Pour python, ei, li, même combat, Denise Vella-Chemla, juillet 2025

On fournit simplement ici le programme python et son résultat, qui illustrent ce qu'a dit Alain Connes en février à l'IHP et à nouveau au RISM à Varèse en juin : Riemann aurait dû utiliser dans sa formule (et il en était pleinement conscient), plutôt que la fonction logarithme intégral $\mathrm{Li}(x^\rho)$ la fonction exponentielle intégrale $\mathrm{Ei}(\rho \log\ x)$, car la fonction logarithme intégral prenant de multiples valeurs égales par ajouts de multiples de $\exp(2\pi\ i)$, les zéros de la fonction ζ de Riemann auraient été très très très nombreux (bien plus qu'ils ne sont), car accompagnant le premier zéro, par exemple, il y aurait eu sur l'axe tous les zéros qui sont équivalents au premier zéro par cette relation d'équivalence "tournante" ; la fonction exponentielle intégrale quant à elle ne provoque pas cette multiplicité des racines.

```
import time
import numpy as np
import mpmath
from mpmath import li, ei
import math
from math import isqrt, sqrt, floor, pi, log, e
from scipy.integrate import quad

class Premiers():
     def __init__(self, n):
          premier = np.full(n, True)
          premier [:2] = False
          for p in range(2, math.isqrt(n)+1):
               if premier[p]:
                    premier[p*p::p] = False
          self.__premiers = np.nonzero(premier)[0]
     def compte(self, x):
          return np.searchsorted(self.__premiers,x, side ='right')

def lafct(t):
     return(1/((t*t-1)*t*log(t)))

def troisiemeterme(x):
     return(quad(lambda x:lafct(x),x,np.inf,limit=1000)[0])

with open('zeros1000', 'r') as f:
     lines = f.readlines()
     zeros = [float(line) for line in lines]
tic = time.time()
P = Premiers(10000000001)
```

```
for x in range(2,1001):
for x in [10,10000,10000000,10000000000]:
     res = li(x)- li(2)
     premierterme = res
     a = P.compte([x])
     print('(',x,') calcul par pgm -->',a[0])
     b = li(sqrt(x))-li(2)
     res = res-0.5*b
     print('pi(',x,') calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).-->', res)
     erreur =(res-a)/a
     print('erreur entre valeur par pgm et valeur par formule li -->', erreur[0])
     secondterme = 0
     for rho in range(len(zeros)-1):
          secondterme = secondterme+ei((0.5+1j*zeros[rho])*log(x))
                                   +ei((0.5-1j*zeros[rho])*log(x))
     res2 = premierterme-secondterme+troisiemeterme(x)-log(2)
     print(' formule Li(x)-sum Ei(x log(x)+integrale- log(2) = ',res2)
     erreur2 =(res2-a)/a
     print('erreur entre valeur par pgm et valeur par formule ei -->', erreur2[0])
tac = time.time()
print('resultat obtenu en',tac-tic,' s.')
```

Le résultat du programme pour x dans $[10, 10000, 10000000, 10000000000]$:

```
pix( 10 ) calcul par pgm --> 4
pi( 10 ) calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).--> 4.48909850785791
erreur entre valeur par pgm et valeur par formule li --> 0.122274626964477
formule Li(x)-sum Ei(x log(x)+integrale- log(2) = (4.28781070063084 + 0.0j)
erreur entre valeur par pgm et valeur par formule ei --> (0.0719526751577111 + 0.0j)

pix( 10000 ) calcul par pgm --> 1229
pi( 10000 ) calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).--> 1230.55156321729
erreur entre valeur par pgm et valeur par formule li --> 0.00126245990015449
formule Li(x)-sum Ei(x log(x)+integrale- log(2) = (1246.54633388521 + 0.0j)
erreur entre valeur par pgm et valeur par formule ei --> (0.0142769193533034 + 0.0j)

pix( 10000000 ) calcul par pgm --> 664579
pi( 10000000 ) calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).--> 664686.402063612
erreur entre valeur par pgm et valeur par formule li --> 0.000161609174547764
formule Li(x)-sum Ei(x log(x)+integrale- log(2) = (664828.337742295 + 0.0j)
erreur entre valeur par pgm et valeur par formule ei --> (0.000375181494292575 + 0.0j)

pix( 10000000000 ) calcul par pgm --> 455052511
pi( 10000000000 ) calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).--> 455050799.159541
erreur entre valeur par pgm et valeur par formule li --> -3.76185257284992e-6
formule Li(x)-sum Ei(x log(x)+integrale- log(2) = (455057512.125852 + 0.0j)
erreur entre valeur par pgm et valeur par formule ei --> (1.09902170208009e-5 + 0.0j)
resultat obtenu en 142.40949630737305 s.
```

En travaillant avec 2 millions de zéros de la fonction ζ au lieu de 1000, on obtient sensiblement les mêmes résultats, mais en utilisant beaucoup plus de temps d'exécution !

Le résultat du programme avec extentions du calcul à 2 millions de zéros de ζ :

```
pix( 10 ) calcul par pgm --> 4
pi( 10 ) calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).--> 4.48909850785791
erreur entre valeur par pgm et valeur par formule li --> 0.122274626964477
formule Li(x)-sum Ei(x log(x)+integrale- log(2) = (4.28817337409392 + 0.0j)
erreur entre valeur par pgm et valeur par formule ei --> (0.0720433435234802 + 0.0j)

pix( 10000 ) calcul par pgm --> 1229
pi( 10000 ) calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).--> 1230.55156321729
erreur entre valeur par pgm et valeur par formule li --> 0.00126245990015449
formule Li(x)-sum Ei(x log(x)+integrale- log(2) = (1246.053094085 + 0.0j)
erreur entre valeur par pgm et valeur par formule ei --> (0.0138755850976425 + 0.0j)

pix( 10000000 ) calcul par pgm --> 664579
pi( 10000000 ) calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).--> 664686.402063612
erreur entre valeur par pgm et valeur par formule li --> 0.000161609174547764
formule Li(x)-sum Ei(x log(x)+integrale- log(2) = (664826.091009064 + 0.0j)
erreur entre valeur par pgm et valeur par formule ei --> (0.000371800807825875 + 0.0j)

pix( 10000000000 ) calcul par pgm --> 455052511
pi( 10000000000 ) calcul par li(x)-li(2)-0.5*(li(sqrt(x))-li(2)).--> 455050799.159541
erreur entre valeur par pgm et valeur par formule li --> -3.76185257284992e-6
formule Li(x)-sum Ei(x log(x)+integrale- log(2) = (455057435.426343 + 0.0j)
erreur entre valeur par pgm et valeur par formule ei --> (1.08216661244684e-5 + 0.0j)
resultat obtenu en 897.2732815742493 s.
```