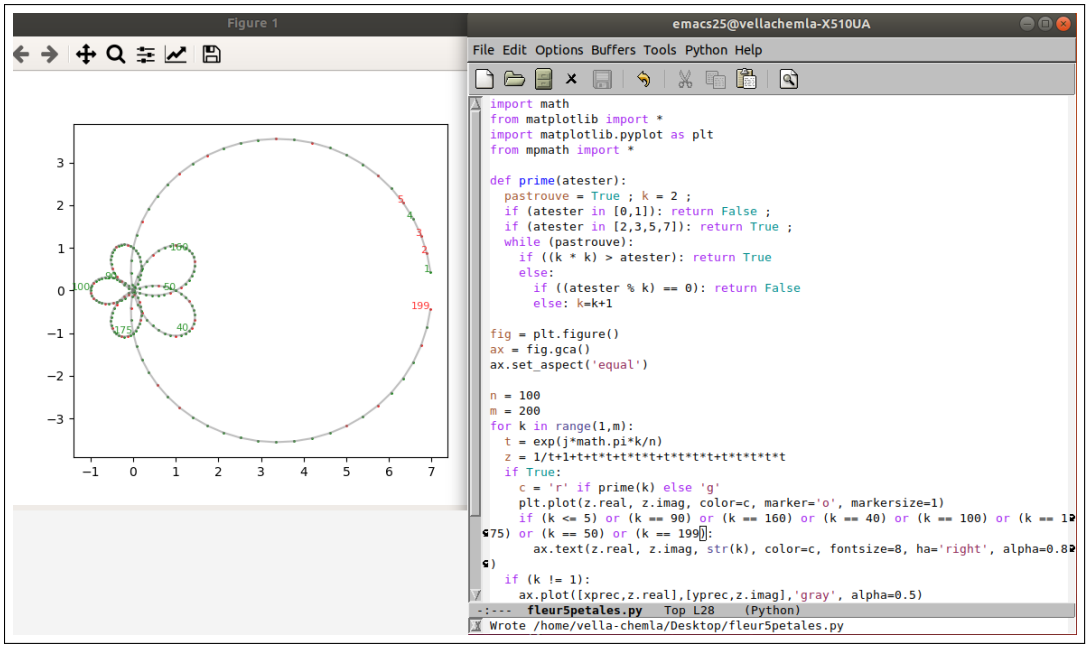
Petite étape (Denise Vella-Chemla, 10.10.2020)

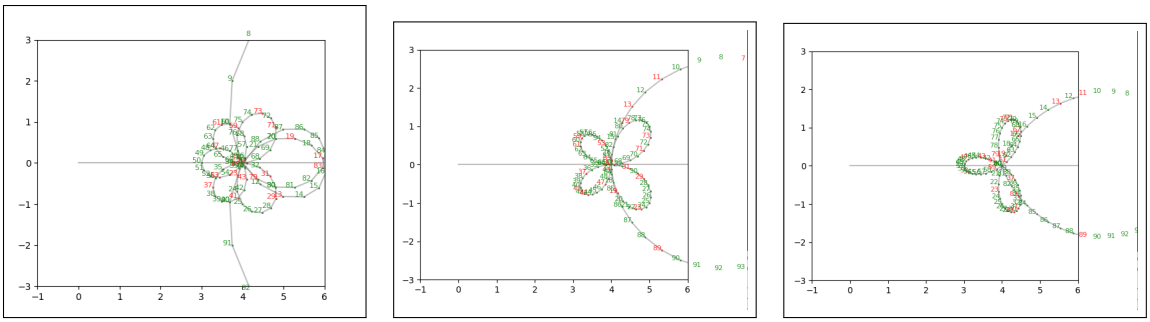
On fait toujours varier le même programme fourni dans une note précédente, pour essayer de comprendre pourquoi les images des entiers par nos fonctions sont ce qu'elles sont : on a trois paramètres sur lesquels on joue pour faire varier les graphiques :

- k, qui varie de 1 à 100, avec quelques variantes ;
- t, qui est en général de la forme $e^{i\pi k/n}$;
- z, le complexe dessiné, dont on "suit le parcours" (on relie les images successives, pour dessiner un lacet), et qui est en général, avec quelques variantes, égal à $\frac{1-t^m}{1-t} = \sum_{k=0}^{k=m-1} t^k$.

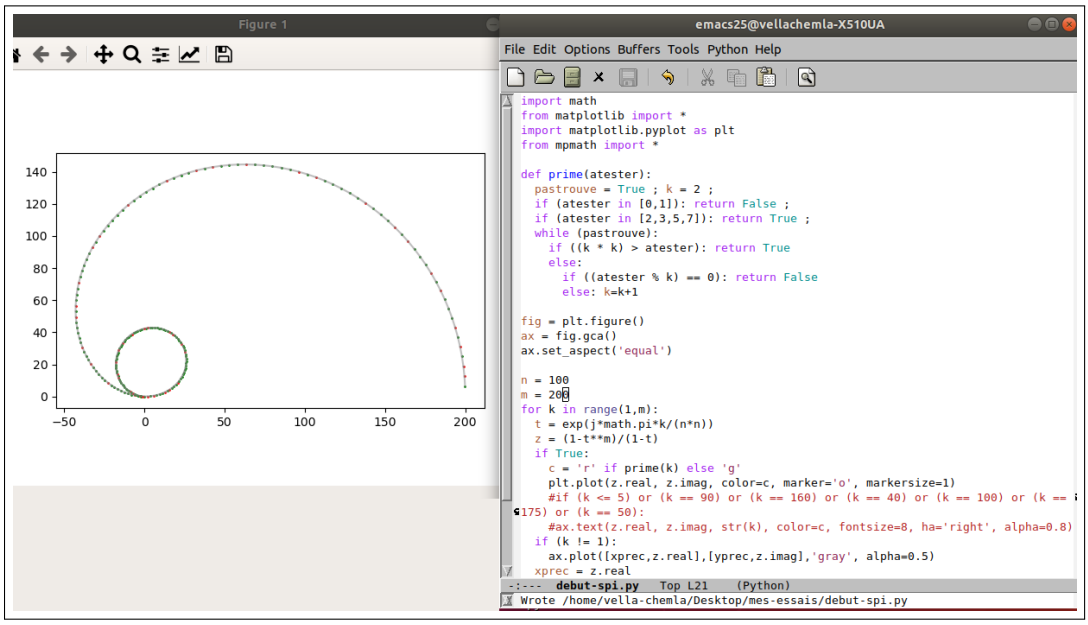
Voyons d'abord le bracelet avec fleur : on l'obtient en prenant la somme de $1/t$ (qu'on dit "à gauche") à t^5 (à droite).



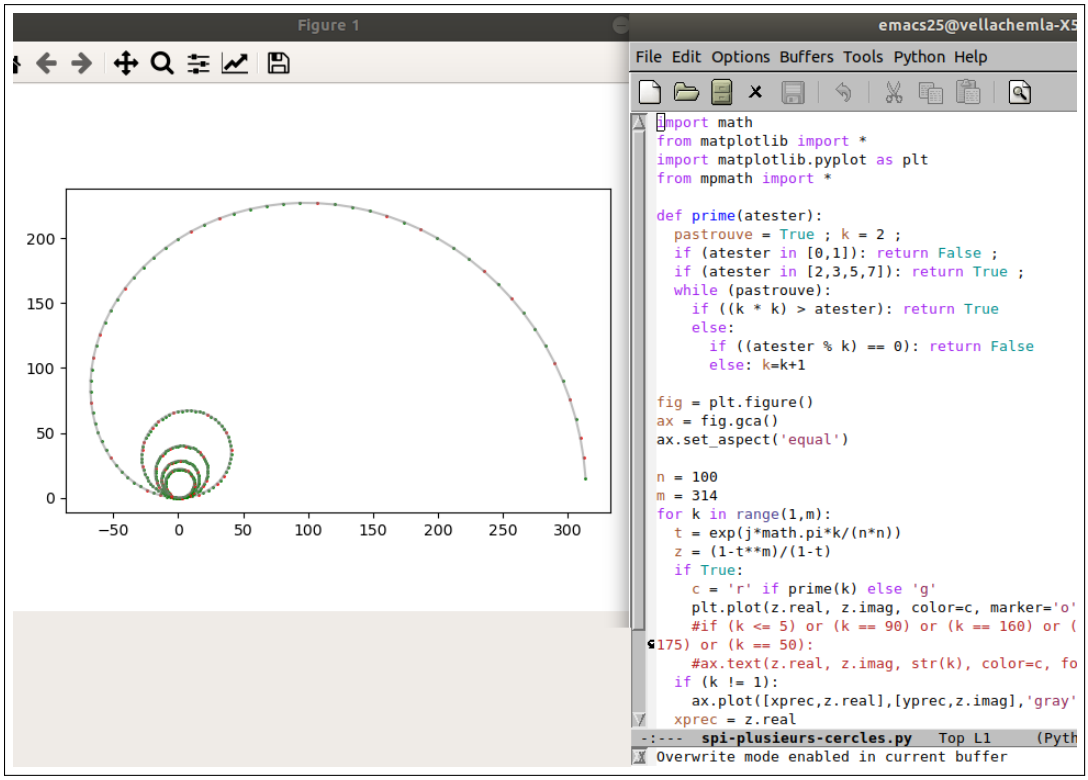
On peut jouer sur le nombre de pétales en changeant la puissance la plus grande.



