Leçon 2 : SVD

L'imagerie numérique a longtemps été un exemple d'application de la SVD. Maintenant, il existe des moyens plus élaborés et plus efficaces. Reste que la SVD est un outils intéressant pour le transfert d'image. Une photo en noir et blanc peut être assimilée à une matrice A dont les éléments correspondent à des niveaux de gris pour les différents pixels qui constituent l'image. Chaque $a_{i,j}$ donne le niveau de gris du pixel (i, j) de l'image : $_{ai,j} \in [0, 1]$ avec par exemple $a_{i,j} = 0$ pour un pixel (i, j) blanc et $a_{i,j} = 1$ pour un pixel $a_{i,j}$ noir.

En mathématiques, le procédé d'algèbre linéaire de décomposition en valeurs singulières (ou SVD, de l'anglais singular value decomposition) d'une matrice est un outil important de factorisation des matrices rectangulaires réelles ou complexes. Ses applications s'étendent du traitement du signal aux statistiques, en passant par la météorologie.

Le théorème spectral énonce qu'une matrice normale peut être diagonalisée par une base orthonormée de vecteurs propres. On peut voir la décomposition en valeurs singulières comme une généralisation du théorème spectral à des matrices arbitraires, qui ne sont pas nécessairement carrées.

Soit A une matrice $m * n$ dont les coefficients appartiennent au corps K, où K est le corps des réels ou celui des complexes. Alors il existe une factorisation de la forme : $A = U\Sigma V^*$ avec U une matrice unitaire $m * m$ sur K, Σ une matrice $m * n$ dont les coefficients diagonaux sont des réels positifs ou nuls et tous les autres sont nuls, et V^* est la matrice adjointe à V, matrice unitaire $n * n$ sur K. On appelle cette factorisation la `décomposition en valeurs singulières` de A.

- La matrice V contient un ensemble de vecteurs de base Base orthonormale|orthonormés de K^n, dits `d'entrée` ou `d'analyse` ;
- La matrice U contient un ensemble de vecteurs de base orthonormés de K^m, dits `de sortie` ;
- La matrice Σ contient dans ses coefficients diagonaux les valeurs singulières de la matrice A, elles correspondent aux racines des valeurs propres de A^*A.
- Une convention courante est de ranger les valeurs σ_i par ordre décroissant. Alors, la matrice Σ est déterminée de façon unique par A (mais U et V ne le sont pas).

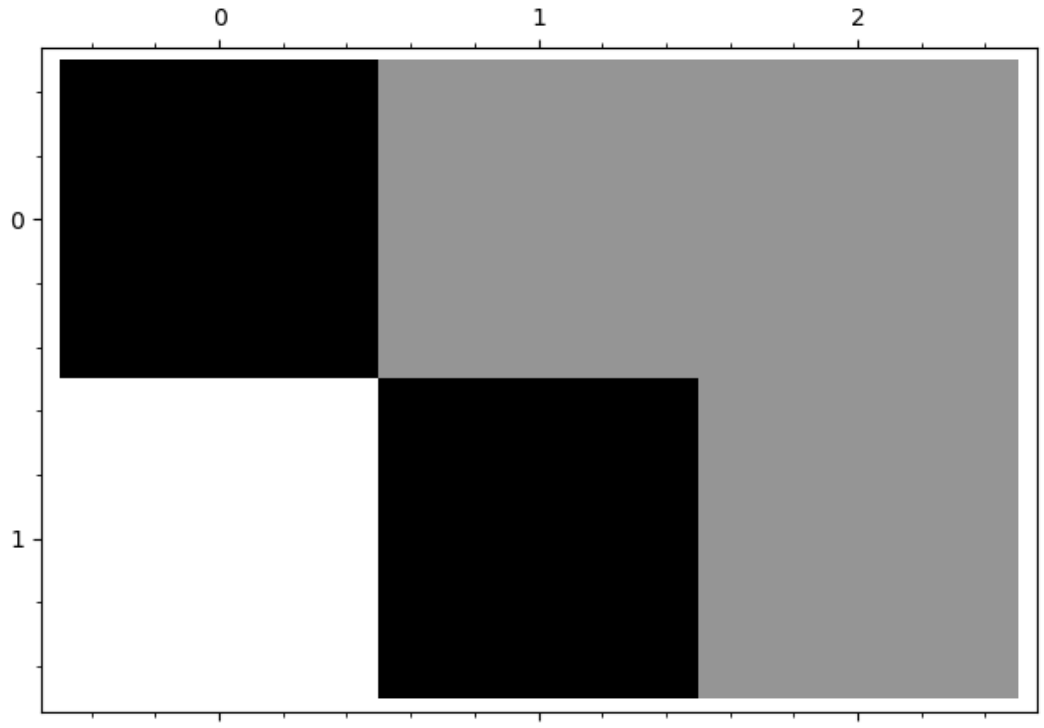
Existence

Soit A une matrice complexe $m * n$. Alors A^*A est positive semi-définie, donc hermitienne. D'après le théorème spectral, il existe une matrice unitaire carrée de côté n, notée V, telle que : $V^*A^*AV = \begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix}$, où Λ est diagonale, définie positive et de même rang r que A. Pour obtenir U, on considère AA^*.

