Compilation des programmes écrits à la fin de l'été 2021, selon différentes sources (Denise Vella-Chemla, 12.10.2021)

1. Une idée qui accroche : ne pas annuler des produits de sinusoïdes

L'idée initiale que l'on souhaiterait développer pour comprendre pourquoi tout pair supérieur à 6 peut se décomposer en somme de deux nombres premiers impairs est simple : on implémente le crible d'Ératosthène par des sinusoïdes.

Les nombres premiers, n'étant divisibles par aucun nombre premier qui serait inférieur à leur racine, sont les points entiers où les produits de sinusoïdes ne s'annulent pas.

Ci-dessous, le détail des sinusoïdes associées aux nombres premiers 2, 3, 5 et 7, qui permettent de voir les nombres premiers de 11 à 97, on les a matérialisés par des points rouges et ils sont bien sûr positionnés hors des zéros des sinusoïdes.

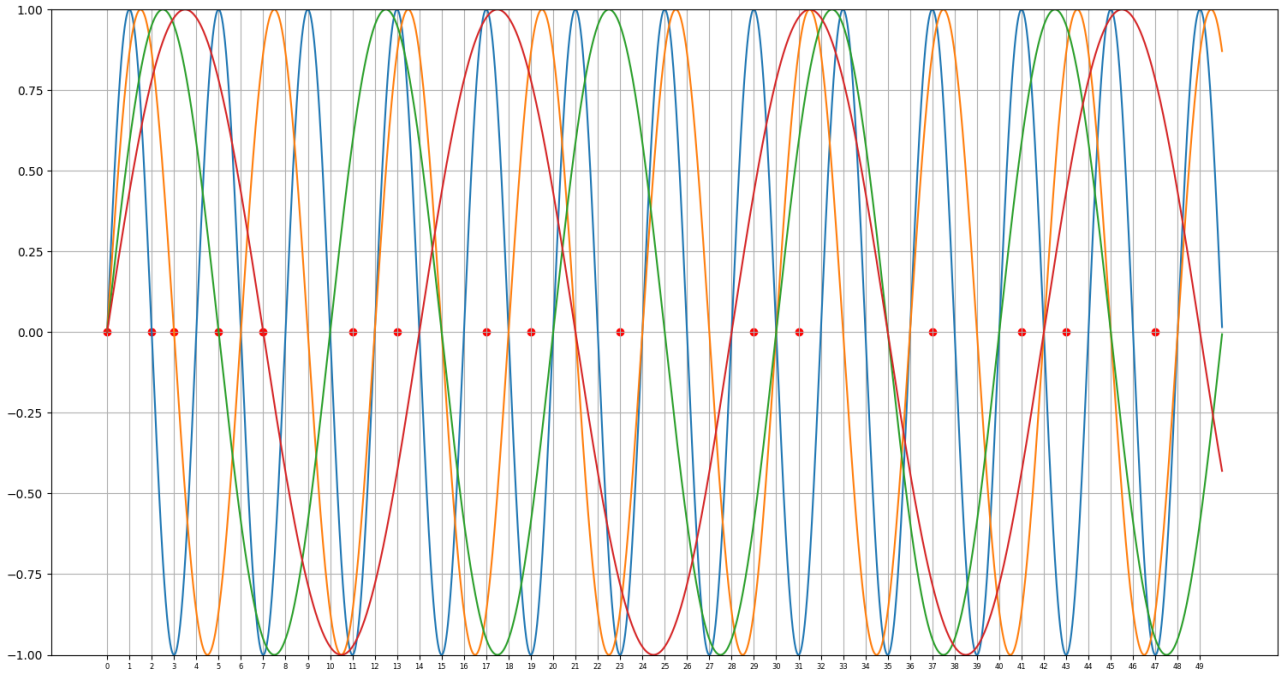


FIGURE 1 : les nombres premiers compris entre 11 et 97 n'annulent pas les sinusoïdes de la forme $\sin(\pi x/p)$ pour p premier égal à 2, 3, 5 ou 7.

Il faudrait utiliser une transformée de ces produits, fournie par le calcul opérationnel, peut-être... (voir les cours de calcul opérationnel dans les livres de Piskounov ou de Denis-Papin).

On trouve les décomposants de Goldbach en mettant en œuvre une symétrie qui rend le produit des produits de sinusoïdes pour x (x est premier) et des produits de sinusoïdes pour $n-x$ non nul ($n-x$ est premier). Ci-dessous les décompositions 19+79, 31+67 et 37+61 de 98.

```
import matplotlib.pyplot as plt
import math
import scipy.misc
import numpy as np

plt.ylim(-1,1)
plt.grid(True)
plt.xticks(range(0,100,1), fontsize=6)
x = np.arange(0, 100, 0.01)
plt.plot(x, [   math.sin(tt*math.pi/2)
                *math.sin(tt*math.pi/3)
                *math.sin(tt*math.pi/5)
                *math.sin(tt*math.pi/7)
                *math.sin((98-tt)*math.pi/2)
                *math.sin((98-tt)*math.pi/3)
                *math.sin((98-tt)*math.pi/5)
                *math.sin((98-tt)*math.pi/7) for tt in x])
plt.scatter([   2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,
                53,59,61,67,71,73,79,83,89,97],
                [0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0],c='red')
plt.show()
```