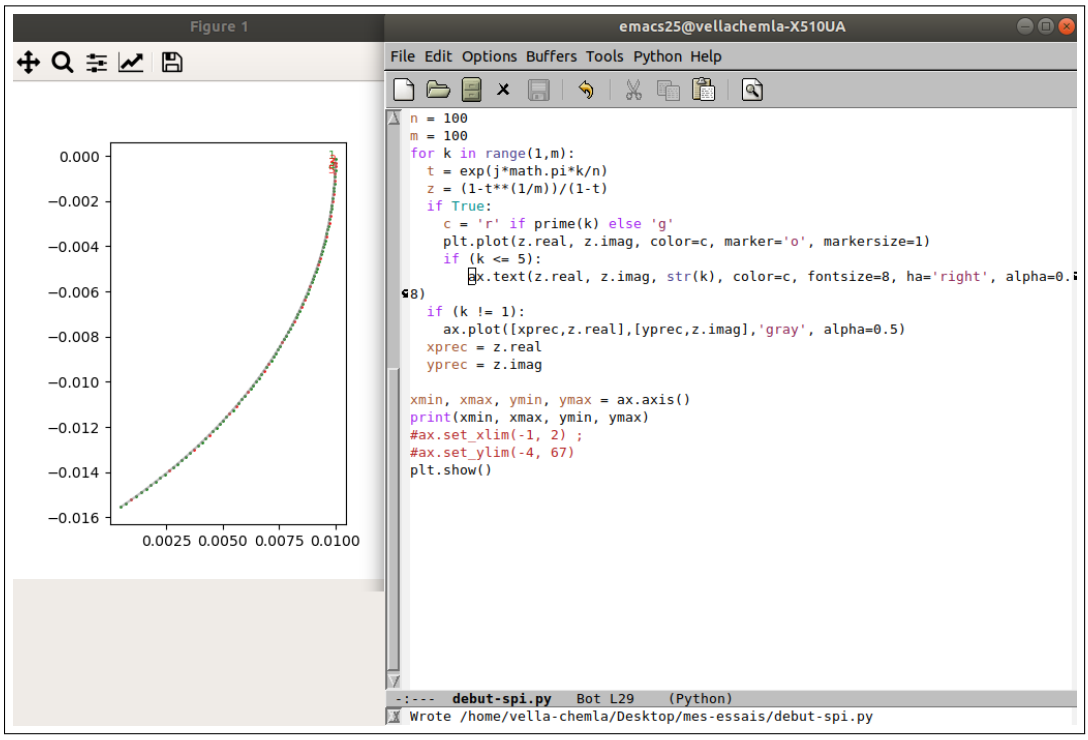
On veut comprendre pourquoi la fleur est au milieu du bracelet, pourquoi le bracelet n'est pas lui-même une seule grosse fleur, pourquoi il y a le haut du cercle, le bas du cercle, et la fleur au milieu à gauche. Alors, on réussit à obtenir un début de spirale, voire même un premier cercle en bas.



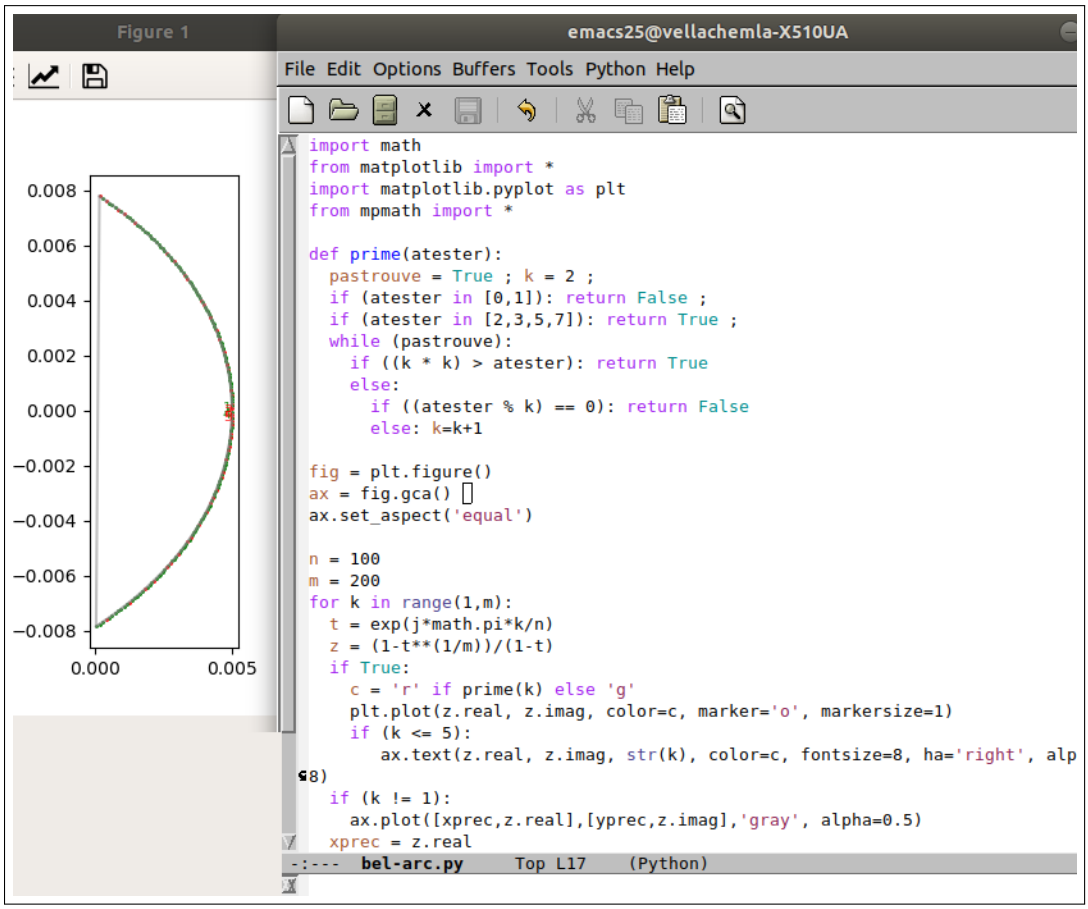
Puis on obtient en allant plus loin des cercles supplémentaires.



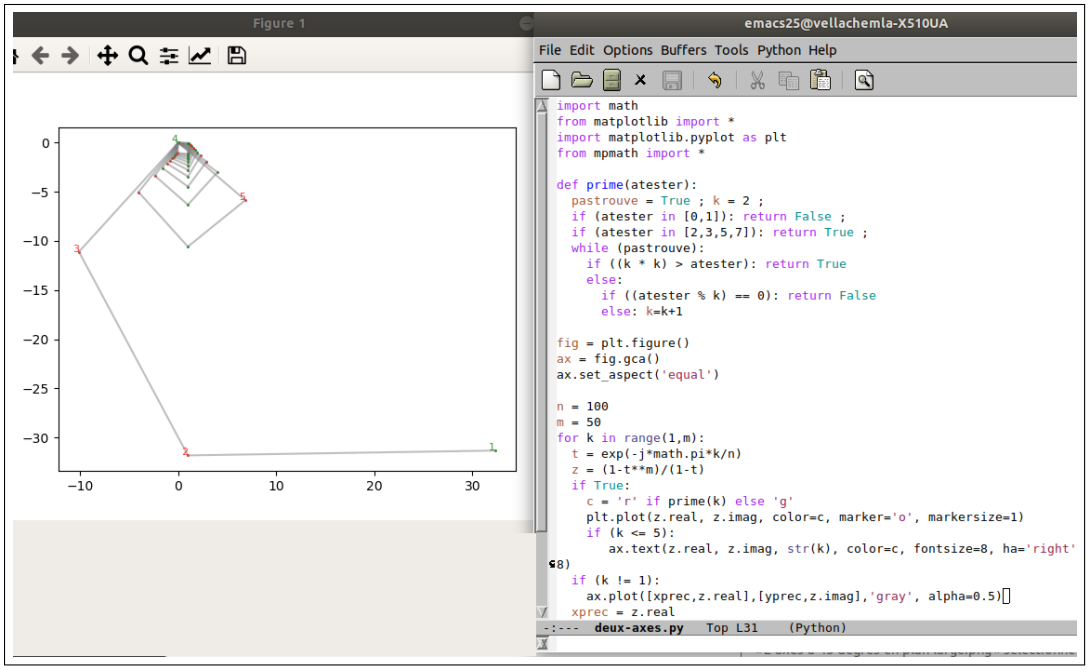
De façon surprenante, quand on met l'inverse de l'exposant, dans le calcul de la somme, la spirale part "à l'envers" (les images des nombres de 1 à 5 sont étiquetées en haut à droite du graphique).