La décomposition en valeurs singulières est très générale, dans le sens où elle s'applique à toute matrice rectangulaire m × n. La décomposition en valeurs propres, en revanche, ne fonctionne que pour certaines matrices carrées. Néanmoins, quand elles sont toutes les deux définies, elles sont liées.

In [1]:
```
A1 = matrix(2,3, [3,1,1, -1,3,1])
matrix_plot(A1)
```

Out[1]:



Question 1 : Calculez une base de vecteurs propres de $A.A^t$ et de $A^t.A$.

In [2]:
```
A = A1
B = A*transpose(A)
show(B)
```

$$\begin{pmatrix} 11 & 1 \\ 1 & 11 \end{pmatrix}$$

In [3]:
```
b = B.eigenspaces_left()
b[0], b[1]
```

Out[3]:
```
((12, Vector space of degree 2 and dimension 1 over Rational Field
  User basis matrix:
  [1 1]), (10, Vector space of degree 2 and dimension 1 over Rational Field
  User basis matrix:
  [ 1 -1]))
```

In [4]:
```
b = B.eigenspaces_right()
b
```

Out[4]:
```
[
(12, Vector space of degree 2 and dimension 1 over Rational Field
User basis matrix:
[1 1]),
(10, Vector space of degree 2 and dimension 1 over Rational Field
User basis matrix:
[ 1 -1])
]
```

Question 2 : Notez U (respectivement V) la matrice de la base orthonormée formée des vecteurs propres de $A.A^t$ (respectivement de $A.A$). Notez D la matrice des valeurs singulières de A. Trouvez une manière rapide pour générer U,V et D.

In [5]:
```
norm(vector([1,1]))
```

Out[5]: $\sqrt{2}$

In [6]:
```
U = matrix(RR,2,2, [1/sqrt(2), 1/sqrt(2), 1/sqrt(2), -1/sqrt(2)])
show(U)
```

$$\begin{pmatrix} 0.707106781186548 & 0.707106781186548 \\ 0.707106781186548 & -0.707106781186548 \end{pmatrix}$$

In [7]:
```
D = matrix(RR,2,3,[sqrt(12),0,0, 0,sqrt(10),0])
show(D)
```

$$\begin{pmatrix} 3.46410161513775 & 0.000000000000000 & 0.000000000000000 \\ 0.000000000000000 & 3.16227766016838 & 0.000000000000000 \end{pmatrix}$$

In [8]:
```
C = transpose(A)*A
show(C)
```

$$\begin{pmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{pmatrix}$$

In [9]:
```
C.eigenspaces_left()
```

Out[9]:
```
[
(12, Vector space of degree 3 and dimension 1 over Rational Field
User basis matrix:
[1 2 1]),
(10, Vector space of degree 3 and dimension 1 over Rational Field
User basis matrix:
[   1 -1/2    0]),
(0, Vector space of degree 3 and dimension 1 over Rational Field
User basis matrix:
[ 1  2 -5])
]
```

In [10]:
```
V = matrix(RR, 3,3, [1/sqrt(6),2/sqrt(5),1/sqrt(30), 2/sqrt(6),-1/sqrt(5),2/sqr
t(30), 1/sqrt(6),0,-5/sqrt(30)])
show(V)
```

$$\begin{pmatrix} 0.408248290463863 & 0.894427190999916 & 0.182574185835055 \\ 0.816496580927726 & -0.447213595499958 & 0.365148371670111 \\ 0.408248290463863 & 0.000000000000000 & -0.912870929175277 \end{pmatrix}$$

In [11]:
```
(U * D) * transpose(V)
```

Out[11]:
```
[ 3.00000000000000  1.00000000000000  1.00000000000000]
[-1.00000000000000  3.00000000000000  1.00000000000000]
```

In [12]:
```
import numpy
u, s, v = numpy.linalg.svd(A1)
```

In [13]:
```
u, s, v
```

Out[13]:
```
(array([[-0.70710678, -0.70710678],
        [-0.70710678,  0.70710678]]),
 array([3.46410162, 3.16227766]),
 array([[-4.08248290e-01, -8.16496581e-01, -4.08248290e-01],
        [-8.94427191e-01,  4.47213595e-01,  5.26260748e-16],
        [-1.82574186e-01, -3.65148372e-01,  9.12870929e-01]]))
```

