Décompositions de matrices en valeurs singulières (Denise Vella-Chemla, 7.3.2019)

On présente ici quelques résultats obtenus en décomposant certaines matrices en valeurs singulières [1].

On a testé plusieurs matrices A, pour lesquelles les tests n'ont pas été probants (l'algorithme de décomposition en valeurs singulières renvoie pour une matrice A qui lui est fournie en entrée 3 matrices U, Σ et V^* telles que $A = U\Sigma V^*$) :

1) la matrice booléenne de divisibilité contenait en $A[i,j]$ le booléen 1 si i divise j et 0 sinon ;
2) la matrice booléenne "premier à" contenait en $A[i,j]$ le booléen 0 si i et j étaient premiers entre eux ($pgcd(i,j) = 1$) et 1 sinon ;
3) la matrice booléenne "décomposants de Goldbach" contenait en $A[i,j]$ le booléen 1 si i et $j-i$ étaient premiers et 0 sinon (elle ne contenait que les lignes correspondant aux nombres pairs n et les colonnes correspondant aux nombres impairs qui peuvent potentiellement décomposer additivement les nombres pairs indices des lignes) ;
4) la dernière matrice testée contenait la fraction $1/j$ en $A[i,j]$ lorsque $pgcd(i,j)$ est différent de 1 et 0 sinon.

Voici l'apparence de la dernière matrice tronquée à 10×10.

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0.25 & 0 & 0.166667 & 0 & 0.125 & 0 & 0.1 \\
0 & 0 & 0.333333 & 0 & 0 & 0.166667 & 0 & 0 & 0.111111 & 0 \\
0 & 0.5 & 0 & 0.25 & 0 & 0.166667 & 0 & 0.125 & 0 & 0.1 \\
0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0.1 \\
0 & 0.5 & 0.333333 & 0.25 & 0 & 0.166667 & 0 & 0.125 & 0.111111 & 0.1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.142857 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0.25 & 0 & 0.166667 & 0 & 0.125 & 0 & 0.1 \\
0 & 0 & 0.333333 & 0 & 0 & 0.166667 & 0 & 0 & 0.111111 & 0
\end{pmatrix}$$

Notre but serait d'obtenir des valeurs singulières dans le "style" du logarithme, c'est-à-dire que les parties entières des petites valeurs se répèteraient peu et augmenteraient assez rapidement tandis que celles des grandes valeurs se répèteraient davantage et l'écart entre 2 valeurs successives serait de plus en plus faible.

1. On a utilisé le programme fourni dans cette page https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/ . Le livre de référence où trouver une présentation de la SVD (abréviation anglaise) est *Matrix computations*, Johns Hopkins éditions, 4ème édition, G.H. Golub et C.F. Van Loan (1983). Le but de la décomposition en valeurs singulières est d'extraire les caractères dominants de l'information codée par une matrice M, c'est-à-dire d'obtenir une autre matrice qui contient en quelque sorte l'essence de l'information contenue dans M. La SVD est notamment utilisée pour débruiter des spectres (cf *Performance du SVD pour débruiter les spectres RMN et Raman*, Guillaume Laurent, William Woelffel, Virgile Barret-Vivin, Emmanuelle Gouillart, Christian Bonhomme, c2i-2016 : 7ème Colloque Interdisciplinaire en Instrumentation, Jan 2016, Saint-Nazaire, France.).

Voici les programmes [2] : le programme en C++ écrit la matrice.

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

int pgcd(int m, int n) {
  while (m  != 0) {
      int r ;

      r = n % m ; n = m ; m = r ;
  }
  return(n) ;
}

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)
    for (x = 1 ; x <= nmax ; ++x)
      if (pgcd(n,x) == 1) mat[n][x] = 0.0 ;
      else mat[n][x] = 1.0/(float)x ;
  for (n = 1 ; n <= nmax ; ++n)
    {
      std::cout << "[" ;
      for (x = 1 ; x <= nmax ; ++x)
        std::cout << mat[n][x] << "," ;
      std::cout << "]," ;
      std::cout << "\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
from numpy import array
from numpy import diag
from numpy import dot
from numpy import zeros
from scipy.linalg import svd

A = array( ##ici il faut coller la matrice ##
           ## obtenue par le programme 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*")
for i in range(100):
   print(s[i]**2)
```

2. à réécrire, le but est de rapidement satisfaire le souhait de “voir ce que ça donne”.

Voici les résultats de quelques programmes.