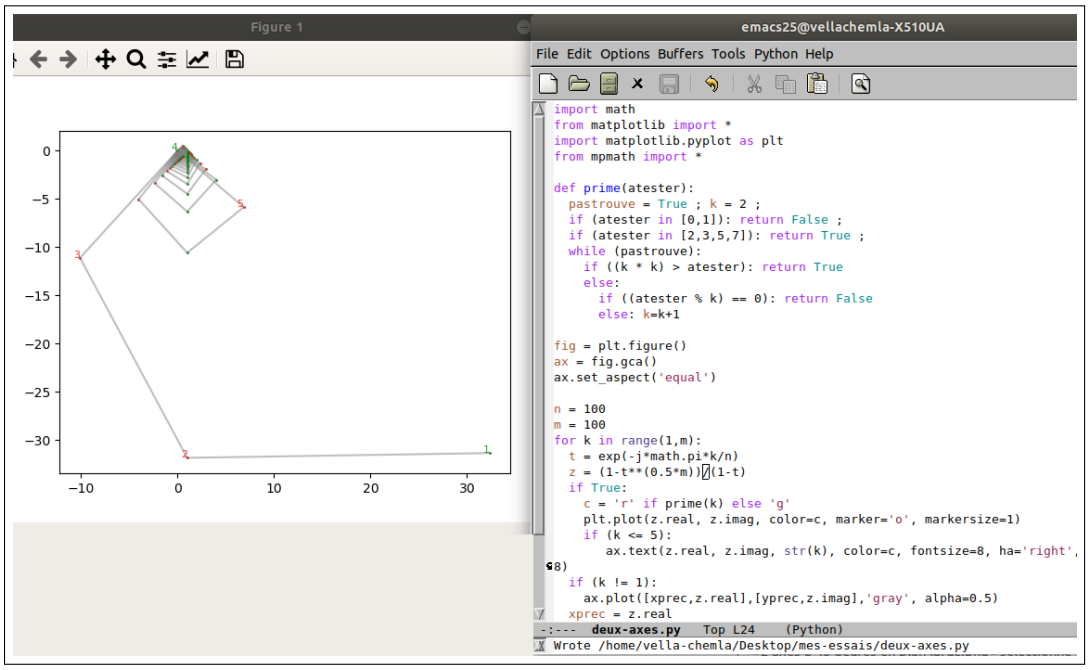
On pense “ok, toi, je vais te faire faire tout le tour”, et là, gros couac : ça dessine un arc, la corde de l’arc est dessinée entre 101 et 102, le fait d’avoir inversé l’exposant a bien sûr (sic !) rendu les valeurs minuscules.

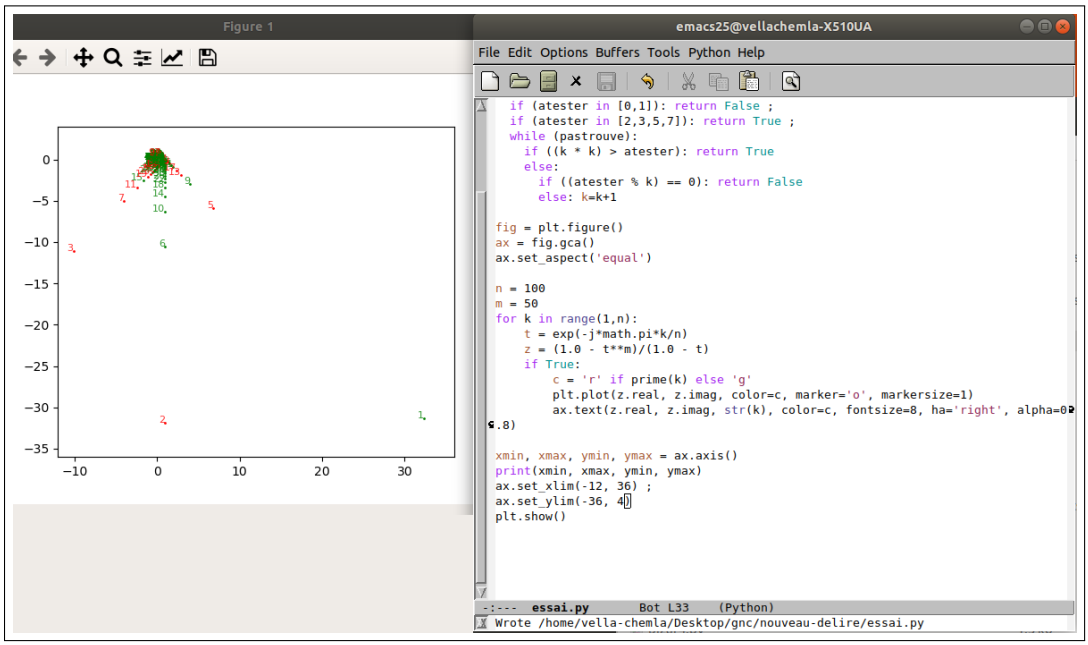


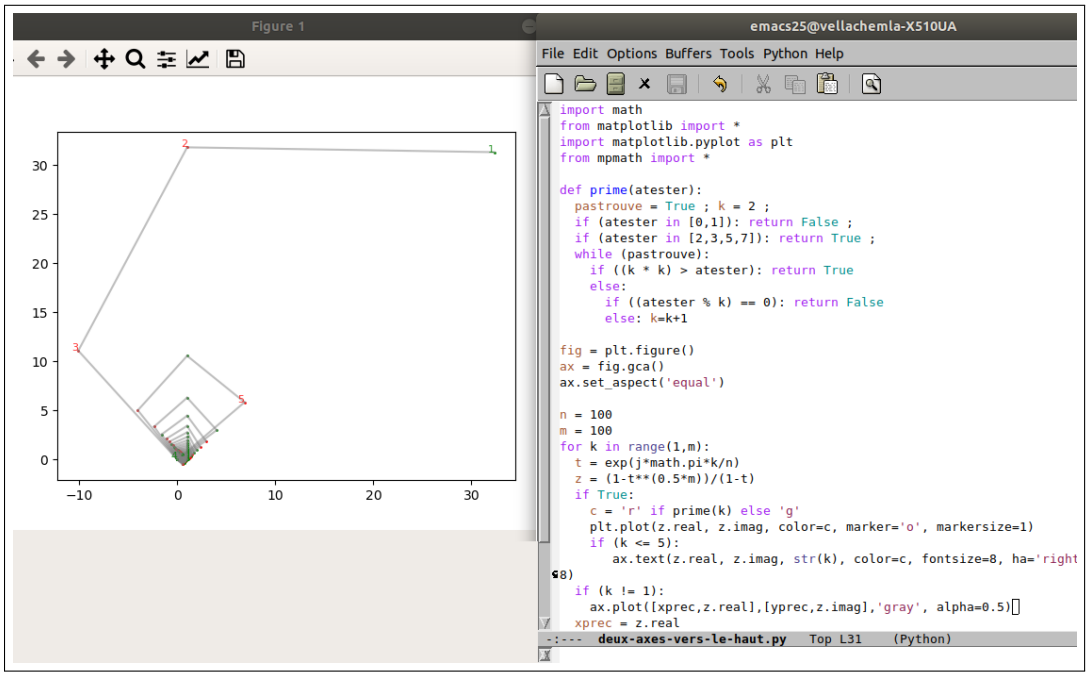
On a réussi par ci, par là, à mettre les nombres des formes $4x+1$ et $4x+3$ sur deux axes perpendiculaires à 45 degrés.



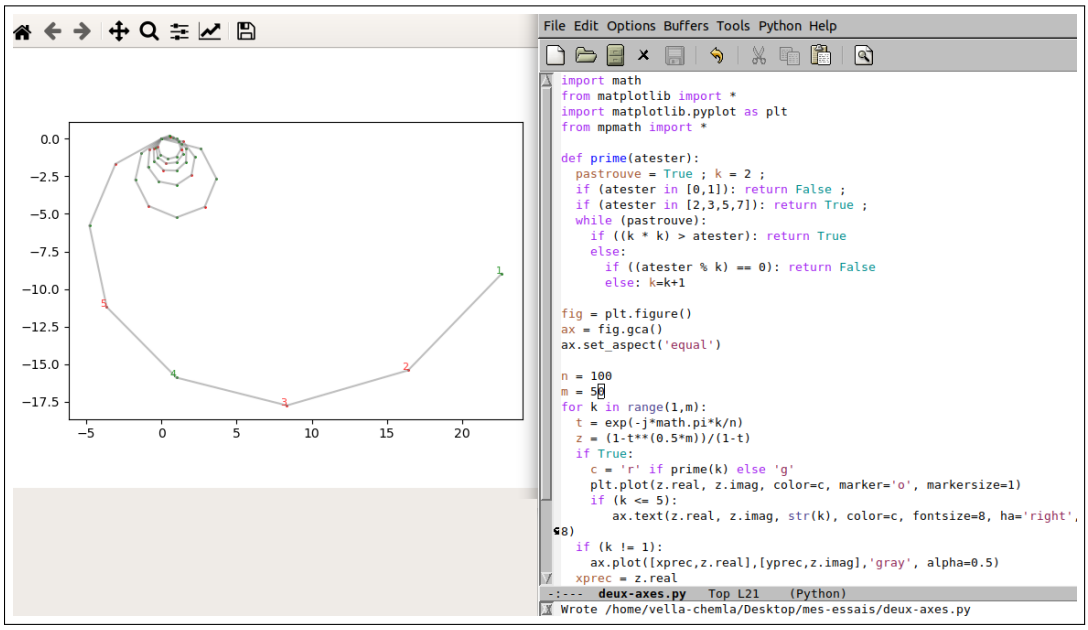
En modifiant les paramètres, on peut faire que les nombres soient “entrants” ou “sortants” sur les axes, qu’ils parcourent les axes vers le bas, ou vers le haut, de multiples variations sont bien sûr possibles et contrôlables. Sur les deux dessins suivants, on obtient un même résultat en utilisant deux programmes différents : dans le premier cas, on prend comme dernier terme de la somme le 50-ième tandis que dans le second cas, on va bien jusqu’au 100-ième terme, mais en multipliant l’exposant par 1/2 au numérateur dans le calcul de la somme.