In [14]:
```
#### Question 3 : Importez l'image coffee.png et stockez-là dans une matrice `A_image`.
```

In [15]:
```
from IPython.core.display import Image
Image(url='https://upload.wikimedia.org/wikipedia/commons/e/eb/Stockschwaemmchen.jpg')
```

Out[15]:

In [16]:
```
# Utiliser n'importe quel image sous format .png
import numpy
import pylab
A_image = 1-numpy.mean(pylab.imread('Stockschwaemmchen.png'),2)
A_image.shape
```

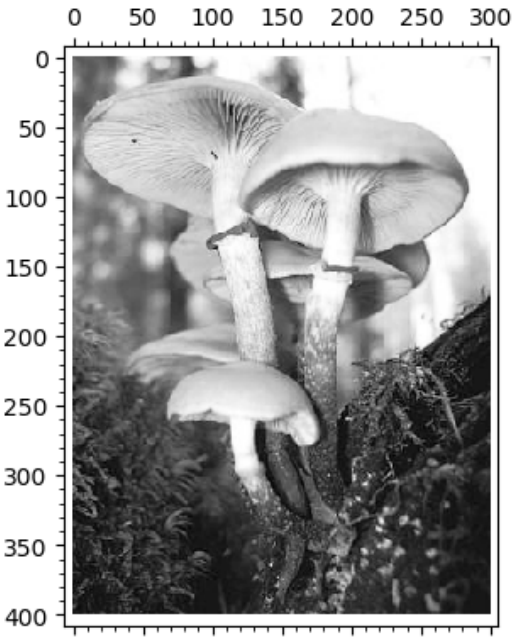
Out[16]:
```
(400, 300)
```

In [17]:
```
B = pylab.imread('Stockschwaemmchen.png')
B.shape
```

Out[17]:
```
(400, 300, 3)
```

In [18]:
```
matrix_plot(A_image)
```

Out[18]:



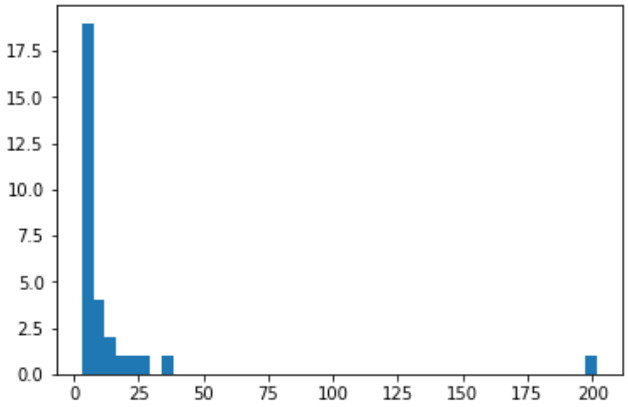
Question 4 : Tracez sur un même graphique la matrice `A_image` ainsi que la matrice `A_approx` qui donne une approximation de l'image en fonction d'un nombre de valeurs singulières réduit.

In [19]:
```
U,s,V = numpy.linalg.svd(A_image)
```

In [20]:
```
D = numpy.zeros(A_image.shape)
D[:len(s),:len(s)] = numpy.diag(s)
```

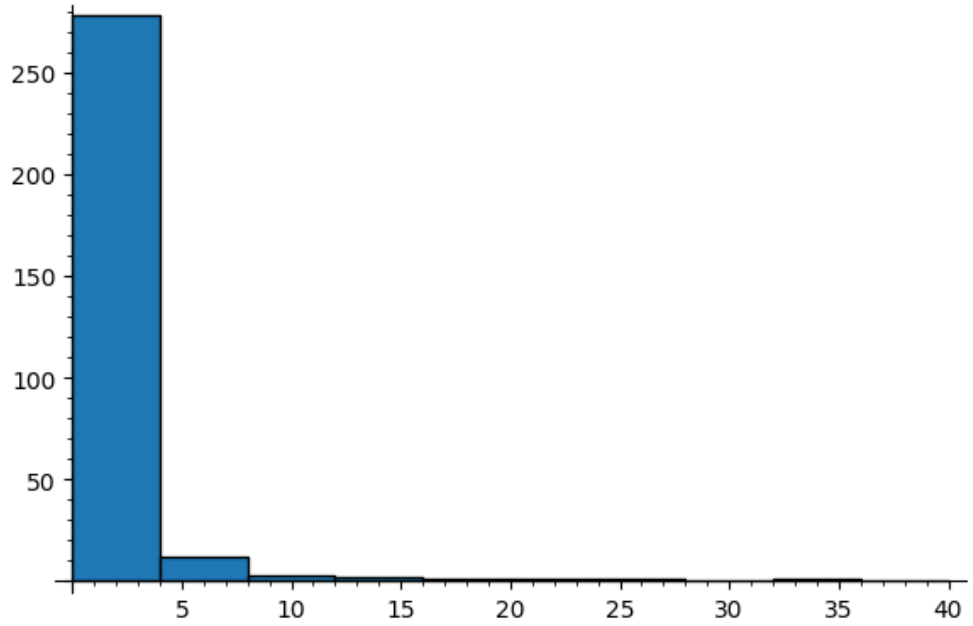
In [21]:
```
import matplotlib.pyplot as plt
plt.hist(s[0:30], bins='auto')
```