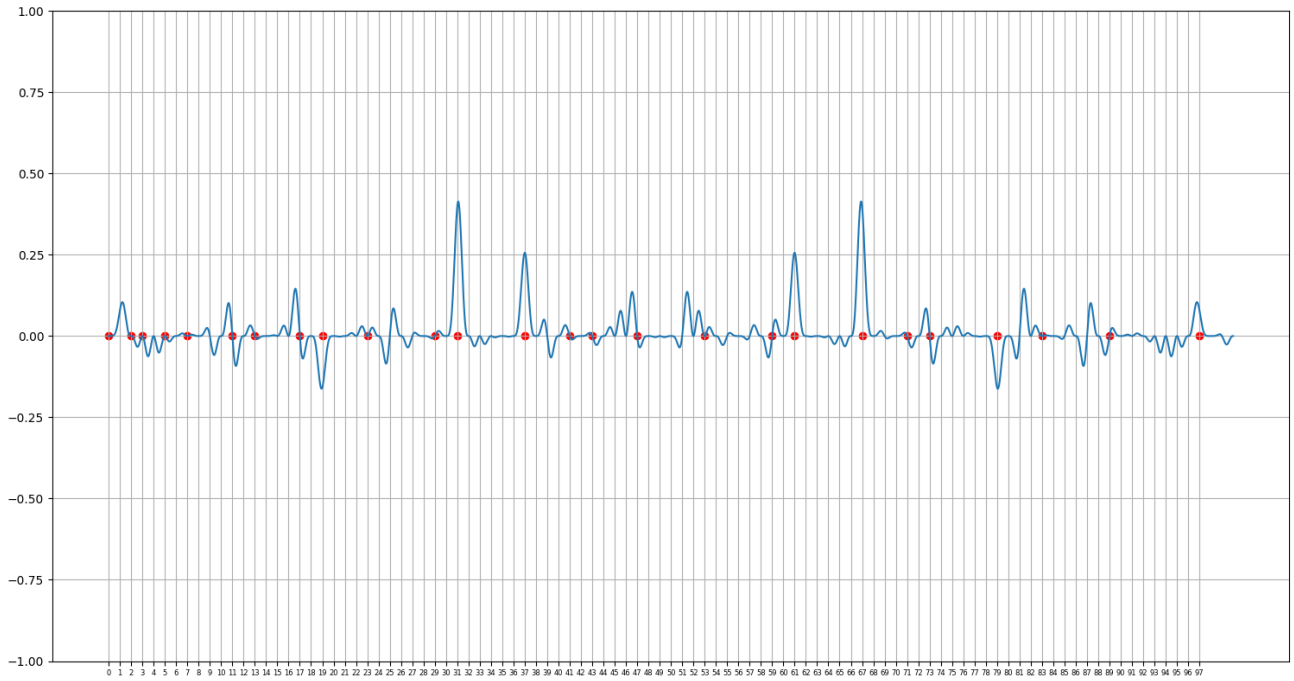


FIGURE 2 : 19, 31, 37, 61, 67, 79 étant des nombres premiers dont le complémentaire à 98est premier (i.e. n'annulant pas un produit de 8 sinusoïdes à lire dans le programme au-dessus de son résultat) sont des décomposants de Goldbach de 98.

On aimerait symétriser le problème : on décale les sinusoïdes de façon à ce que 0 corresponde à $n/2$ (ici 98/2=49) pour que les décomposants de Goldbach soient des nombres x et $-x$ (pour 98, on voit le décalage des pics en -30 et 30, -18 et 18 et -12 et 12).

```
1  import matplotlib.pyplot as plt
2  import math
3  import scipy.misc
4  import numpy as np
5
6  plt.ylim(-1,1)
7  plt.grid(True)
8  plt.xticks(range(-50,50,1), fontsize=6)
9  x = np.arange(-50, 50, 0.01)
10 plt.plot(x, [   math.sin((tt+49)*math.pi/2)
11                *math.sin((tt+49)*math.pi/3)
12                *math.sin((tt+49)*math.pi/5)
13                *math.sin((tt+49)*math.pi/7)
14                *math.sin((98-(tt+49))*math.pi/2)
15                *math.sin((98-(tt+49))*math.pi/3)
16                *math.sin((98-(tt+49))*math.pi/5)
17                *math.sin((98-(tt+49))*math.pi/7) for tt in x])
18 plt.scatter([-30,-18,-12,12,18,30],[0,0,0,0,0,0],c='red')
19 plt.show()
```

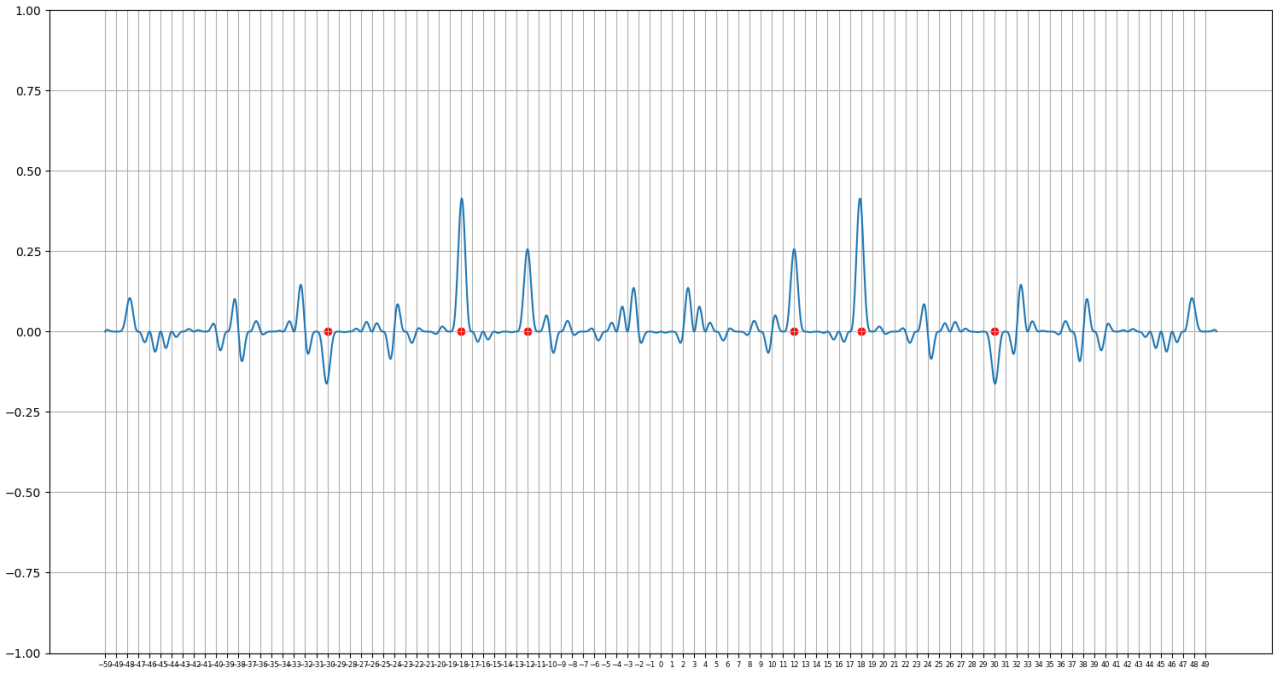


FIGURE 3 : en décalant les sinusoïdes de 49 à gauche, les décomposants de Goldbach ont été décalés en -30, -18, -12, 12, 18 et 30 et le produit de sinusoïdes est maintenant une fonction paire.

Redheffer avait également eu cette idée des produits de sinusoïdes en 1951 [1].

2. Lumière sur scie circulaire de van der Pol

On a implémenté une manière d'approximer les zéros de la fonction zeta de Riemann proposée dans deux articles de van der Pol, un article présentant un dispositif expérimental (une sorte de scie circulaire sur laquelle sont envoyés des rayons lumineux) [2] et un article faisant le lien entre le calcul opérationnel et la théorie des nombres [3].

