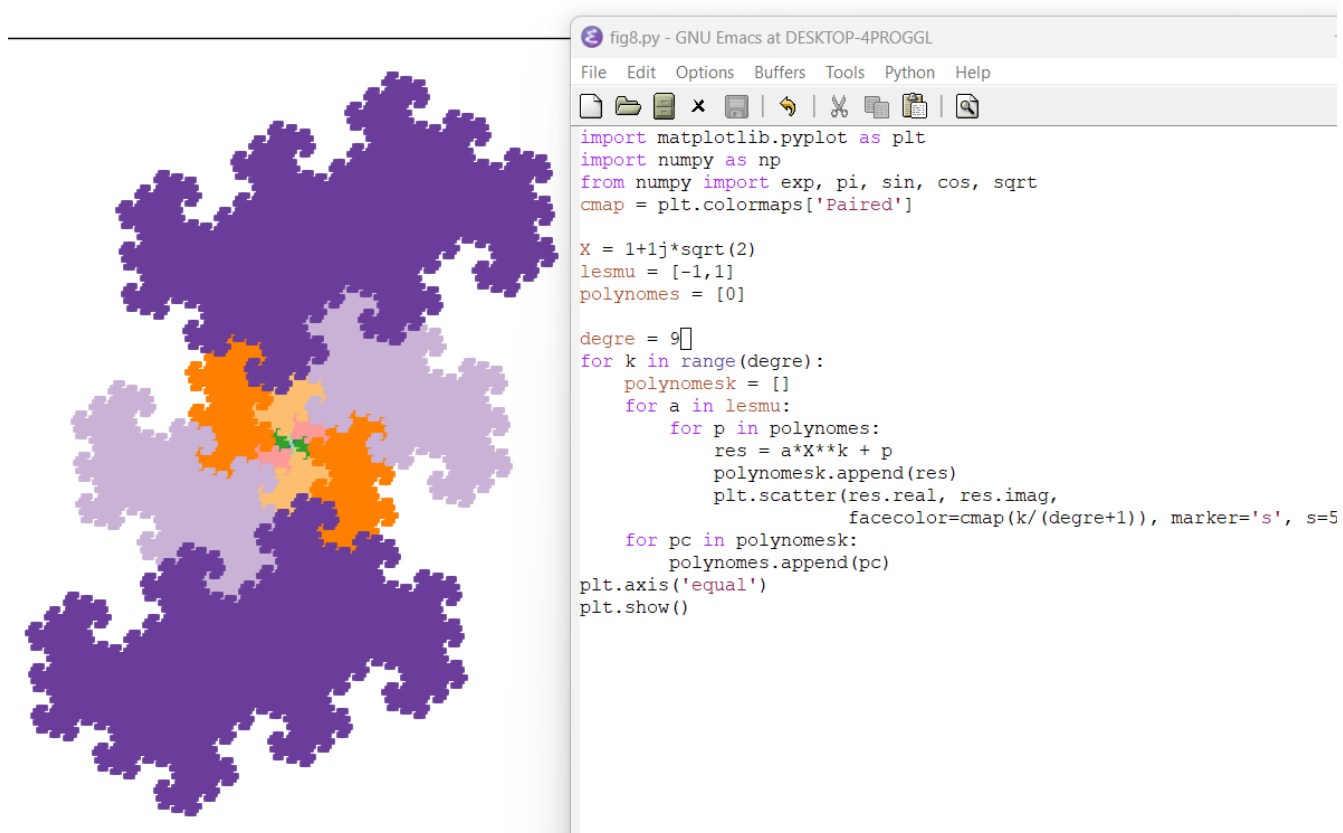
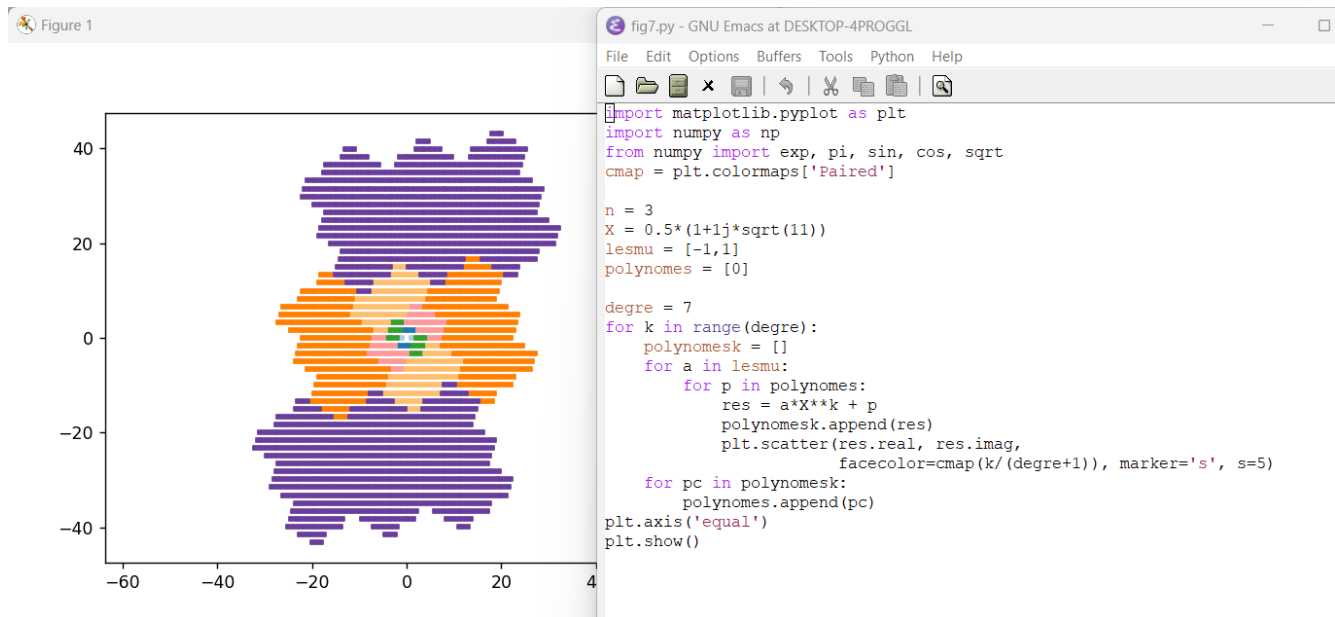
```
pc2.py - GNU Emacs at DESKTOP-4PROGGL
File Edit Options Buffers Tools Python Help
import matplotlib.pyplot as plt
import numpy as np
from numpy import pi, sin, cos

a = 3/(2*pi)
nbtours = 10
print(np.pi)
theta = np.linspace(0, nbtours*2*pi, nbtours*360)
rho = a*theta
x = rho*cos(theta)
y = rho*sin(theta)
plt.plot(x,y)
for nb in range(10):
    ici = nb*360
    icix = rho[ici]*cos(theta[ici])
    icy = rho[ici]*sin(theta[ici])
    plt.plot([icix, icy], 'o')
    plt.annotate(3*nb, xy=(icix, icy))
    ici = ici+120
    icix = rho[ici]*cos(theta[ici])
    icy = rho[ici]*sin(theta[ici])
    plt.plot([icix, icy], 'o')
    plt.annotate(3*nb+1, xy=(icix, icy))
    ici = ici+120
    icix = rho[ici]*cos(theta[ici])
    icy = rho[ici]*sin(theta[ici])
    plt.plot([icix, icy], 'o')
    plt.annotate(3*nb+2, xy=(icix, icy))
plt.axis('equal')
plt.show()
-\\--- pc2.py All L1 (Python E1Doc)
```