```
A
[[0.        0.        0.        ... 0.        0.        0.       ]
 [0.        0.5       0.        ... 0.0102041 0.        0.01     ]
 [0.        0.        0.333333  ... 0.        0.010101  0.       ]
 ...
 [0.        0.5       0.        ... 0.0102041 0.        0.01     ]
 [0.        0.        0.333333  ... 0.        0.010101  0.       ]
 [0.        0.5       0.        ... 0.0102041 0.        0.01     ]]

U
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [-1.32089477e-01  8.04762777e-02  3.68366580e-02 ... -4.60212122e-03
   5.52844163e-04  6.40499527e-04]
 [-3.49658245e-02 -1.96487394e-01  3.35452247e-02 ... -3.23711970e-03
  -1.21452459e-05  6.46622254e-03]
 ...
 [-1.33690720e-01  7.59852504e-02  2.81185431e-02 ...  1.73062952e-02
  -8.25613042e-02  8.36641413e-02]
 [-3.61404533e-02 -1.96628928e-01  3.16888927e-02 ...  3.19719470e-02
  -1.76012510e-01  2.03879401e-02]
 [-1.36344738e-01  6.87864238e-02 -1.82705303e-01 ...  6.34874962e-02
  -1.34902218e-01  5.25652046e-02]]

s
[4.65611391e+00 1.94936677e+00 9.33790933e-01 5.71790070e-01
 4.81439410e-01 2.93927295e-01 2.43090394e-01 2.13429409e-01
 1.44860809e-01 1.40061142e-01 1.35108704e-01 1.18800440e-01
 9.02263084e-02 8.02390309e-02 7.17259919e-02 6.34123028e-02
 5.86410266e-02 5.30018321e-02 4.31607748e-02 3.92038692e-02
 3.58469368e-02 3.51075312e-02 3.39330107e-02 3.32686737e-02
 3.25243339e-02 3.07325421e-02 2.93690945e-02 2.84151560e-02
 2.25946666e-02 1.93762024e-02 1.88679000e-02 1.69492000e-02
 1.63934000e-02 1.59040990e-02 1.52578836e-02 1.49254000e-02
 1.44892874e-02 1.40845000e-02 1.38715120e-02 1.36986000e-02
 1.26582000e-02 1.23193040e-02 1.20482000e-02 1.12360000e-02
 1.05183231e-02 1.03093000e-02 1.01818389e-02 9.07762826e-03
 8.89885398e-03 8.30783062e-03 8.11130666e-03 7.60308069e-03
 7.36229665e-03 7.07033010e-03 6.93997657e-03 6.67738363e-03
 6.49705085e-03 6.03641202e-03 5.71822874e-03 5.00493168e-03
 7.16854622e-16 5.20003341e-16 3.53270803e-16 3.53270803e-16
 3.53270803e-16 3.53270803e-16 3.53270803e-16 3.53270803e-16
 3.53270803e-16 3.53270803e-16 3.53270803e-16 3.53270803e-16
 3.53270803e-16 3.53270803e-16 3.53270803e-16 3.53270803e-16
 3.53270803e-16 3.53270803e-16 3.53270803e-16 3.53270803e-16
 3.53270803e-16 3.53270803e-16 3.53270803e-16 3.53270803e-16
 3.53270803e-16 3.53270803e-16 3.53270803e-16 3.53270803e-16
 3.53270803e-16 3.53270803e-16 1.88615847e-16 1.24449645e-16
 9.30644727e-17 4.19372065e-17 2.97735936e-17 1.56921106e-17
 1.45903823e-17 1.39463671e-17 1.10132647e-17 1.04494666e-17]