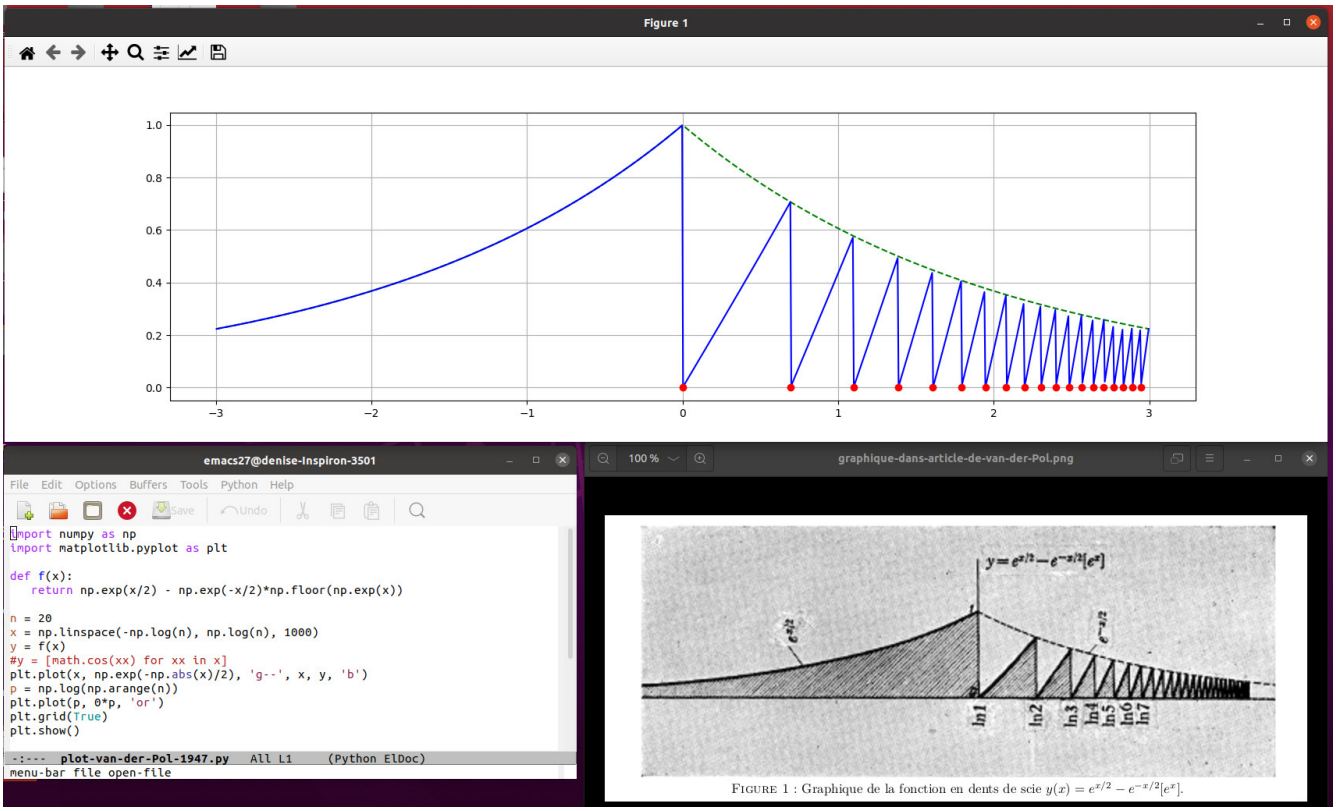


FIGURE 1 : Graphique de la fonction en dents de scie $y(x) = e^{x/2} - e^{-x/2}[e^x]$.

```python
from scipy.integrate import quad
import matplotlib.pyplot as plt
import math

xlim = 10
nbpts = 2000

def y(x): return math.exp(-x/2)*(math.exp(x) % 1)

def z(t):
    f = lambda x: -y(x) * math.cos(-x*t)
    g = lambda x: -y(x) * math.sin(-x*t)
    a = quad(f, -xlim, xlim, limit=200)[0]
```

1. voir http://denise.vella.chemla.free.fr/Redheffer.pdf, p. 1)
2. voir la traduction de l'article ici http://denisevellachemla.eu/trad-van-der-Pol.pdf, en se reportant aux égalités (10) à (12).
3. http://denisevellachemla.eu/trad-vdp-premiers.pdf, voir l'égalité (12a).

```
    b = quad(g, -xlim, xlim, limit=200)[0]
    return math.sqrt(a*a + b*b)

n = int(math.exp(xlim))
a, b, c = 0, 100, nbpts
t = [a+k*(b-a)/c for k in range(c)]
plt.ylim(0,0.2)
plt.grid(True)
plt.xticks(range(0,100,10), fontsize=6)
plt.plot(t, [abs(z(tt)) for tt in t])
plt.show()
```

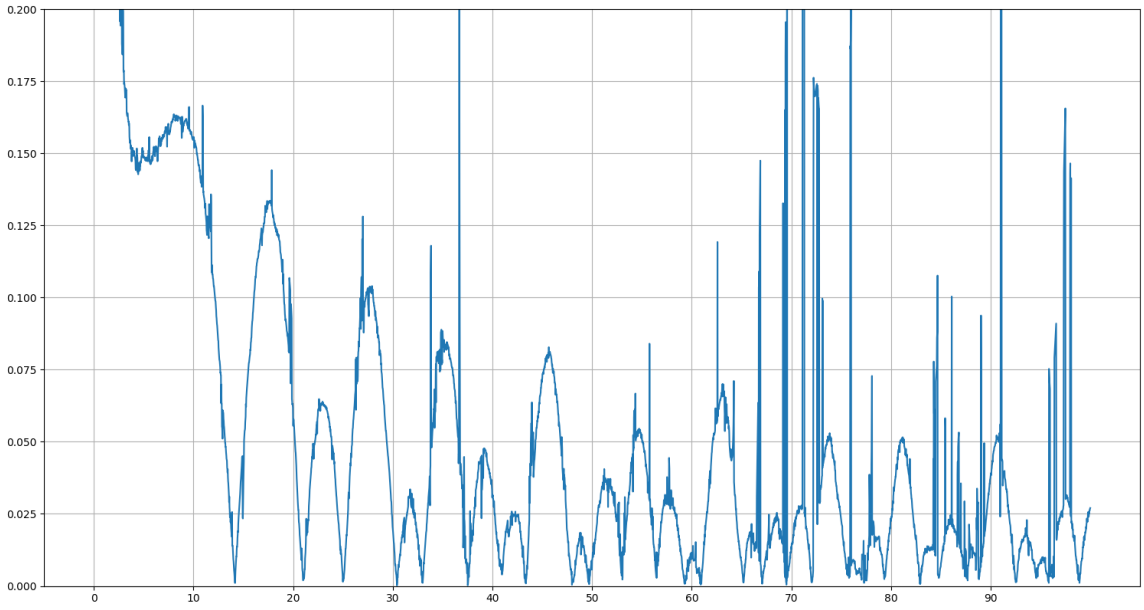


FIGURE 4 : Implémentation de l'équation (10) de l'article de van der Pol de 1947.

Ci-dessus, on a utilisé l'équation (10) de l'article de 1947 de van der Pol. Ci-dessous, on peut approximer zeta dans la bande critique en utilisant l'équation (9) du même article et visualiser les 3 premiers zéros par exemple.