Fractales d'anneaux de polynômes : il s'agissait de programmer les exemples de l'article de Connes-Consani de juillet 2023 “Sur la métaphysique de $\mathbb{F}_1$ ”.





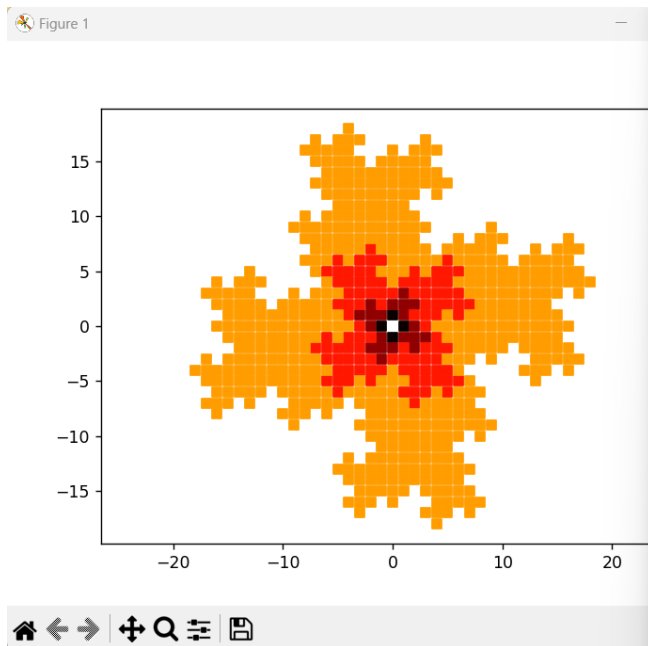
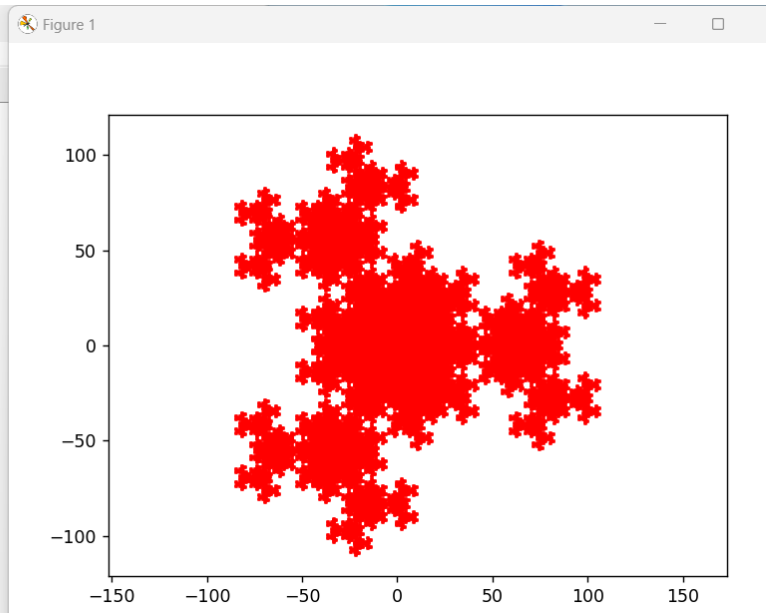
```

triflake.py - GNU Emacs at DESKTOP-4PROGGL
File Edit Options Buffers Tools Python Help
import matplotlib.pyplot as plt
import numpy as np
from numpy import exp, pi, sin, cos
cmap = plt.colormaps['Paired']

n = 3
X = -2
lesmu = [exp(1j*2*pi*k/n) for k in range(n)]
polynomes = [0]

degree = 7
for k in range(degree):
    polynomesk = []
    for a in lesmu:
        for p in polynomes:
            res = a*X**k + p
            polynomesk.append(res)
            plt.scatter(res.real, res.imag,
                        color='red', marker='s', s=2)
        for pc in polynomesk:
            polynomes.append(pc)
plt.axis('equal')
plt.show()