V*
[[ 0.00000000e+00 -7.48461738e-01 -2.19634386e-01 ... -1.55138572e-02
  -7.55676328e-03 -1.54023185e-02]
 [ 0.00000000e+00  2.95566924e-01 -8.24432046e-01 ...  3.81076411e-03
  -2.38595787e-02  2.40336935e-03]
 [ 0.00000000e+00  1.05568221e-01  1.27979748e-01 ... -6.85256944e-04
   2.07862919e-03 -2.39656320e-02]
 ...
 [ 0.00000000e+00  0.00000000e+00 -1.49050665e-01 ...  3.29510795e-02
  -9.49321653e-02 -1.40688520e-01]
 [ 0.00000000e+00  0.00000000e+00  4.65274363e-03 ... -2.63724869e-02
   2.37148263e-01  1.40517023e-01]
 [ 0.00000000e+00  0.00000000e+00 -2.41910608e-02 ...  1.47993123e-03
  -7.57626842e-02 -3.77435695e-02]]

A=UsV*
```

Voyons les carrés des éléments diagonaux de Σ dans un tableau (à lire par colonnes) :

21.6793967164	0.00128500287704	0.00016023002724	$5.13880549673e-31$	$1.24800260396e-31$
3.8000308122	0.00123253874694	0.000151765251357	$2.70403474271e-31$	$1.24800260396e-31$
0.871965506593	0.00115144921495	0.00014515912324	$1.24800260396e-31$	$1.24800260396e-31$
0.32694388399	0.00110680465294	0.000126247696	$1.24800260396e-31$	$1.24800260396e-31$
0.231783905087	0.00105783229614	0.000110635121457	$1.24800260396e-31$	$1.24800260396e-31$
0.0863932544963	0.000944489142309	0.00010628166649	$1.24800260396e-31$	$1.24800260396e-31$
0.059092939799	0.000862543711767	0.000103669843365	$1.24800260396e-31$	$1.24800260396e-31$
0.0455521125966	0.000807421088811	$8.24033347536e-05$	$1.24800260396e-31$	$1.24800260396e-31$
0.0209846541007	0.000510518956639	$7.91896020941e-05$	$1.24800260396e-31$	$1.24800260396e-31$
0.0196171235262	0.000375437221318	$6.90200496897e-05$	$1.24800260396e-31$	$1.24800260396e-31$
0.0182543619638	0.00035599765041	$6.57932956662e-05$	$1.24800260396e-31$	$3.55759375824e-32$
0.0141135446318	0.00028727538064	$5.7806836049e-05$	$1.24800260396e-31$	$1.54877142586e-32$
0.00814078671969	0.00026874356356	$5.42034119737e-05$	$1.24800260396e-31$	$8.66099608286e-33$
0.00643830208238	0.000252940365325	$4.99895677871e-05$	$1.24800260396e-31$	$1.75872929239e-33$
0.00514461791193	0.000232803011583	$4.8163274861e-05$	$1.24800260396e-31$	$8.86466876797e-34$
0.00402112014764	0.00022276756516	$4.45874520913e-05$	$1.24800260396e-31$	$2.46242336274e-34$
0.00343876999992	0.00020993944904	$4.2211669715e-05$	$1.24800260396e-31$	$2.128792558e-34$
0.0028091942088	0.00019837314025	$3.64382701353e-05$	$1.24800260396e-31$	$1.94501154613e-34$
0.00186285247845	0.000192418844822	$3.50493411279e-05$	$1.24800260396e-31$	$1.21292000214e-34$
0.00153694336395	0.00018765164196	$2.26981399349e-05$	$1.24800260396e-31$	$1.09191351983e-34$

Ce résultat ne correspondant pas à ce qu'on souhaite (écarts de plus en plus faibles mais valeurs décroissantes), on remplace les carrés des éléments diagonaux de la matrice Σ (correspondant à l'écriture des $s[i]^2$ dans le programme en python) par $25 - s[i]^2$) pour que les petits éléments soient moins répétés que les grands éléments. Il se trouve qu'on a utilisé une matrice finie de taille 100×100 et que 25 est approximativement la valeur de $100/ln\ 100$ [3].