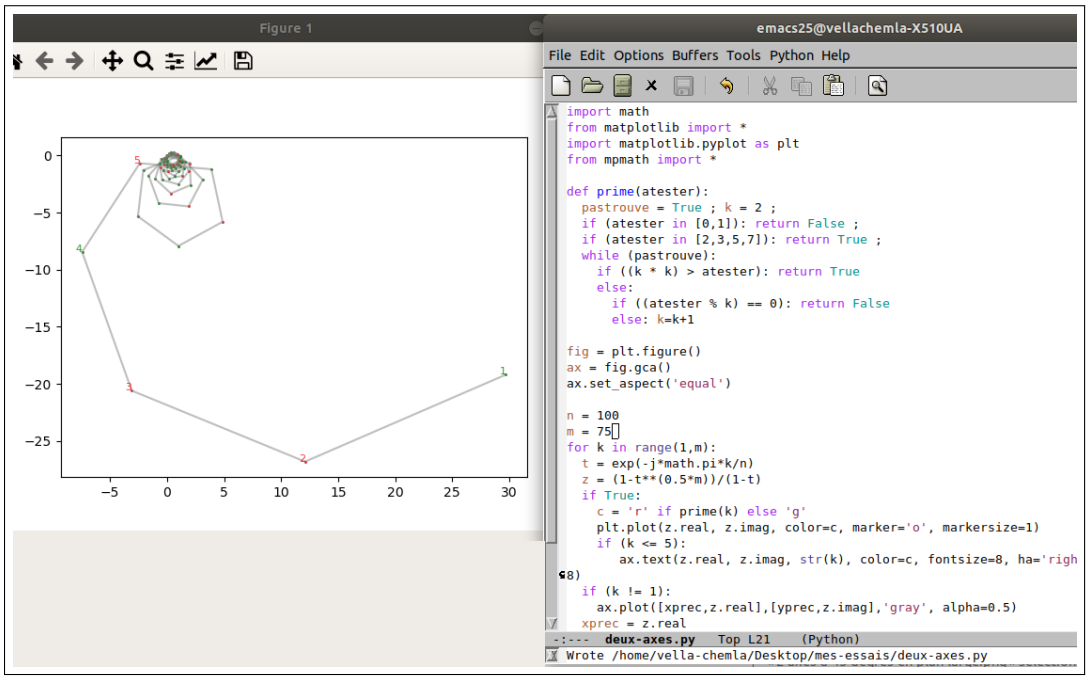




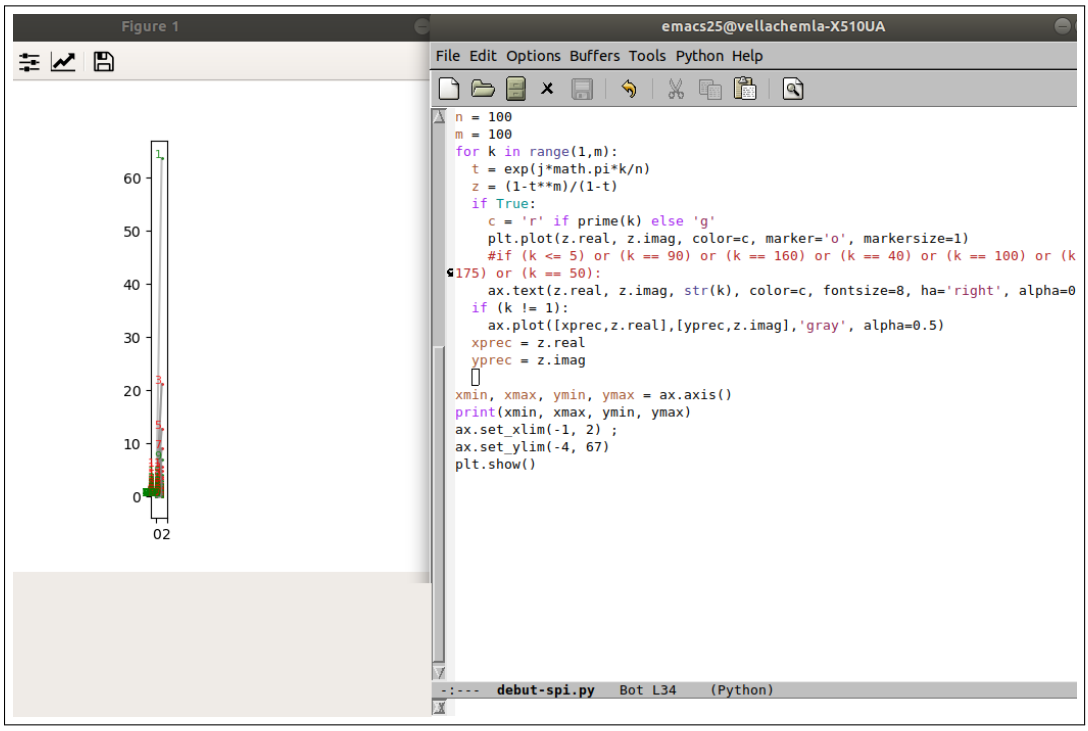
On peut multiplier les axes sur lesquels les nombres sont alignés. Ci-dessous, on distingue 6 axes, on a fait 3 variations : on n'est allé que jusqu'à $m = n/2$, on a pris la puissance opposée de e dans le calcul de t, on a pris la moitié de la puissance au numérateur dans le calcul de la somme...



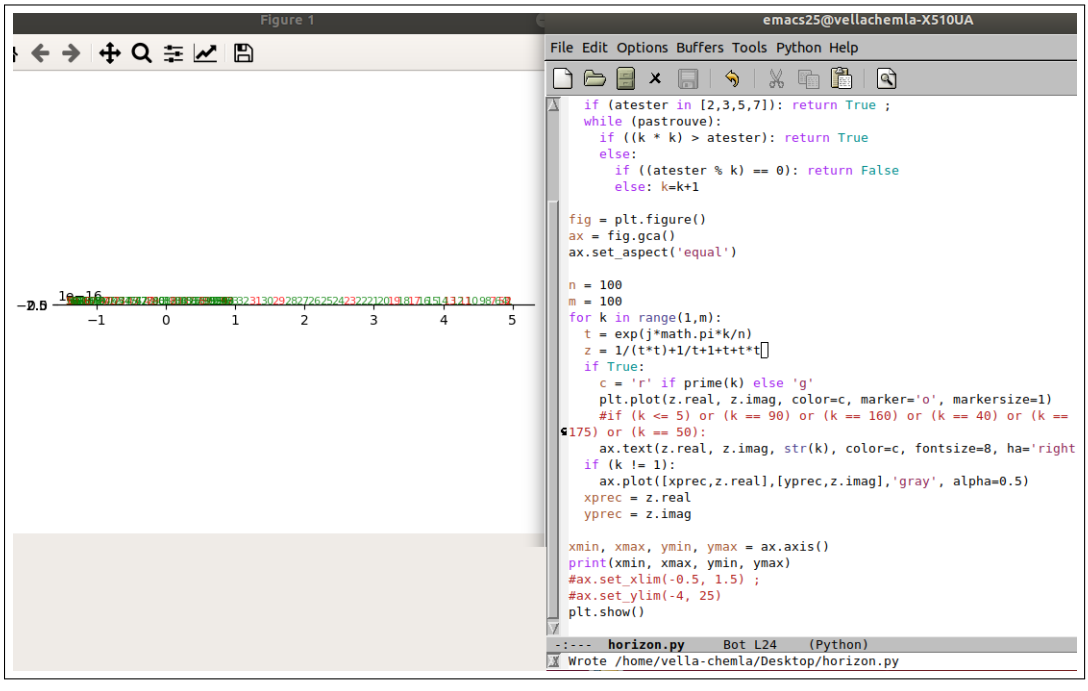
Il suffit parfois d'un rien pour décaler un graphique, lui conservant sa forme mais le "tordant" légèrement (là, le dernier sommant est le 75-ième).



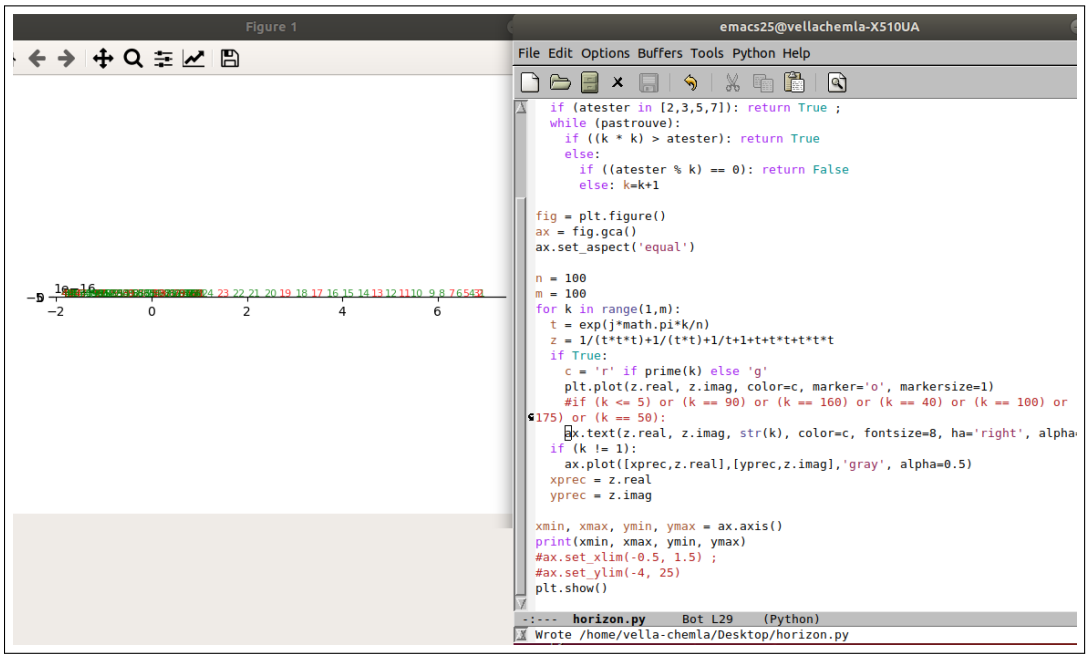
Le but reste l'alignement vertical, qu'on obtient en prenant $m = n$.