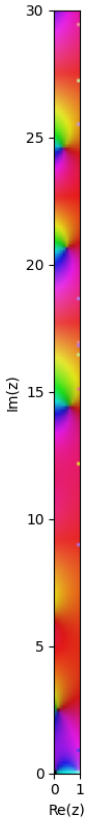
```
import time
import mpmath
import scipy.integrate as synt

def f(s,u): return -(u % 1) / u**(s+1)

def z(s):
     a, b, l = 0, 4, 100
     x = synt.quad(lambda u:f(s,u).real, a, b, limit=l)[0]
     y = synt.quad(lambda u:f(s,u).imag, a, b, limit=l)[0]
     return s*(x+1j*y)

```

```
13 tic=time.time()
14 mpmath.cplot(z, [0,1], [0,30], points = 2000)
15 tac=time.time()
16 print('temps␣ coul ␣:␣', tac-tic, '␣s.')
```



Noter qu'on a remplacé $e^x - \lfloor e^x \rfloor$ par $e^x \bmod 1$ (le symbole % est utilisé pour coder le reste modulaire en python).

Rappel : Valeurs des parties imaginaires des premiers zéros non triviaux de la fonction zeta de Riemann au millième (trouvés sur le site d'Andrew Odlyzko)

1	$\to 14.134$	11	$\to 52.970$	21	$\to 79.337$
2	$\to 21.022$	12	$\to 56.446$	22	$\to 82.910$
3	$\to 25.010$	13	$\to 59.347$	23	$\to 84.735$
4	$\to 30.424$	14	$\to 60.831$	24	$\to 87.425$
5	$\to 32.935$	15	$\to 65.112$	25	$\to 88.809$
6	$\to 37.586$	16	$\to 67.079$	26	$\to 92.491$
7	$\to 40.918$	17	$\to 69.546$	27	$\to 94.651$
8	$\to 43.327$	18	$\to 72.067$	28	$\to 95.870$
9	$\to 48.005$	19	$\to 75.704$	29	$\to 98.831$
10	$\to 49.773$	20	$\to 77.144$		

3. Les 2 fonctions fournies lors d'une conférence ainsi que dans le livre de Barry Mazur

Dans une conférence [4] de vulgarisation présentant les nombres premiers et la fonction zeta de Riemann, Barry Mazur fournit deux fonctions que l'on peut implémenter.

On fournit ci-dessous les diapos qui définissent les fonctions f et g.

La fonction g montre les parties imaginaires des zéros de zeta comme apparaissant dans certains pics de base suffisamment large. Mazur le nomme le spectre de Riemann.

La fonction f établit le lien qui permet de "voir" les nombres premiers (2, 3, 5, 7, etc.), en les calculant à partir des zéros de la fonction zeta de Riemann, i.e. à partir du spectre de Riemann : les nombres premiers sont comme pointés. Cependant, si on "dé-zoome", on constate que le pointage est très perturbé, on est loin d'un peigne de Dirac, avec ses pics bien nets, et une fonction nulle partout en dehors des pics.

On a noté quelques nombres premiers, pas forcément sur l'axe des abscisses, pour faciliter la visualisation.

packaging the information given by prime powers

$$g(t) \quad =$$

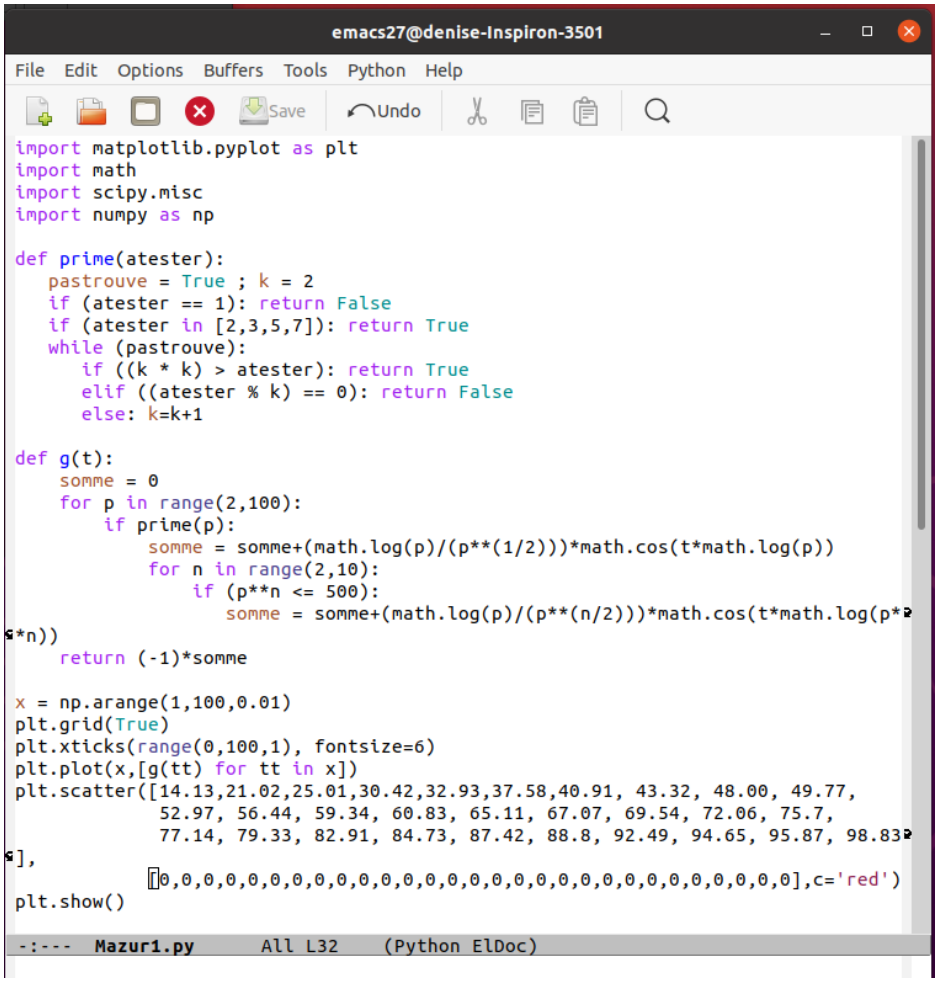
$$= \quad -\sum_{p^n} \frac{\log(p)}{p^{n/2}} \cos(t \log(p^n).)$$

From the Riemann Spectrum to primes

$$f(s) \quad =$$

$$= \quad 1 + \sum_{i} \cos(\theta_i \cdot \log(s))).$$

4. https://www.youtube-nocookie.com/embed/way0jAWpjZA?autoplay=1iv$_{l}oad_{p}olicy$ = 3$loop$ = 1$modestbranding$ = 1$playlist$ = $way0jAWpjZA$.