Les images des valeurs singulières par cette modification sont :

3.32060328361	24.9987149971	24.99983977	25.0	25.0
21.1999691878	24.9987674613	24.9998482347	25.0	25.0
24.1280344934	24.9988485508	24.9998548409	25.0	25.0
24.673056116	24.9988931953	24.9998737523	25.0	25.0
24.7682160949	24.9989421677	24.9998893649	25.0	25.0
24.9136067455	24.9990555109	24.9998937183	25.0	25.0
24.9409070602	24.9991374563	24.9998963302	25.0	25.0
24.9544478874	24.9991925789	24.9999175967	25.0	25.0
24.9790153459	24.999489481	24.9999208104	25.0	25.0
24.9803828765	24.9996245628	24.99993098	25.0	25.0
24.981745638	24.9996440023	24.9999342067	25.0	25.0
24.9858864554	24.9997127246	24.9999421932	25.0	25.0
24.9918592133	24.9997312564	24.9999457966	25.0	25.0
24.9935616979	24.9997470596	24.9999500104	25.0	25.0
24.9948553821	24.999767197	24.9999518367	25.0	25.0
24.9959788799	24.9997772324	24.9999554125	25.0	25.0
24.99656123	24.9997900606	24.9999577883	25.0	25.0
24.9971908058	24.9998016269	24.9999635617	25.0	25.0
24.9981371475	24.9998075812	24.9999673019	25.0	25.0
24.9984630566	24.9998123484	24.9999749507	25.0	25.0

3. Cela corrobore l'idée que $ln\ n$ code l'information associée au nombre n.

Si on remplit la première colonne de la matrice de 1 (après tout, 1 divise tous les nombres), le résultat est moins satisfaisant :

2885.88743285	2999.99849938	2999.99982527	2999.99999526
2991.60741185	2999.99874998	2999.99984697	3000.0
2996.87846951	2999.99877043	2999.99984829	3000.0
2999.21330917	2999.99886724	2999.99986524	3000.0
2999.7011722	2999.9989062	2999.99988532	...
2999.79145017	2999.99895047	2999.99988951	
2999.91681115	2999.99910746	2999.9998961	
2999.94540223	2999.99913746	2999.99991428	
2999.95483379	2999.99922954	2999.99991908	
2999.97993775	2999.99948977	2999.99992611	
2999.98053749	2999.99962465	2999.99993332	
2999.98208001	2999.99965615	2999.99994218	
2999.98604257	2999.99971921	2999.99994313	
2999.99211913	2999.99974383	2999.99994769	
2999.99364057	2999.99974805	2999.99995042	
2999.99485926	2999.99976723	2999.99995187	
2999.99602603	2999.99978472	2999.99995542	
2999.99676973	2999.99979009	2999.99995867	
2999.99721198	2999.99980595	2999.99996619	
2999.99814259	2999.99980808	2999.99996787	

On souhaite également tester 3 matrices auxquelles on s'est intéressé à plusieurs reprises :

5) la matrice diagonale des $\exp \frac{2\pi}{p}$ avec p premier (on code le fait pour un nombre d'être composé par un élément diagonal nul) ;
6) la matrice triangulaire basse des $\cos \frac{2\pi no}{t}$;
7) une matrice similaire à celle utilisée ci-dessus mais qui verrait ses coefficients fractionnaires $1/k$ remplacés par des $1/ln\ k$.

Voici les valeurs obtenues comme images par la fonction $f(x) = 25 - x$ des $s[i]^2$ pour la matrice dans le cas 5) (c'est une matrice diagonale qui contient des coefficients 0 pour les indices qui sont des nombres composés et qui contient des $e^{\frac{2\pi}{p}}$ pour les indices qui sont des nombres premiers) :