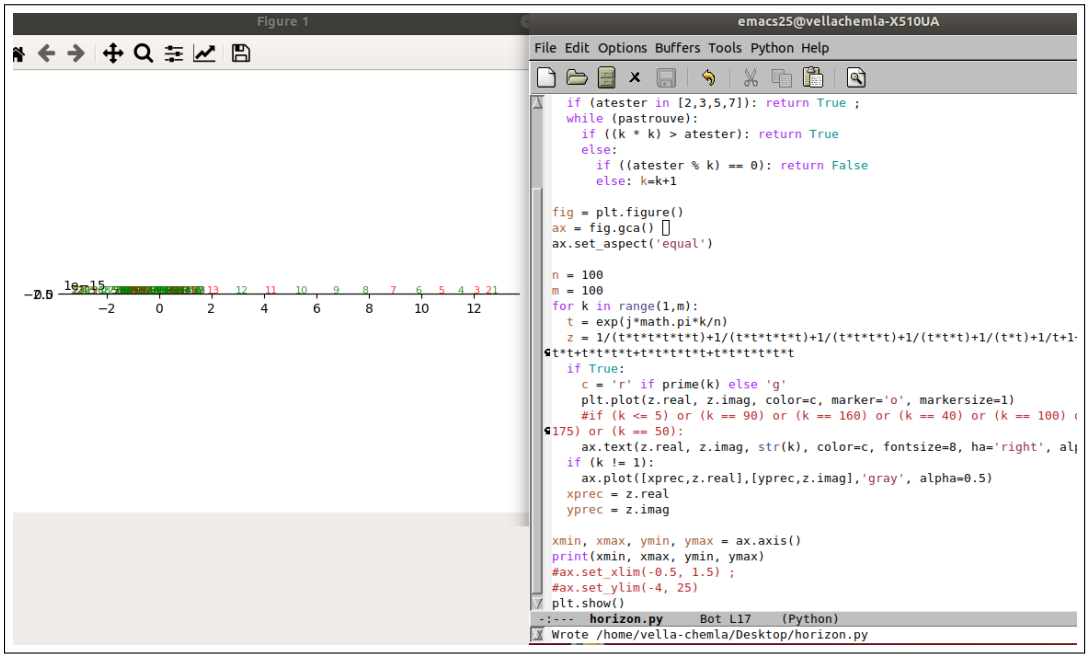
On obtient des alignements horizontaux en faisant varier les termes de la somme depuis l'inverse d'une certaine puissance (qu'on dira "à gauche"), jusqu'à la puissance en question (qu'on dira "à droite"), par exemple ici en sommant depuis $1/t^2$ (à gauche) jusqu'à t^2 (à droite) (ci-dessous, les nombres 1, 2 et 3, etc. partent depuis l'extrémité droite du dessin, vers la gauche).



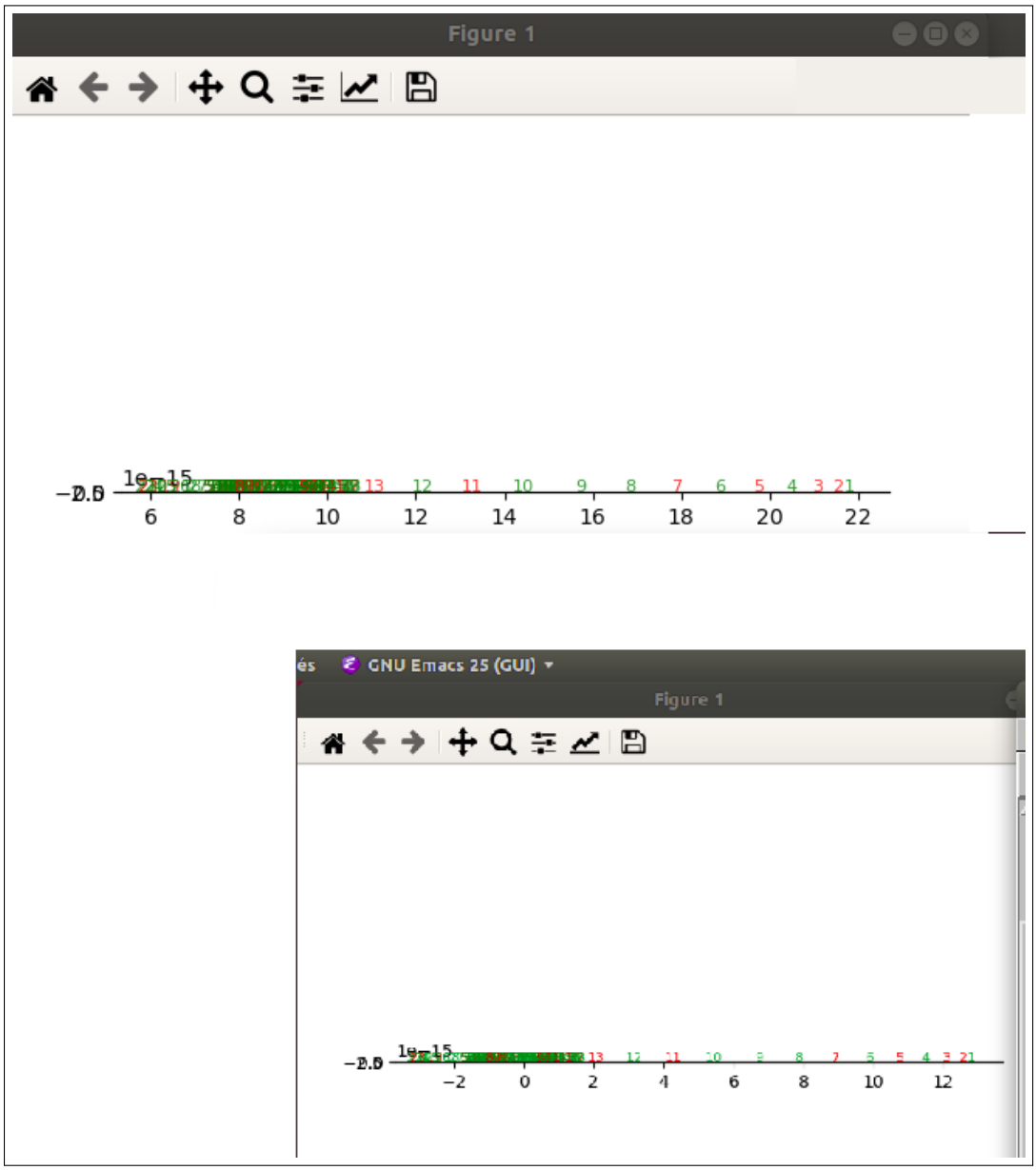
En allant "jusqu'aux cubes", les nombres sont plus écartés.



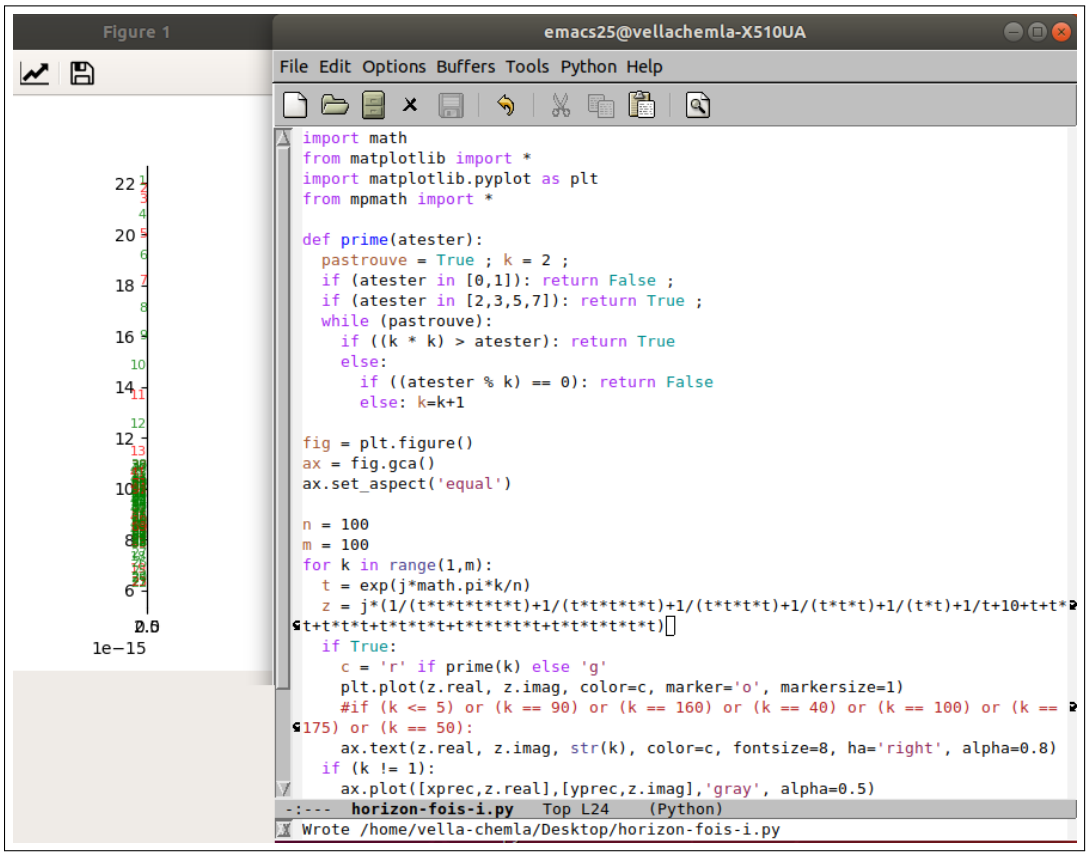
Même remarque jusqu'à la puissance 6 (côté inverse et côté puissances pures).



