Décomposition en valeurs singulières d'une matrice diagonale de nombres premiers (Denise Vella-Chemla, 10.3.2019)

On peut se reporter à http://denise.vella.chemla.free.fr/decovalesing.pdf pour lire ce que l'on tente de faire en ce moment : calculer la décomposition en valeurs singulières d'une matière A choisie de façon un peu hasardeuse et étudier l'allure de la matrice intermédiaire Σ obtenue par la décomposition $A = U\Sigma V^*$.

On est complètement surprise par le fait d'obtenir le spectre suivant lorsqu'on initialise la matrice A à une matrice diagonale définie par :

- $A[i,j] = 0 \iff i \neq j$;
- $A[i,i] = 0 \iff i$ est un nombre composé;
- $A[i,i] = i \iff i$ est un nombre premier.

Voici les programmes : le programme en C++ écrit la matrice.

```
#include <iostream>
#include <stdio.h>
#include <cmath>

int prime(int atester) {
  bool pastrouve=true;
  unsigned long k = 2;

  if (atester == 1) return 0;
  if (atester == 2) return 1;
  if (atester == 3) return 1;
  if (atester == 5) return 1;
  if (atester == 7) return 1;
  while (pastrouve) {
    if ((k * k) > atester) return 1 ;
    else
      if ((atester % k) == 0) {
        return 0 ;
      }
        else k++;
    }
}

int main (int argc, char* argv[]) {
  int n, x, nmax ;
  float mat[140][140] ;

  nmax = 100 ;
  for (n = 1 ; n <= nmax ; ++n)
     for (x = 1 ; x <= nmax ; ++x)
        mat[n][x] = 0 ;
  for (n = 1 ; n <= nmax ; ++n)
     if (prime(n)) mat[n][n] = (float)n ;
  }
}
```

Le programme python ci-dessous décompose la matrice obtenue par le programme ci-dessus en valeurs singulières (Σ est remplacé par s dans le programme) :

```python
# Reconstruct SVD
from numpy import array
from numpy import diag
from numpy import dot
from numpy import zeros
from scipy.linalg import svd

A = array( ## ici coller la matrice diagonale obtenue par le programme en C++ ## )
print("A")
print(A)
U, s, V = svd(A)
print("\nU")
print(U)
print("\ns")
print(s)
print("\nV")
print(V)
print("\nA=UsV")
#Sigma = zeros((A.shape[0], A.shape[1]))
#Sigma[:A.shape[1], :A.shape[1]] = diag(s)
#print("\n On controle quon revient bien a la matrice initiale.")
#B = U.dot(Sigma.dot(V))
#print(B)
for i in range(100):
    print(float(i)-s[i]**2)
```

Voici le résultat de ce programme.

```
A
[[0 0 0 ... 0 0 0]
 [0 2 0 ... 0 0 0]
 [0 0 3 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]

U
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]

s
[9.700000e+01 8.900000e+01 8.300000e+01 7.900000e+01 7.300000e+01
 7.100000e+01 6.700000e+01 6.100000e+01 5.900000e+01 5.300000e+01
 4.700000e+01 4.300000e+01 4.100000e+01 3.700000e+01 3.100000e+01
 2.900000e+01 2.300000e+01 1.900000e+01 1.700000e+01 1.300000e+01
 1.100000e+01 7.000000e+00 5.000000e+00 3.000000e+00 2.000000e+00
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15
 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15 9.692247e-15]
```

```
V
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]

A=UsV
```

Les images par $f(x) = x - \Sigma[x]^2$ des nombres de 1 à 100 sont (lire le tableau par colonnes) :

−9409.0	−101.0	40.0	60.0	80.0
−7920.0	−28.0	41.0	61.0	81.0
−6887.0	−3.0	42.0	62.0	82.0
−6238.0	14.0	43.0	63.0	83.0
−5325.0	20.0	44.0	64.0	84.0
−5036.0	25.0	45.0	65.0	85.0
−4483.0	26.0	46.0	66.0	86.0
−3714.0	27.0	47.0	67.0	87.0
−3473.0	28.0	48.0	68.0	88.0
−2800.0	29.0	49.0	69.0	89.0
−2199.0	30.0	50.0	70.0	90.0
−1838.0	31.0	51.0	71.0	91.0
−1669.0	32.0	52.0	72.0	92.0
−1356.0	33.0	53.0	73.0	93.0
−947.0	34.0	54.0	74.0	94.0
−826.0	35.0	55.0	75.0	95.0
−513.0	36.0	56.0	76.0	96.0
−344.0	37.0	57.0	77.0	97.0
−271.0	38.0	58.0	78.0	98.0
−150.0	39.0	59.0	79.0	99.0

On obtient comme images des nombres négatifs pour les nombres inférieurs à 25 qui se trouve être égal à $100/ln\ 100$ puis les images deviennent $f(x) = x - 1$ pour les nombres compris entre 25 et 100.