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

On peut se reporter à cette première note ou à cette seconde note pour avoir une idée de ce que l'on tente de faire en ce moment : il s'agit de calculer la décomposition en valeurs singulières d'une matière A choisie de façon un peu hasardeuse et d'étudier l'allure de la matrice intermédiaire Σ obtenue par la décomposition $A = U\Sigma V^*$.

La matrice A est une matrice triangulaire basse définie par :

$$A[n,x] \ = \ \begin{cases} \dfrac{ln\ n}{2\pi} \text{ pour } x \text{ de 1 à } n \text{ inclus si } x|n \ ; \\ \\ 0 \text{ sinon.} \end{cases}$$

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

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

int prime(int atester)
{
  unsigned long diviseur=2;
  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[1441][1441] ;

  nmax = 1420 ;
  for (n = 1 ; n <= nmax ; ++n)
    for (x = 1 ; x <= nmax ; ++x)
      mat[n][x] = 0 ;
  for (n = 1 ; n <= nmax ; ++n)
    for (x = 1 ; x <= n ; ++x)
      if ((n%x) == 0) mat[n][x] = log(float(n))/(2.0*M_PI) ;
  for (n = 1 ; n <= nmax ; ++n)
    {
      std::cout << "[" ;
      for (x = 1 ; x <= nmax ; ++x)
        std::cout << mat[n][x] << "," ;
      std::cout << "]," ;
      std::cout << "\n" ;
    }
}
```

Des contraintes d'occupation mémoire obligent à utiliser une matrice de taille 1420×1420.

On rappelle que la matrice Σ contient sur sa diagonale les valeurs singulières de A, i.e. les racines carrées positives des valeurs propres de AA^* ou de A^*A (qui sont égales même si AA^* et A^*A ne le sont pas forcément).

Les carrés des $\Sigma[x]$ sont les valeurs propres de AA^* ou de A^*A.

Le programme python ci-dessous décompose la matrice, obtenue par le programme C++, en valeurs singulières (Σ est remplacé par s dans le programme python) puis il calcule les valeurs $561 - Sigma[i]^2$ (561 est la valeur approximative de la plus grande valeur propre, augmentée de 14, au hasard) :

```
# 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 obtenue par le programme en C++ ci-dessus ##)
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(1420):
    print(561.0-s[i]**2)
```

Voici le résultat de ce programme.

```
[[0.       0.       0.       ... 0.       0.       0.      ]
 [0.110318 0.110318 0.       ... 0.       0.       0.      ]
 [0.17485  0.       0.17485  ... 0.       0.       0.      ]
 ...
 [1.15499  1.15499  0.       ... 1.15499  0.       0.      ]
 [1.1551   0.       1.1551   ... 0.       1.1551   0.      ]
 [1.15521  1.15521  0.       ... 0.       0.       1.15521 ]]

U
[[-1.11022302e-16 -2.22044605e-16 -8.32667268e-16 ... -3.74640945e-14
  -6.55399315e-14 -1.00000000e+00]
 [-2.37741008e-03 -2.20525478e-04  2.39321467e-03 ... -2.98632712e-02
  -9.63851830e-01  8.56161866e-14]
 [-3.24211617e-03 -5.43586719e-03 -1.32043909e-03 ...  9.53958320e-01
  -7.86474610e-03 -2.08945830e-14]
 ...
 [-2.49213541e-02 -2.36216548e-03  2.52226092e-02 ...  1.68724033e-04
   1.67998280e-04 -5.89805982e-17]
 [-2.40657814e-02 -4.10170863e-02 -9.30135671e-03 ...  4.28189685e-04
   8.55092204e-05  1.52655666e-16]
 [-3.82290192e-02  2.60573330e-02  3.63728214e-02 ...  3.22972602e-05
   2.90778634e-04 -1.52655666e-16]]

s
[5.19519165e+01 2.33923756e+01 2.29849024e+01 ... 1.09106584e-02
 5.80333762e-03 4.39334124e-15]
```

```
V
[[-6.76167566e-01 -4.43423166e-01 -2.87138977e-01 ... -5.54049143e-04
  -5.35079087e-04 -8.50065758e-04]
 [-4.23779456e-01  3.77018141e-01 -3.03460160e-01 ... -1.16631058e-04
  -2.02539653e-03  1.28681636e-03]
 [ 3.74925019e-01  1.23704447e-01 -5.48503307e-01 ...  1.26743463e-03
  -4.67437144e-04  1.82808029e-03]
 ...
 [ 1.49022340e-02 -1.78557679e-02  4.46248656e-02 ...  1.78609360e-02
   4.53319943e-02  3.41960279e-03]
 [-1.72699861e-02 -3.34339569e-02  1.70089522e-02 ...  3.34352999e-02
   1.70198095e-02  5.78822753e-02]
 [-3.39422117e-02  3.39422117e-02  3.39422117e-02 ... -3.39422117e-02
   3.39422117e-02 -9.54297405e-14]]

A=UsV
```

Les images par $f(x) = 561 - \Sigma[x]^2$ des nombres de 1 à 100 sont (lire le tableau par colonnes) :

−2138.00163247	494.352578525	526.030438578	538.735890534	544.586861925
13.7967646102	496.423846665	526.962704161	538.906093642	544.639440943
32.6942613319	500.887531138	528.449732725	539.041910525	544.916130435
149.91021977	502.006046786	529.444517937	539.120948115	545.267733941
240.926080288	502.707132998	530.321019491	539.270944879	545.633447427
326.689323827	504.732573721	531.265305219	540.233068476	545.932498456
341.547037108	506.744921471	531.898239776	540.582091414	546.010740386
373.437083444	509.938685426	532.348022699	541.056424656	546.090943439
388.848448189	513.795276939	532.772537589	541.41614529	546.190883682
417.15295015	514.926930875	533.047292939	541.697715318	546.212261214
430.24116732	515.221874876	533.901683211	542.211203976	546.340190077
443.507958568	516.353247519	534.573267235	542.319979404	546.409623531
452.738552489	517.564578917	534.962065599	542.57018078	546.5786899
453.488661267	518.94473361	535.460457317	543.216188088	546.827693581
473.073059951	521.647161951	536.065685687	543.334098616	546.96084195
476.230907792	522.407058441	536.345286542	543.522994082	546.972997394
478.300876214	522.767562886	536.808390947	543.531950249	547.085008873
479.612324772	523.947602098	537.503330118	543.889064808	547.15630978
483.374316104	524.564302067	538.255446817	543.984084876	547.225608769
493.069420209	525.186869247	538.559008719	544.067879807	547.298583047

Les 100 dernières valeurs (pour x allant de 1410 à 1420) obtenues par le programme sont :

560.994042303
560.994228392
560.995469467
560.995857567
560.996124514
560.996425799
560.996510886
560.996835094
560.997738359
560.998028077
560.998281736
560.998537424
560.99868932
560.999219922
560.999231672
560.999605627
560.999714664
560.999880958
560.999966321
561.0

On est satisfait du fait que les valeurs croissent bien comme un logarithme.