Si au milieu de la somme, on met bêtement un 2 au lieu de 1, ça translate, c'est tout.



Quand on a un alignement horizontal, on peut aussi très bêtement le transformer en alignement vertical en multipliant tout par i (ceci est une solution de facilité, sic).



On peut aussi obtenir des alignements sous-prétexte qu'on aurait pris l'exponentielle complexe de multiples de $2i\pi$, les deux dessins ci-après l'illustrent en montrant surtout la façon dont le fait de mettre n^3 ou bien n^4 au dénominateur de l'exposant fait chuter drastiquement les valeurs des images.

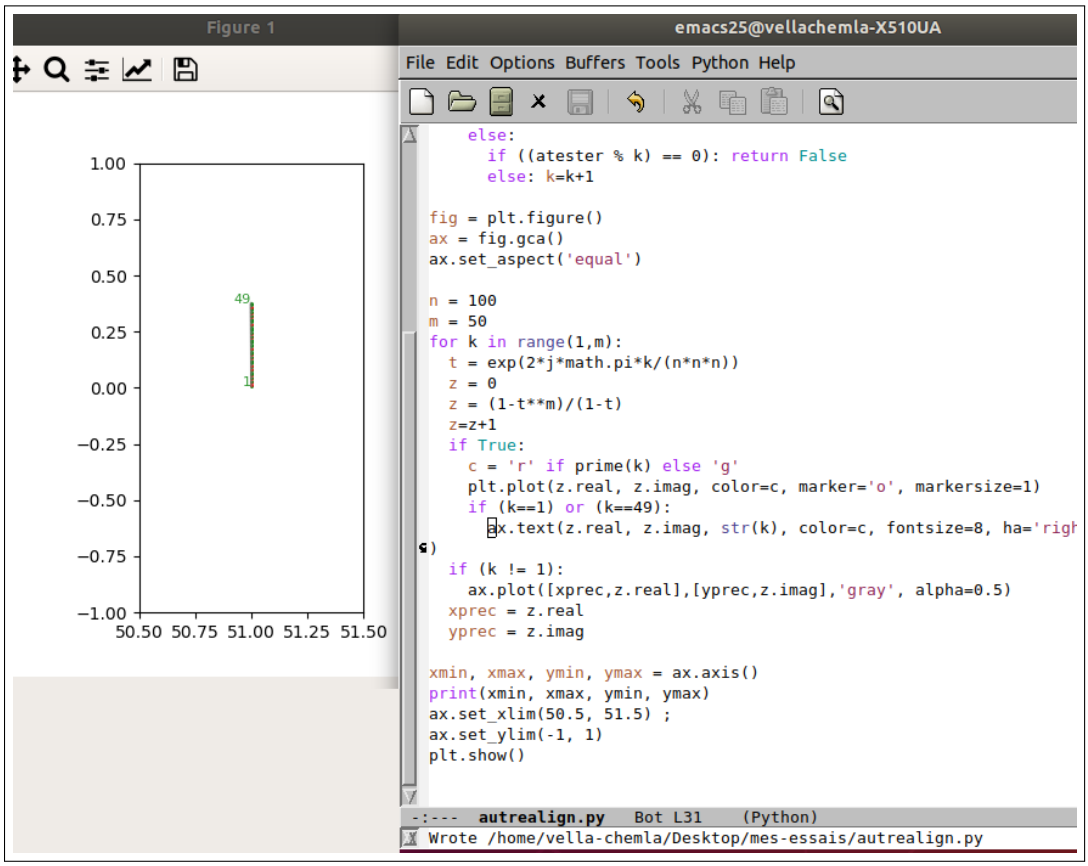


Figure 1
emacs25@vellachemla-X510UA
File Edit Options Buffers Tools Python Help
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
50.50 50.75 51.00 51.25 51.50
49
1
else:
if ((atester % k) == 0): return False
else: k=k+1
fig = plt.figure()
ax = fig.gca()
ax.set_aspect('equal')
n = 100
m = 50
for k in range(1,m):
t = exp(2*j*math.pi*k/(n*n*n))
z = 0
z = (1-t**m)/(1-t)
z=z+1
if True:
c = 'r' if prime(k) else 'g'
plt.plot(z.real, z.imag, color=c, marker='o', markersize=1)
if (k==1) or (k==49):
ax.text(z.real, z.imag, str(k), color=c, fontsize=8, ha='righ
t')
if (k != 1):
ax.plot([xprec,z.real],[yprec,z.imag],'gray', alpha=0.5)
xprec = z.real
yprec = z.imag
xmin, xmax, ymin, ymax = ax.axis()
print(xmin, xmax, ymin, ymax)
ax.set_xlim(50.5, 51.5) ;
ax.set_ylim(-1, 1)
plt.show()
-:--- autrealign.py Bot L31 (Python)
Wrote /home/vella-chemla/Desktop/mes-essais/autrealign.py