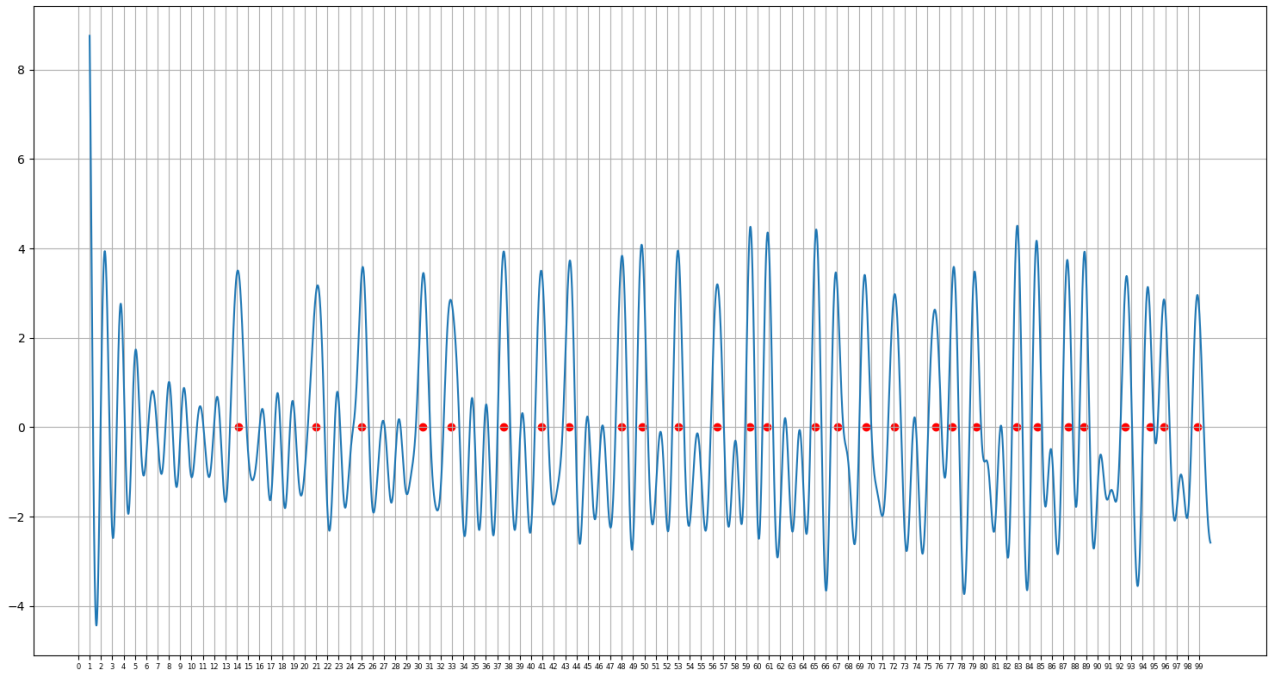
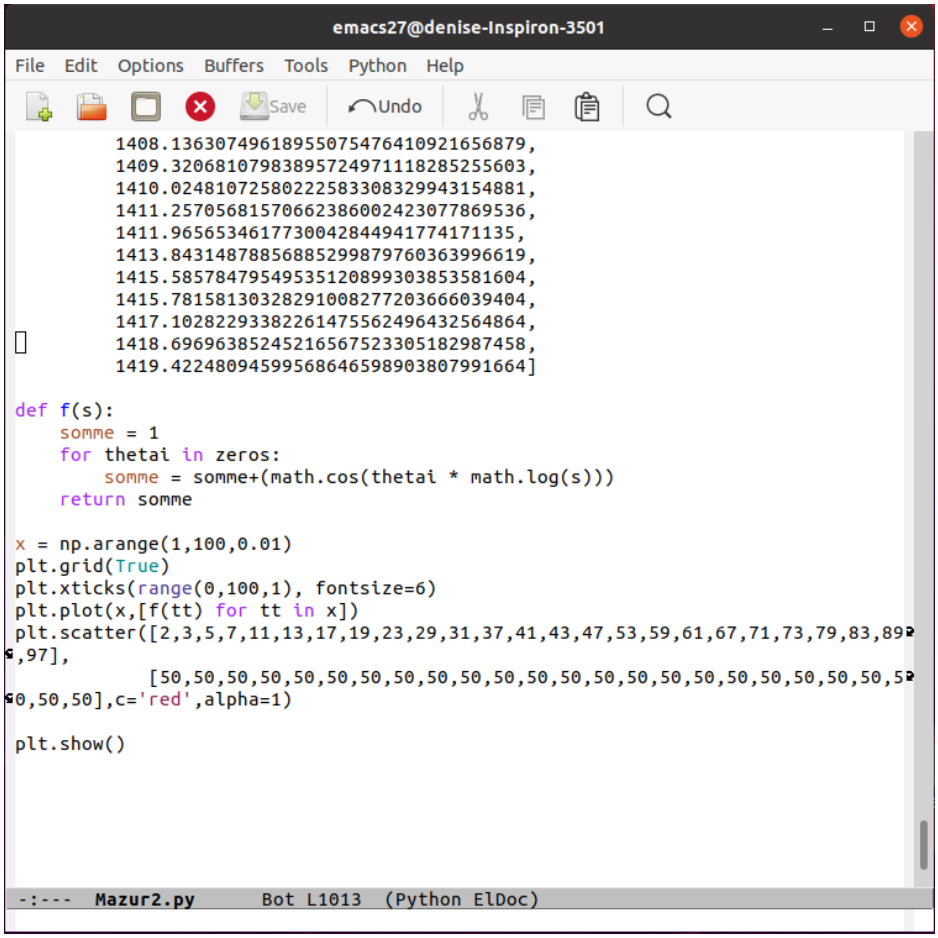


FIGURE 5 : la fonction g met les parties imaginaires des zéros de la fonction zeta de Riemann au centre de pics.