```
  A
[[ 0.       0.       0.      ...  0.       0.       0.     ]
 [ 0.      23.1407   0.      ...  0.       0.       0.     ]
 [ 0.       0.       8.12053 ...  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.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]
```

```
s
[2.31407000e+01 8.12053000e+00 3.51359000e+00 2.45370000e+00
 1.77039000e+00 1.62145000e+00 1.44715000e+00 1.39193000e+00
 1.31414000e+00 1.24192000e+00 1.22468000e+00 1.18509000e+00
 1.16561000e+00 1.15734000e+00 1.14303000e+00 1.12586000e+00
 1.11237000e+00 1.10849000e+00 1.09832000e+00 1.09253000e+00
 1.08988000e+00 1.08278000e+00 1.07864000e+00 1.07315000e+00
 1.06692000e+00 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15
 2.31222042e-15 2.31222042e-15 2.31222042e-15 2.31222042e-15]

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

−510.49199649	23.8121615856	25.0	25.0	25.0
−40.9430074809	23.8121615856	25.0	25.0	25.0
12.6546853119	23.8275874716	25.0	25.0	25.0
18.97935631	23.8365357504	25.0	25.0	25.0
21.8657192479	23.8483490775	25.0	25.0	25.0
22.3708998975	23.8616817136	25.0	25.0	25.0
22.9057568775	25.0	25.0	25.0	25.0
23.0625308751	25.0	25.0	25.0	25.0
23.2730360604	25.0	25.0	25.0	25.0
23.4576347136	25.0	25.0	25.0	25.0
23.5001588976	25.0	25.0	25.0	25.0
23.5955616919	25.0	25.0	25.0	25.0
23.6413533279	25.0	25.0	25.0	25.0
23.6605641244	25.0	25.0	25.0	25.0
23.6934824191	25.0	25.0	25.0	25.0
23.7324392604	25.0	25.0	25.0	25.0
23.7626329831	25.0	25.0	25.0	25.0
23.7712499199	25.0	25.0	25.0	25.0
23.7936931776	25.0	25.0	25.0	25.0
23.8063781991	25.0	25.0	25.0	25.0

Les résultats sont peu probants mais on obtient l'allure générale souhaitée : croissance et écarts entre deux nombres successifs allant s'amenuisant.

Les résultats sont aussi peu probants dans le cas 6) (matrice contenant des cosinus), on ne les reporte pas ici.

Dans le cas 4) de la matrice contenant des fractions comprises entre 0 et 1 pour les coefficients-noeuds de divisibilité, on obtient cet ensemble de valeurs sur la diagonale de Σ :

On rappelle l'allure de A dans le cas 3 :

```
A
[[1.        0.        0.        ... 0.        0.        0.       ]
 [1.        0.5       0.        ... 0.        0.        0.       ]
 [1.        0.        0.333333  ... 0.        0.        0.       ]
 ...
 [1.        0.5       0.        ... 0.0102041 0.        0.       ]
 [1.        0.        0.333333  ... 0.        0.010101  0.       ]
 [1.        0.5       0.        ... 0.        0.        0.01     ]]
```