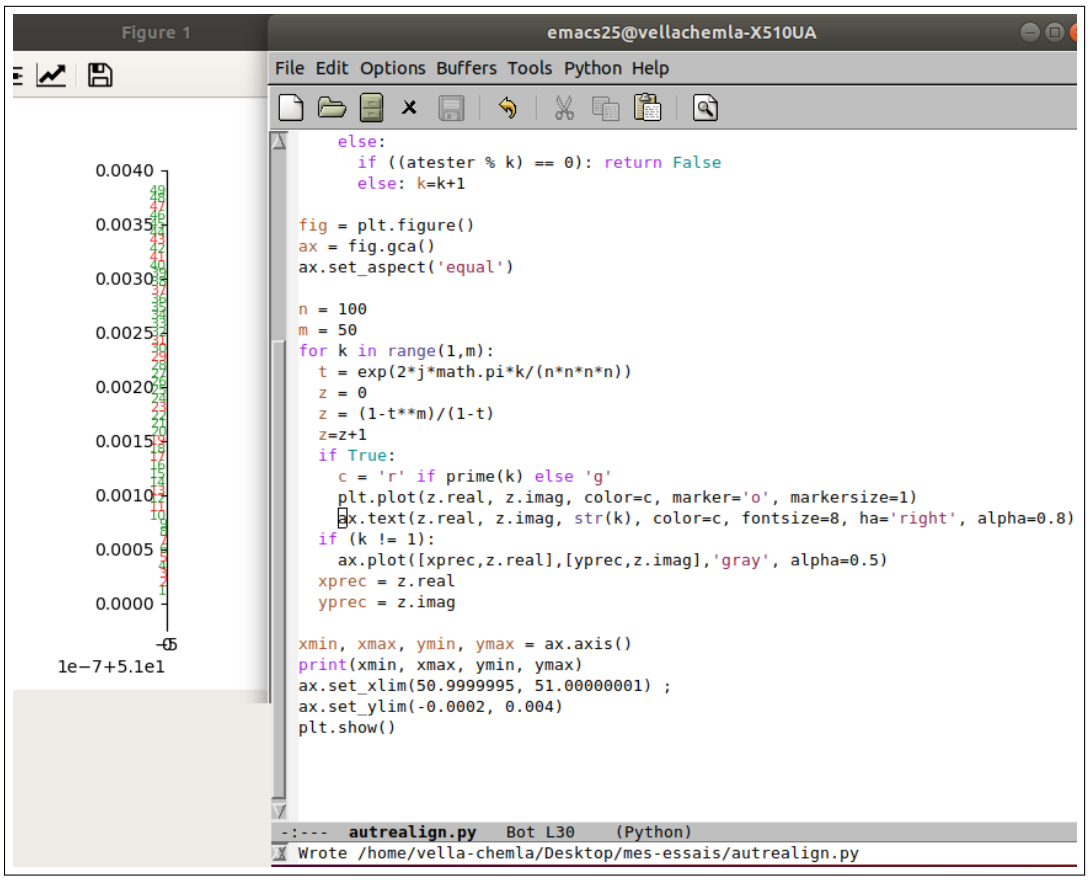
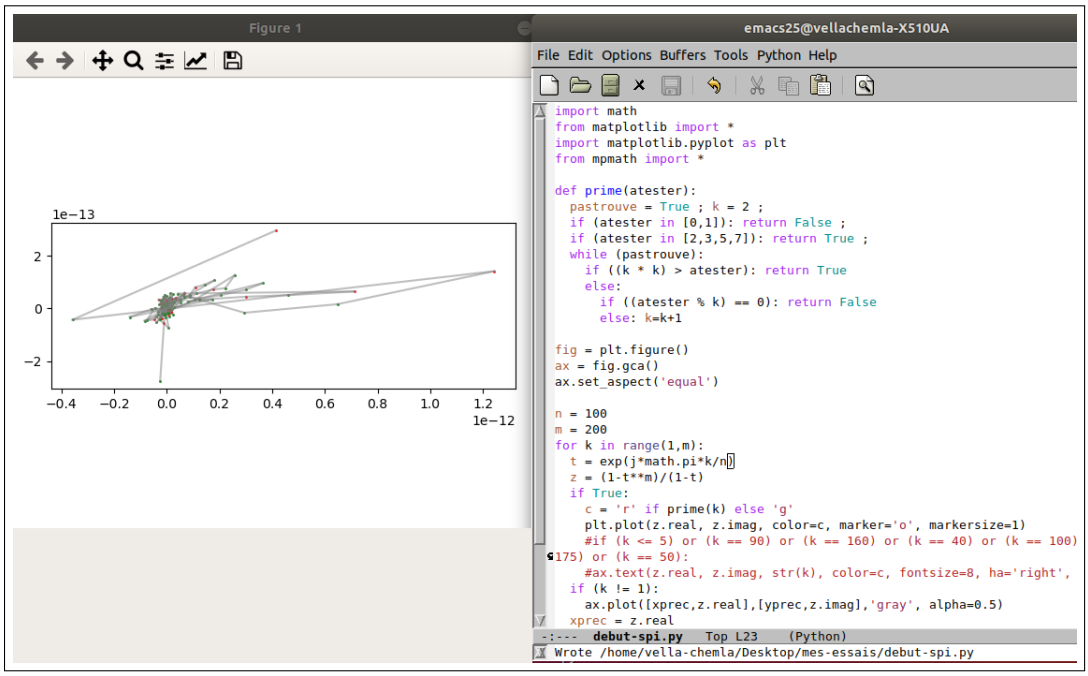
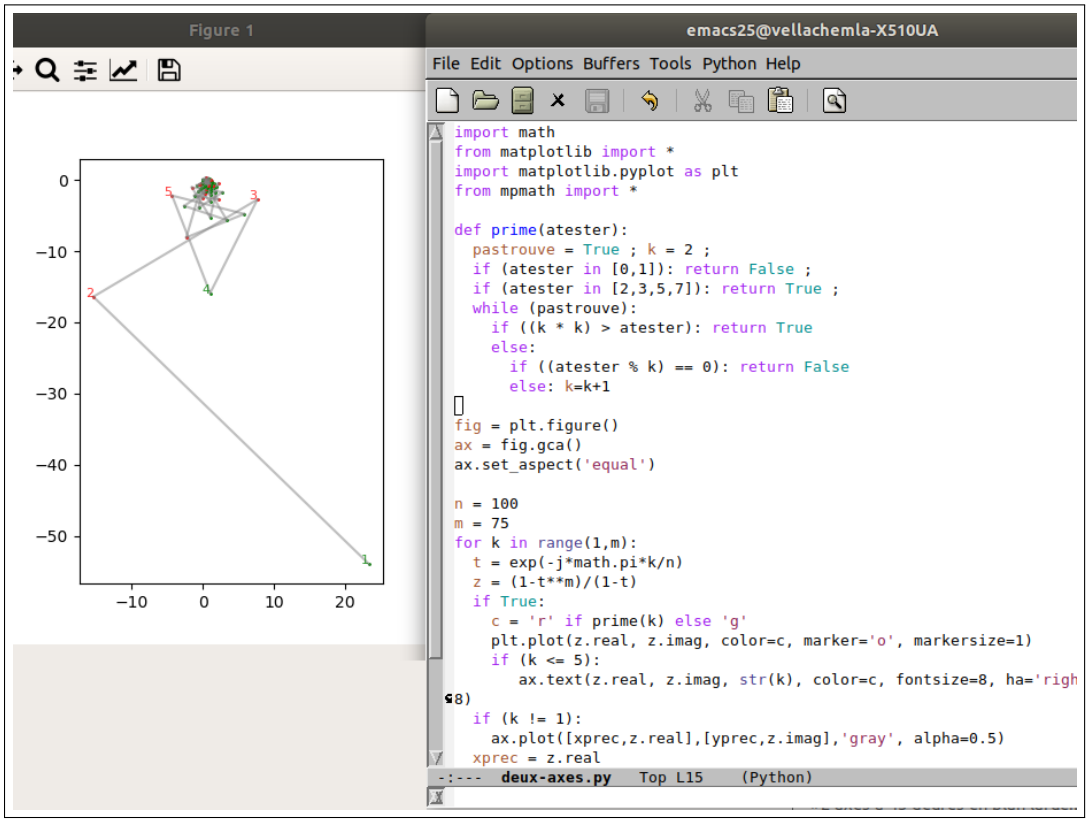


Figure 1
emacs25@vellachemla-X510UA
File Edit Options Buffers Tools Python Help
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
1e−7+5.1e1
else:
if ((atester % k) == 0): return False
else: k=k+1
fig = plt.figure()
ax = fig.gca()
ax.set_aspect('equal')
n = 100
m = 50
for k in range(1,m):
t = exp(2*j*math.pi*k/(n*n*n*n))
z = 0
z = (1-t**m)/(1-t)
z=z+1
if True:
c = 'r' if prime(k) else 'g'
plt.plot(z.real, z.imag, color=c, marker='o', markersize=1)
ax.text(z.real, z.imag, str(k), color=c, fontsize=8, ha='right', alpha=0.8)
if (k != 1):
ax.plot([xprec,z.real],[yprec,z.imag],'gray', alpha=0.5)
xprec = z.real
yprec = z.imag
xmin, xmax, ymin, ymax = ax.axis()
print(xmin, xmax, ymin, ymax)
ax.set_xlim(50.9999995, 51.00000001) ;
ax.set_ylim(-0.0002, 0.004)
plt.show()
-:--- autrealign.py Bot L30 (Python)
Wrote /home/vella-chemla/Desktop/mes-essais/autrealign.py

De toute façon, il faut toujours garder à l'esprit qu'on n'est pas à l'abri d'un accident de parcours illisible, comme sur l'exemple 1 :

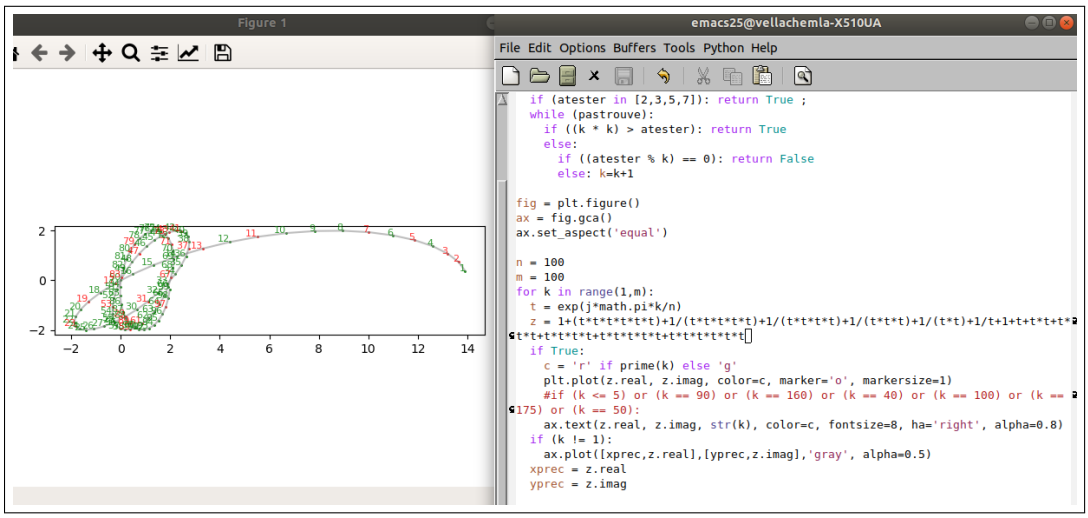


Même remarque à propos de l'exemple 2.



Merveilleux accident de parcours (regarder le code, un signe / est bêtement devenu un +) qui donne

naissance à une superbe signature !



Il faudra revenir sur celle qu'on a envie d'appeler l'araignée. Elle est plaisante parce qu'elle "rapproche très près" les décomposants de 100 et on distingue bien ici les décompositions de Goldbach de 100, deux points rouges proches. Peut-être tendraient-ils à être confondus...