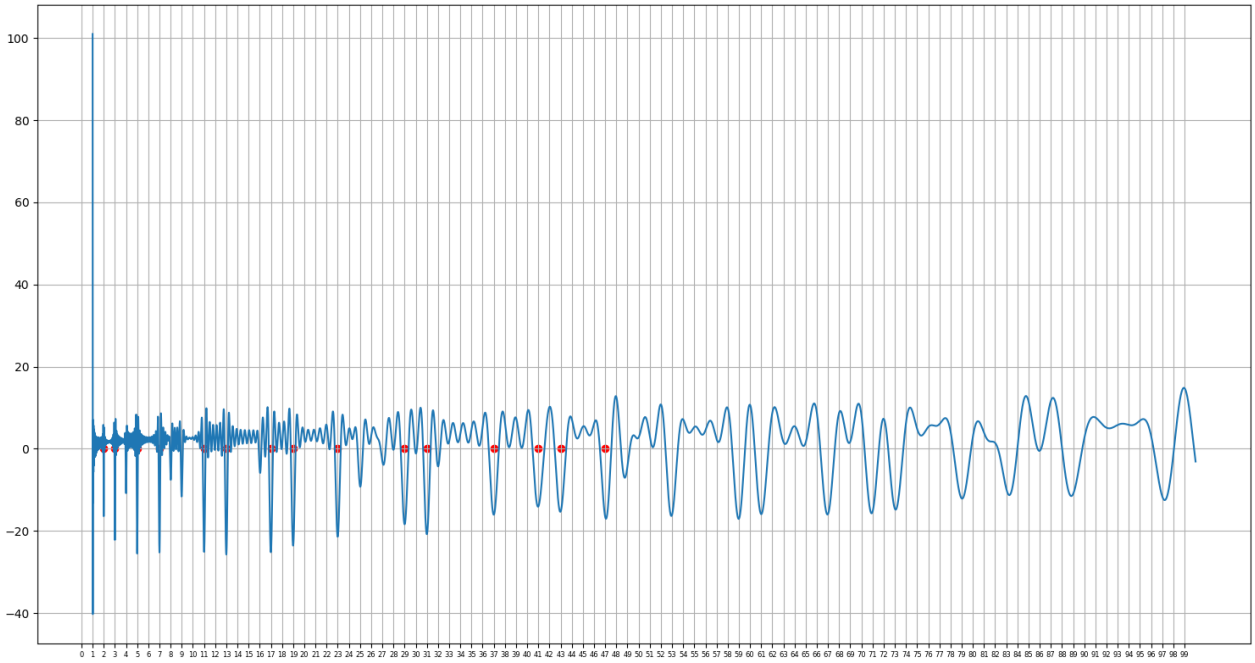
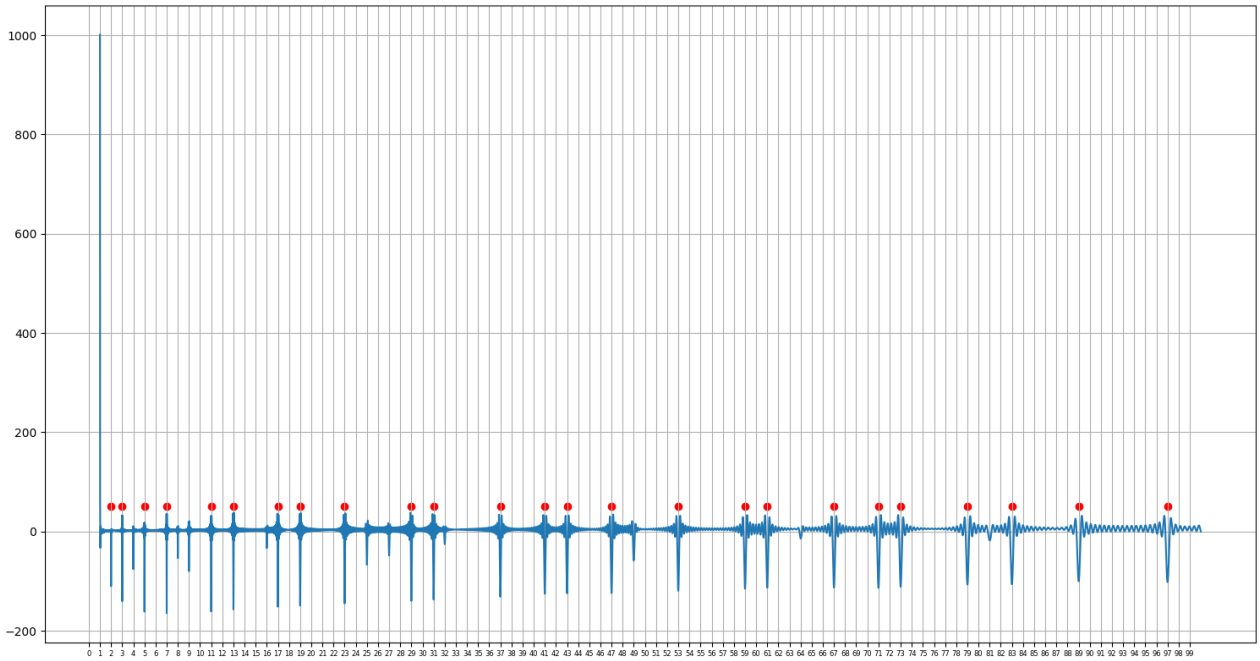
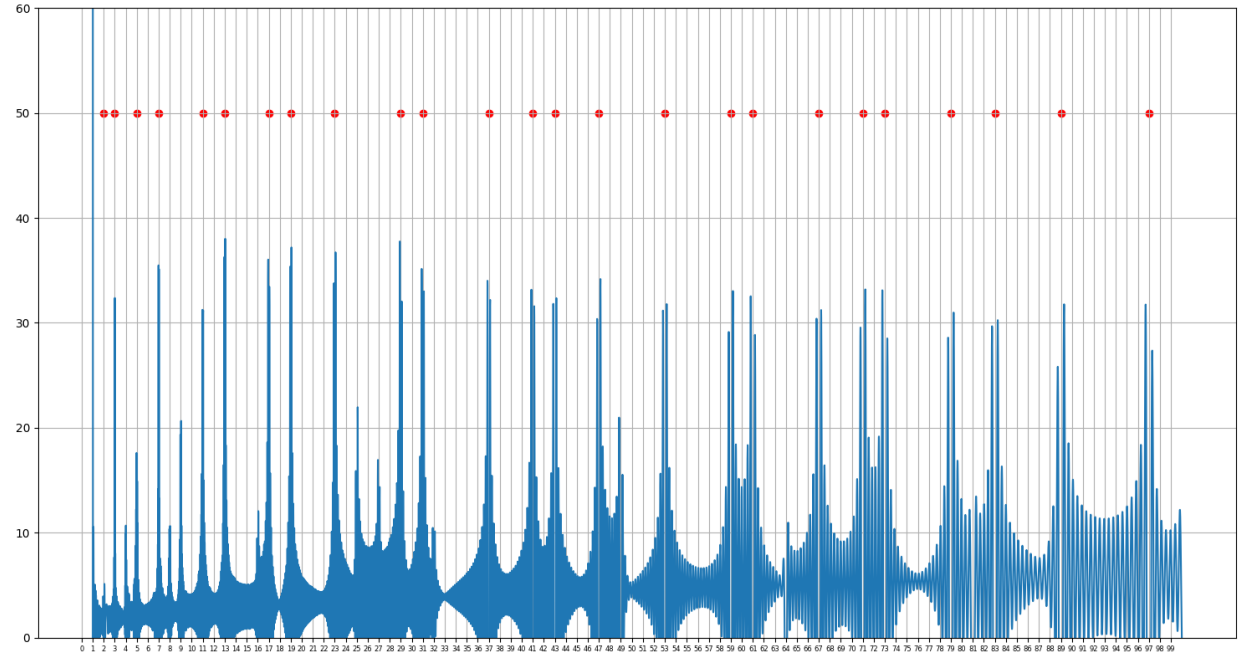


FIGURE 5 : la fonction f permet d'obtenir, à partir des parties imaginaires des zéros de la fonction zeta de Riemann, en les faisant comme "pointer" par des sortes de dents d'un peigne de Dirac grossier, les nombres premiers 2, 3, 5, 7, etc. Plus on utilise de zéros (en commençant par les zéros de parties imaginaires les plus petites), plus on arrive à serrer de zéros ; plus on s'approche, plus on voit combien la fonction est perturbée.