108.674196502	0.00544989557862	0.00070154742262	0.000173796294339	$5.83548107524e-05$
6.37298298289	0.00493690542223	0.000674223550113	0.000156799078991	$5.7057463612e-05$
2.6313657741	0.0045987448796	0.000516204896639	0.000152790854719	$5.56007207607e-05$
0.790103378471	0.00403496458407	0.000464568247524	0.000141107212868	$5.20215590349e-05$
0.694992900526	0.0035188383274	0.000443570230473	0.000134541456887	$5.01516458977e-05$
0.26478712676	0.00278928685558	0.000404310189158	0.000132709451977	$4.67938830844e-05$
0.156211680215	0.00266398835853	0.000377332187717	0.000122297480475	$4.23923682868e-05$
0.0997187232614	0.00234967138721	0.000343045441381	0.000114474406405	$4.13880537755e-05$
0.0961606311889	0.00213991461893	0.000283145252026	0.000109830981444	$3.71099979134e-05$
0.0732673040176	0.0017509125305	0.000279703565144	0.000100589506544	$2.9167962694e-05$
0.0431835467644	0.00147994062137	0.000258312606659	$9.62424917471e-05$	$2.83785129084e-05$
0.0384593923818	0.0014636420175	0.00025506622578	$9.15174824484e-05$	$2.59672256912e-05$
0.0189708063317	0.00140542049113	0.00025060466223	$8.47712494842e-05$	$2.39457024115e-05$
0.0173481942063	0.00122058997984	0.000232917051133	$8.20546933454e-05$	$2.23150556148e-05$
0.0158902951871	0.00113845425252	0.000230384857779	$7.6353894481e-05$	$1.96574180229e-05$
0.0150803889736	0.00106674760612	0.000214903368888	$7.50016919415e-05$	$1.85404122388e-05$
0.0124047888128	0.00103596646136	0.000213529253679	$6.69506935326e-05$	$1.56248441348e-05$
0.0110721870025	0.000902278713479	0.000193575428772	$6.66349532198e-05$	$9.66809601276e-06$
0.00742389179676	0.000816317993325	0.000188458582255	$6.25382901749e-05$	$8.30007852048e-06$
0.0065713832573	0.000747999183379	0.000176250380598	$6.07565020569e-05$	$3.74036676728e-06$

On teste enfin l'utilisation de logarithmes dans les matrices (choix 7) ci-dessus) et les résultats sont plus conformes à nos attentes, même si les valeurs ne croissent pas assez rapidement.

Si on remplit la matrice par ces instructions :

```
for (n = 1 ; n <= nmax ; ++n)
    for (x = 1 ; x <= nmax ; ++x)
      if (pgcd(n,x) == 1) mat[n][x] = 0.0 ;
      else mat[n][x] = 1.0/log((float)x) ;
```

on obtient les valeurs suivantes :

−340.242100729	24.8262430882	24.9487864996	25.0	25.0
−32.9246718823	24.8316836653	24.9503668438	25.0	25.0
6.86863939688	24.8344723583	24.9522171004	25.0	25.0
16.2046556135	24.8407229417	24.9572008064	25.0	25.0
19.1024435697	24.8566646035	24.9678820754	25.0	25.0
21.6454659163	24.8775942229	24.9779736917	25.0	25.0
22.7153080672	24.8943091605	24.9789149881	25.0	25.0
22.9549748887	24.9032099609	24.9796843567	25.0	25.0
23.7751969843	24.9208201266	24.98076991	25.0	25.0
23.9021021233	24.9365609994	24.9829513981	25.0	25.0
23.9586122828	24.9398543995	24.9855631096	25.0	25.0
24.1157845746	24.940826032	24.9866367966	25.0	25.0
24.3776181934	24.9434368911	24.988599772	25.0	25.0
24.5212135021	24.9442507526	24.9890129852	25.0	25.0
24.5846452856	24.9446520334	24.9900005896	25.0	25.0
24.588005638	24.9449656552	24.9907539043	25.0	25.0
24.6322843577	24.9455846086	24.9911241054	25.0	25.0
24.7257218098	24.9456760444	24.9924839875	25.0	25.0
24.7691421779	24.947622185	24.9929701764	25.0	25.0
24.8168690809	24.9486198275	24.995261237	25.0	25.0

Si on remplit la matrice par ces instructions :

```
for (n = 1 ; n <= nmax ; ++n)
  for (x = 1 ; x <= nmax ; ++x)
    if ((n%x) == 0) mat[n][x] = 1.0/log((float)x) ;
```

on obtient les valeurs suivantes :