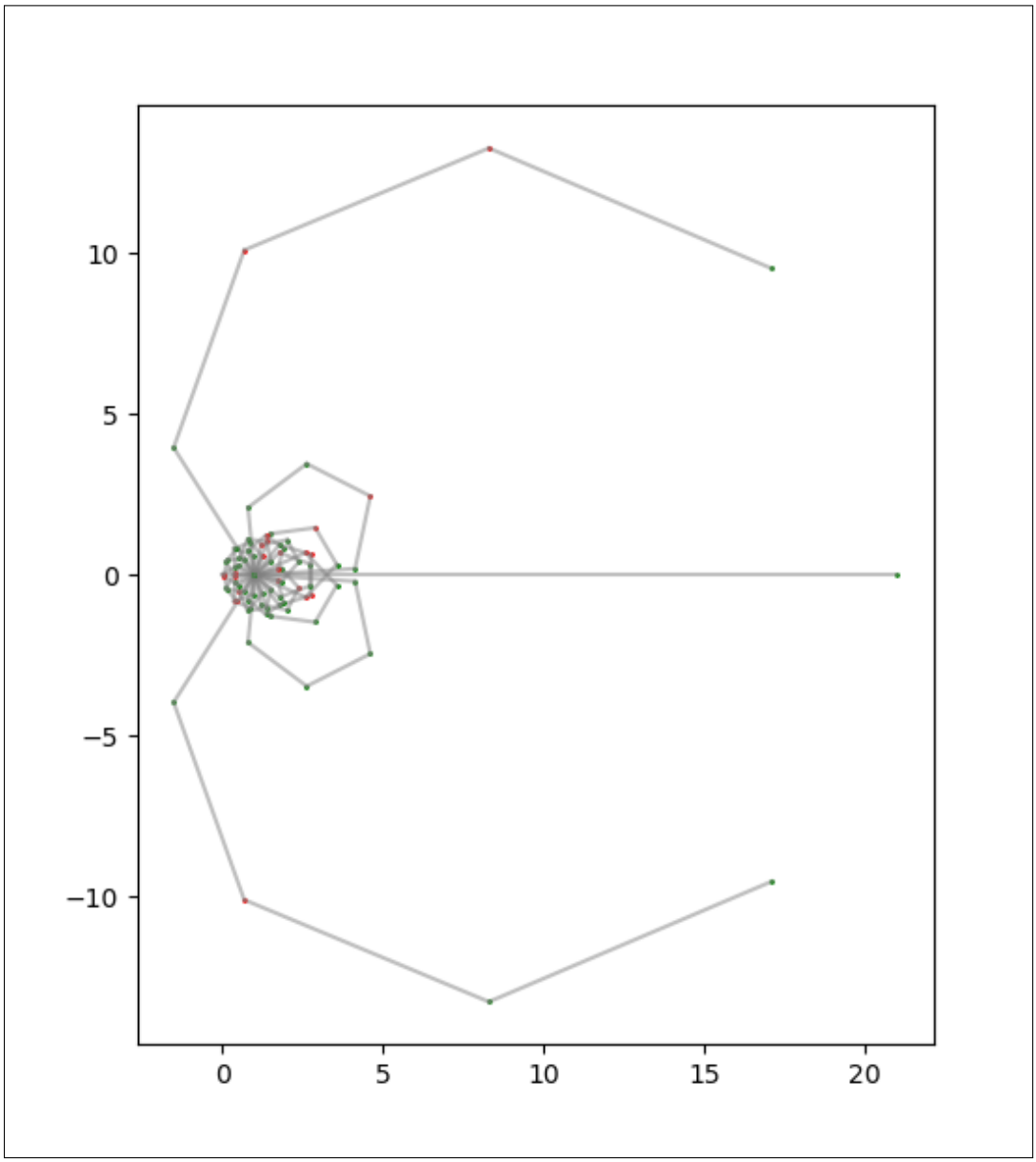
```
import math
from matplotlib import *
import matplotlib.pyplot as plt
from mpmath import *

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

fig = plt.figure()
ax = fig.gca()
ax.set_aspect('equal')

n = 100
m = 100
for k in range(1,m):
  t = exp(2*j*math.pi*k/n)
  z = 0
  for m in range(-1,19):
     z = z+t**m
  z=z+1
  if True:
    c = 'r' if prime(k) else 'g'
    plt.plot(z.real, z.imag, color=c, marker='o', markersize=1)
    ax.text(z.real, z.imag, str(k), color=c, fontsize=8, ha='right', alpha=0.8)
  if (k != 1):
    ax.plot([xprec,z.real],[yprec,z.imag],'gray', alpha=0.5)
  xprec = z.real
  yprec = z.imag

xmin, xmax, ymin, ymax = ax.axis()
print(xmin, xmax, ymin, ymax)
ax.set_xlim(-3, 6) ;
ax.set_ylim(-3, 3)
plt.show()
```


10
5
0
−5
−10
0
5
10
15
20

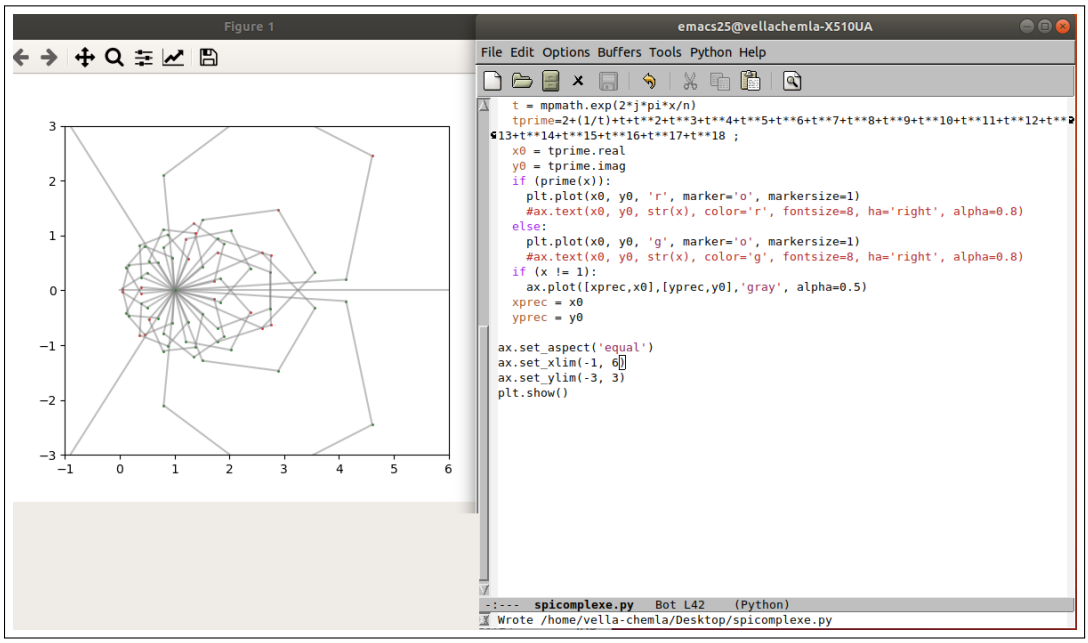
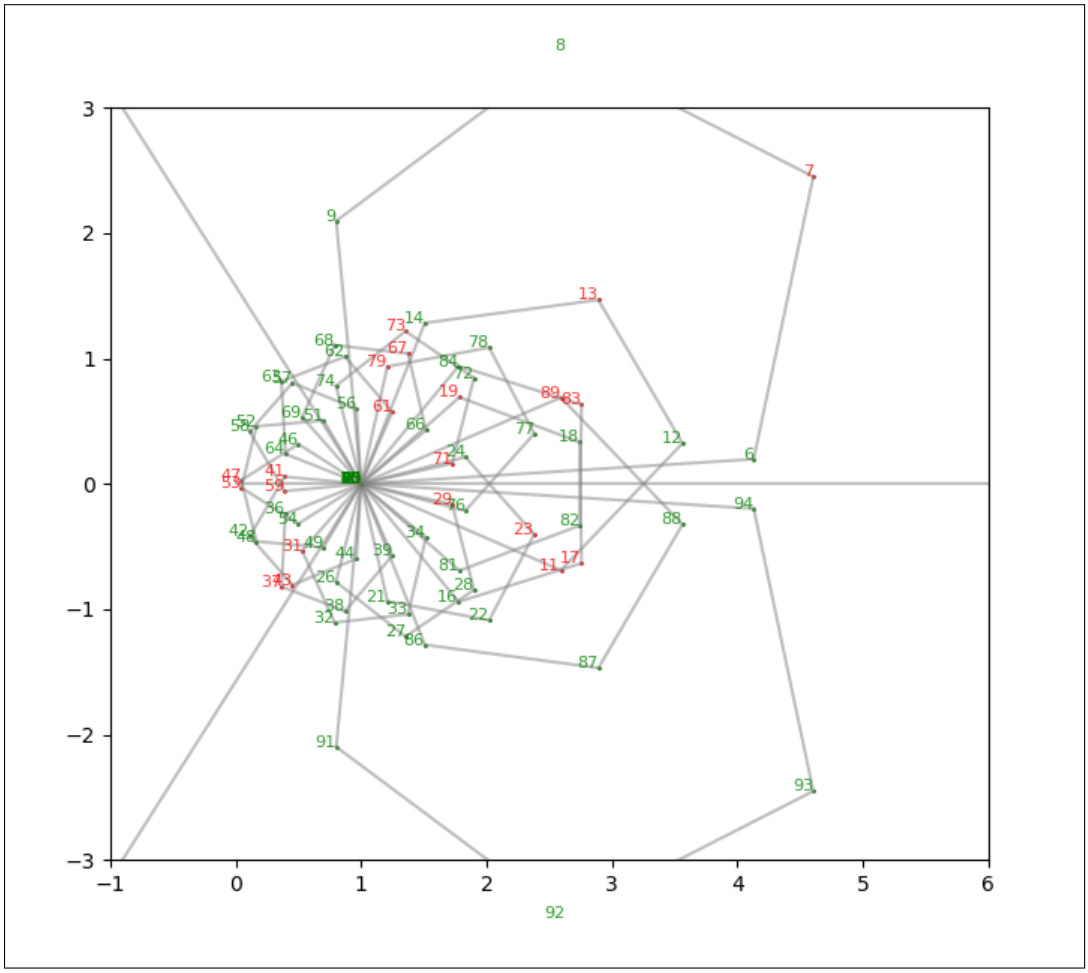


Figure 1
emacs25@vellachemla-X510UA
File Edit Options Buffers Tools Python Help
t = mpmath.exp(2*j*pi*x/n)
tprime=2+(1/t)+t+t**2+t**3+t**4+t**5+t**6+t**7+t**8+t**9+t**10+t**11+t**12+t**13+t**14+t**15+t**16+t**17+t**18 ;
x0 = tprime.real
y0 = tprime.imag
if (prime(x)):
plt.plot(x0, y0, 'r', marker='o', 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=0.5)
xprec = x0
yprec = y0
ax.set_aspect('equal')
ax.set_xlim(-1, 6)
ax.set_ylim(-3, 3)
plt.show()
:--- spicomplexe.py Bot L42 (Python)
Wrote /home/vella-chemla/Desktop/spicomplexe.py


8
92