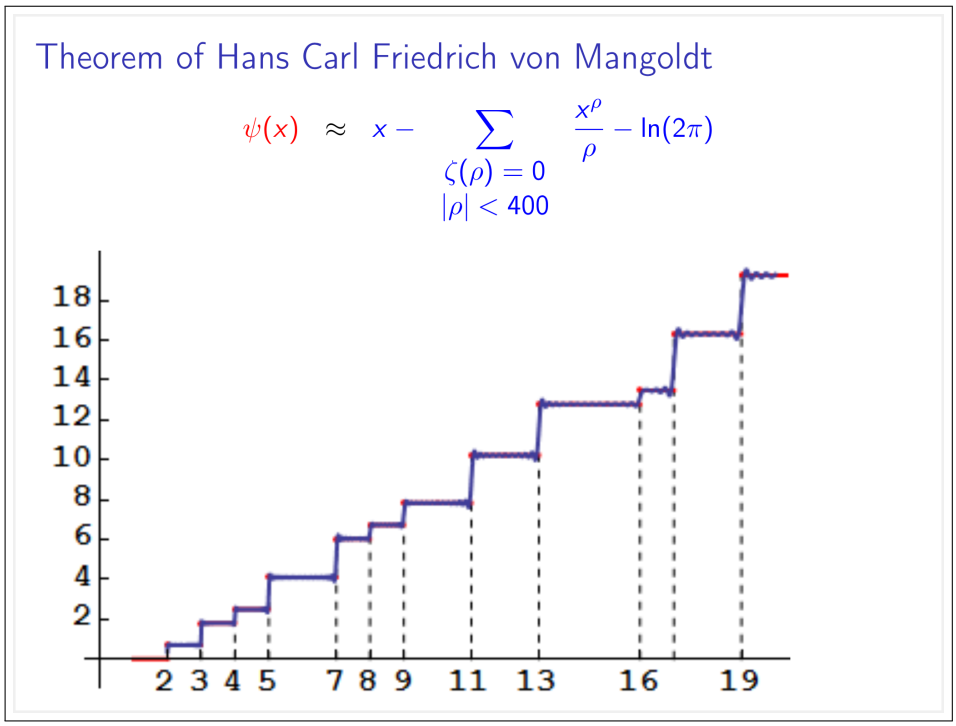


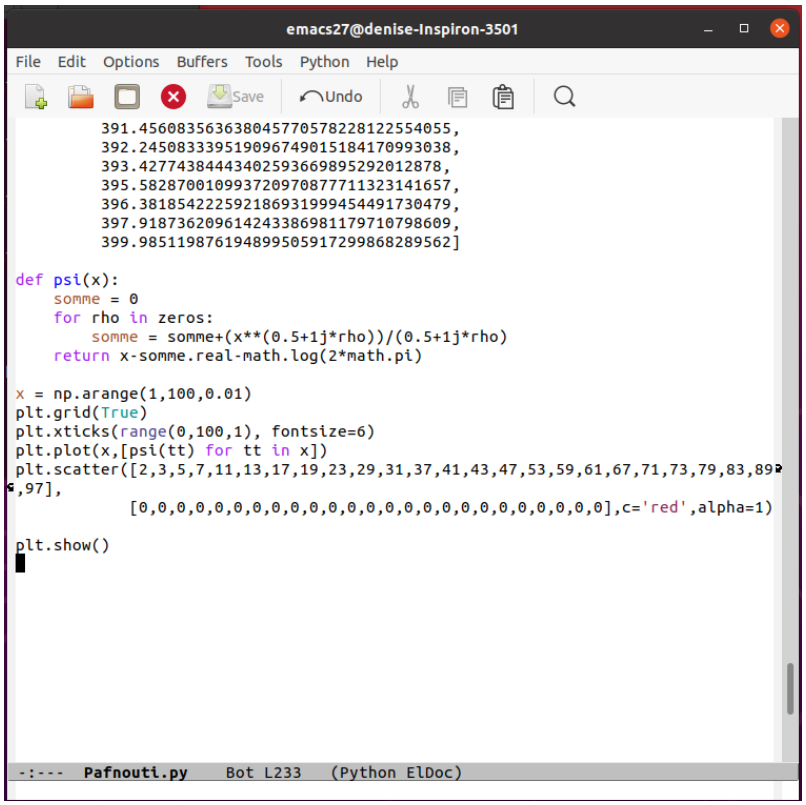
4. La fonction ψ de Chebyshev, dont la correction a été démontrée par von Mangoldt, et fournie dans une conférence et des articles de Matiyasevich

Dans une conférence à Stanford [5], Yuri Matiyasevich fournit une fonction en escalier, qui utilise les zéros de zeta, et dont les marches coïncident avec les nombres premiers ou leurs puissances pures.

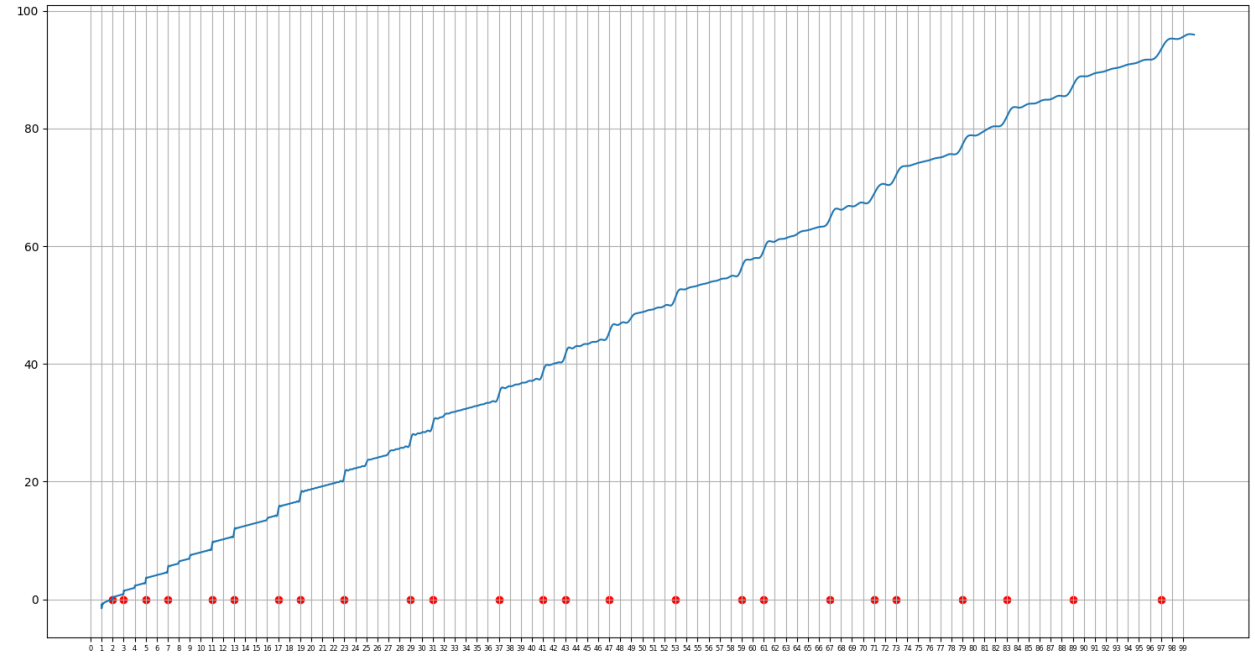
5. lien et diaporama dans cette page https://logic.pdmi.ras.ru/ yumat/talks/talks.php?istate=state$_s how_t alkiid = 1377.