−10000000043.8	24.6753930433	24.9088639774	24.9566688735	24.9831513577
−32.6859009407	24.7011864118	24.9171041991	24.9605319801	24.9838171359
3.67895525934	24.7096788938	24.9224975527	24.9638082774	24.9843949814
17.326865157	24.7262349709	24.9307607898	24.966603928	24.9856747293
17.6319034623	24.7742597573	24.937243029	24.9679254797	24.9865216083
21.0390946697	24.780618833	24.9399367996	24.9713034216	24.9871382626
22.9342403634	24.8088764612	24.9402832193	24.9724298888	24.9878668791
23.1459963855	24.8180603728	24.9418196297	24.9728342474	24.9890896505
23.2631881628	24.820230044	24.9424042863	24.9738779192	24.9908592712
23.3207372372	24.8268940873	24.9439873005	24.9754361531	24.9920328012
23.8179791045	24.8338485077	24.9440982491	24.9760992002	24.9923586381
24.0875149449	24.8475764374	24.9446956735	24.9771711954	24.9925642087
24.2908156796	24.8520231185	24.9451496545	24.9776185429	24.9927293587
24.3359412848	24.8561925801	24.9453422072	24.978048135	24.9933533639
24.4043345131	24.8644732539	24.945506709	24.9789155519	24.9948330173
24.4917013581	24.8764346086	24.9467976191	24.979027889	24.9952422296
24.5188030564	24.8810703698	24.9482855416	24.9797029738	24.9958943006
24.5370699049	24.8947195631	24.949828304	24.981319458	24.9973388302
24.5912772705	24.9009438149	24.9517075316	24.9816265066	24.9979935231
24.6353758213	24.9049357469	24.9521417396	24.9821938674	24.9991433106

Si on remplit la matrice par ces instructions :

```
for (n = 1 ; n <= nmax ; ++n)
   for (x = 2 ; x <= nmax ; ++x)
      if (((n%x) == 0) && (pow(x,vp(n,x)) == n)) // puissance "pure"
        mat[n][x] = 1.0/log((float)x) ;
```

avec la fonction $vp(n, p)$ (pour valuation p-adique) définie ainsi :

```
int vp(int m, int p)
{
  if ((m%p) != 0) return 0 ;
  else return vp(m/p, p)+1 ;
}
```

on obtient les valeurs suivantes :

11.4782557783	24.898284293	24.9317803063	24.943833844	24.9503668438
21.4307171924	24.900990343	24.9325403271	24.9442204247	24.9506129382
24.176624281	24.9057952028	24.9332718109	24.9445976679	24.950854874
24.2215022649	24.9099387896	24.9346573931	24.9449656552	24.9510922352
24.3356643273	24.9118064433	24.9353137078	24.9453249341	24.9513250506
24.4367510818	24.9135557678	24.9359479828	24.9456760444	24.951553789
24.5977332191	24.9151984832	24.9365609994	24.9460190538	24.9517789144
24.652418068	24.918204	24.9371545239	24.9463540285	24.9520000101
24.7610588334	24.9195835179	24.9377282912	24.9466814955	24.9522171004
24.826084311	24.9208894372	24.9382850194	24.9470019746	24.9524306452
24.8380500951	24.9226826697	24.9388239138	24.94731552	24.9526406651
24.8480006034	24.9233053442	24.9393466542	24.947622185	24.9542476861
24.8564173601	24.9244255915	24.9398543995	24.947922478	24.9547473696
24.8636404056	24.9254939302	24.9403473109	24.9485045905	24.95988736
24.8754220621	24.9265129228	24.940826032	24.9487864996	24.963488203
24.8803006074	24.9274866659	24.9412911946	24.9490631211	24.9690438208
24.8846562179	24.9284191379	24.9417439022	24.9493340417	24.9760850348
24.8885722191	24.9293115479	24.9426129229	24.94959975	24.9778914044
24.8921146853	24.930168238	24.9430304255	24.9498602814	24.9878829985
24.8953380448	24.9309902862	24.9434368911	24.9501161176	25.0

La première de ces trois propositions d'utilisation du logarithme népérien (noté *log* en C++) semble la plus intéressante, la croissance des valeurs étant plus progressive que dans les deux derniers cas.