```



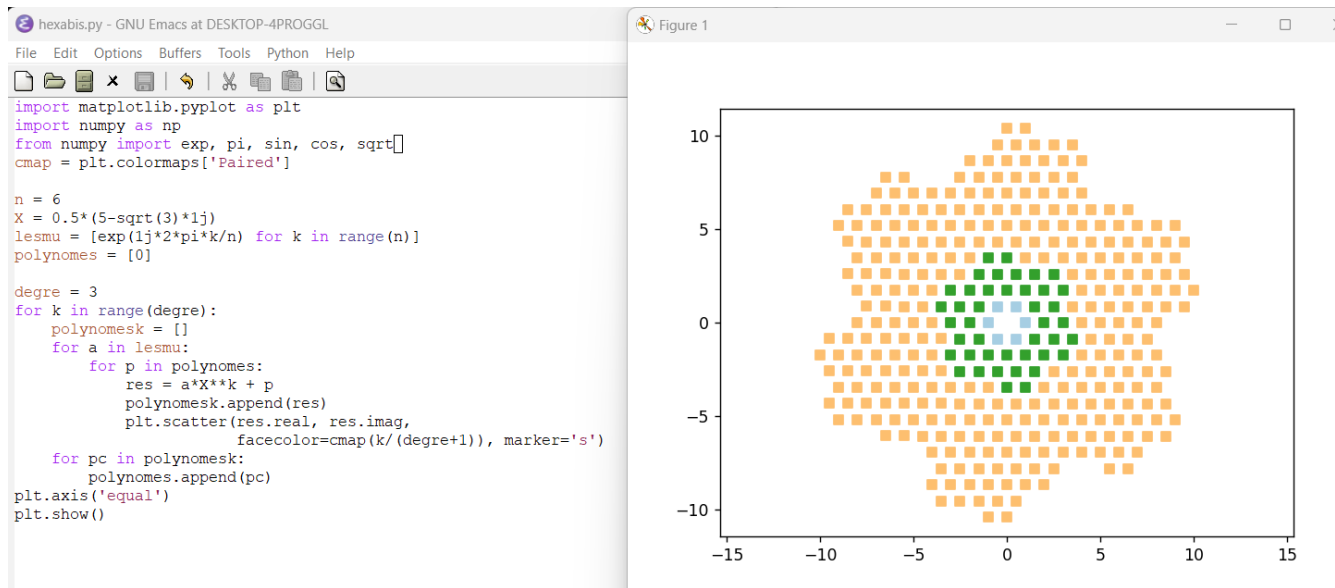
```

unfractaldes.py - GNU Emacs at DESKTOP-4PROGGL
File Edit Options Buffers Tools Python Help
import matplotlib.pyplot as plt
import numpy as np
from numpy import exp, pi, sin, cos
cmap = plt.colormaps['hot']

n = 4
X = 1+2j
lesmu = [exp(1j*2*pi*k/n) for k in range(n)]
polynomes = [0]

degree = 4
for k in range(degree):
    polynomesk = []
    for a in lesmu:
        for p in polynomes:
            res = a*X**k + p
            polynomesk.append(res)
            plt.scatter(res.real, res.imag,
                        facecolor=cmap(k/(degree+1)), marker='s')
        for pc in polynomesk:
            polynomes.append(pc)
plt.axis('equal')
plt.show()

```



Mon pentaflake plein de trous, sans bijection donc, mais j'aime bien !