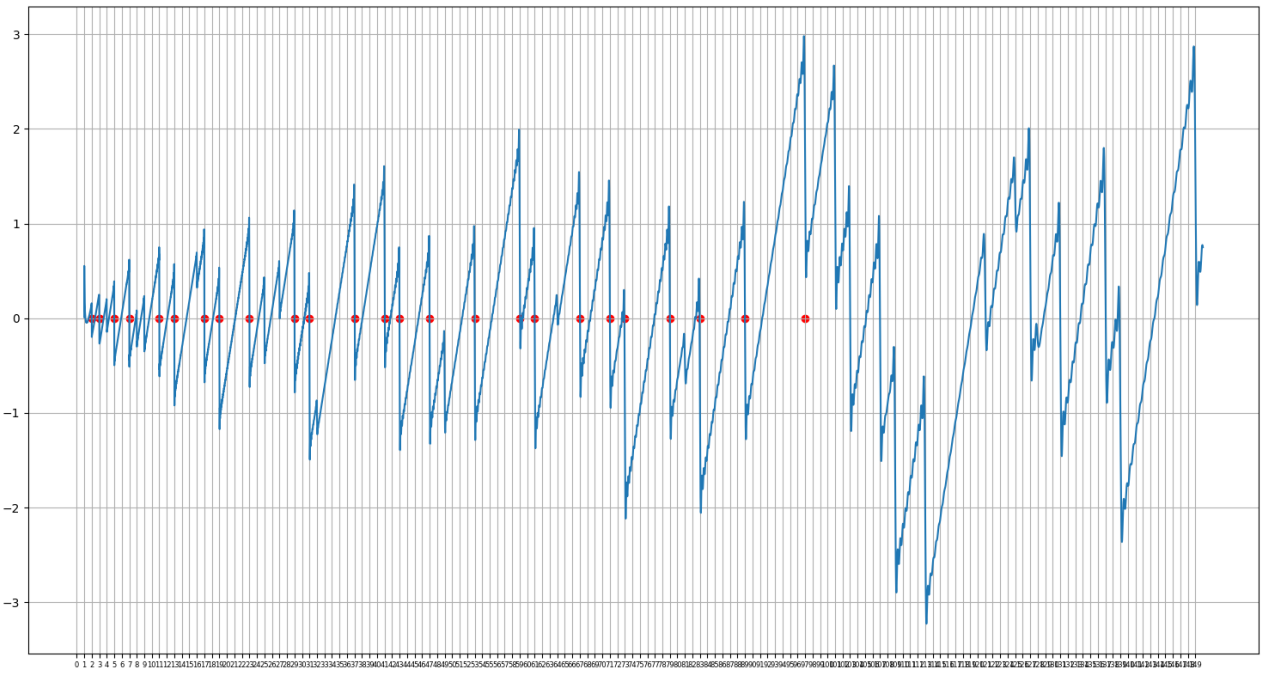
On fournit copie de la diapo qui fournit la fonction. On la calcule par programme, ainsi que la somme intérieure qui est soustraite à x, pour voir son allure générale. On marque les nombres premiers inférieurs à 100 sur l'axe des abscisses, pour améliorer la visualisation [6].





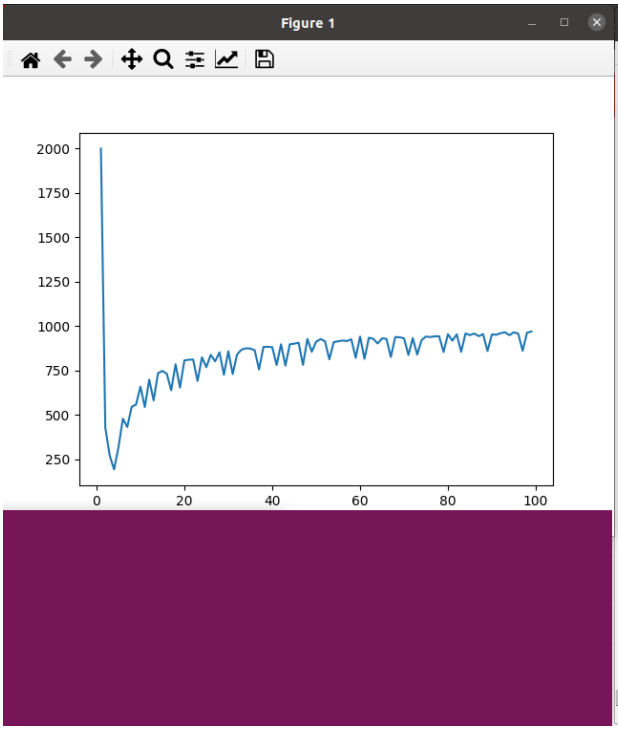
6. On omet de noter les puissances pures, comme 9 ou 16, bien qu'elles donnent cependant bien lieu à des petites marches.





5. Une fonction S dont on ne se rappelle plus trop d'où elle vient, mais qui "pique" aussi les premiers et qui utilise la fonction logarithme intégral

La fonction ci-dessous pointe les nombres premiers en appliquant la fonction logarithme intégral aux zéros de zeta puis en effectuant d'autres traitements, mais on ne se rappelle plus son origine.



emacs27@denise-Inspiron-3501

File Edit Options Buffers Tools Python Help

Save Undo

```
import math
from math import *
import cmath
from cmath import *
import mpmath
from mpmath import *
import scipy
from scipy import integrate
import numpy
from numpy import *
import matplotlib.pyplot as plt

zeros=[]
with open('leszeros1000', 'r') as f:
    for p in f.readlines():
        z = float(p.split()[1])
        zeros.append(z)
f.close()
print(z)
print('')
lesx = []
lesy = []
for x in range(1,100):
    lesx.append(x)
    somme = 0.0
    for k in range(1,1000):
        Lidexpuissrho=li(x**(0.5+zeros[k]*j))-li(2.0)
        somme = somme+2.0/sqrt(1.0+(Lidexpuissrho.imag/Lidexpuissrho.real)**2.0)
    print(x)
    print(somme)
    lesy.append(somme)
plt.plot(lesx,lesy)
plt.show()
```

-:--- **pgmcalcS.py** All L31 (Python ElDoc)

Wrote /home/denise/Bureau/revenir-sur-ses-pas/pgmcalcS.py