Out[21]:
```
(array([19.,  4.,  2.,  1.,  1.,  1.,  0.,  1.,  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.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  1.]),
 array([  3.0962427,   7.514891 ,  11.933539 ,  16.352188 ,  20.770836 ,
         25.189486 ,  29.608133 ,  34.026783 ,  38.44543  ,  42.86408  ,
         47.282726 ,  51.701378 ,  56.120026 ,  60.538673 ,  64.95732  ,
         69.37597  ,  73.79462  ,  78.213264 ,  82.63191  ,  87.05057  ,
         91.469215 ,  95.88786  , 100.30651  , 104.72516  , 109.14381  ,
        113.562454 , 117.9811   , 122.39975  , 126.8184   , 131.23705  ,
        135.6557   , 140.07434  , 144.493    , 148.91164  , 153.33029  ,
        157.74895  , 162.16759  , 166.58624  , 171.00488  , 175.42354  ,
        179.84218  , 184.26083  , 188.67947  , 193.09813  , 197.51677  ,
        201.93542  ], dtype=float32),
 <a list of 45 Patch objects>)
```



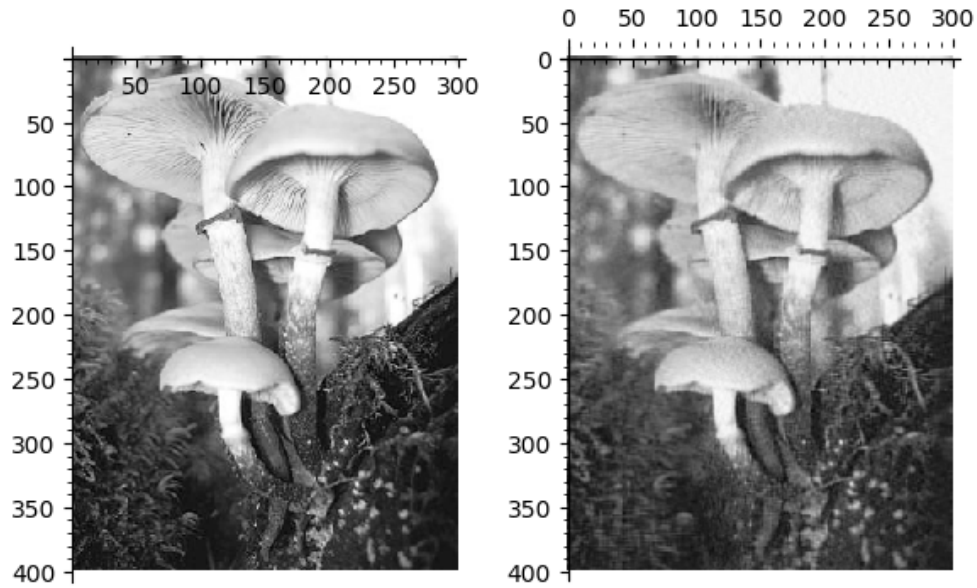
In [22]:
```
from sage.plot.histogram import Histogram
H = histogram(s, bins=10, normed=False, range=[0,40])
show(H)
```

```
/Applications/sage-8.6/local/lib/python2.7/site-packages/sage/plot/histogram.p
y:315: DeprecationWarning: the 'normed' option is deprecated. Use 'density' in
stead.
See https://trac.sagemath.org/25260 for details.
  g.add_primitive(Histogram(datalist, options=options))
/Applications/sage-8.6/local/lib/python2.7/site-packages/matplotlib/axes/_axe
s.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has been replace
d by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/Applications/sage-8.6/local/lib/python2.7/site-packages/sage/plot/histogram.p
y:116: DeprecationWarning: Passing normed=False is deprecated, and has no effe
ct. Consider passing the density argument instead.
  ydata, xdata = numpy.histogram(self.datalist, **opt)
```



In [23]:
```
i = 50
A_approx = numpy.dot(numpy.dot(U[:,:i], D[:i,:i]), V[:i,:])
graphics_array([matrix_plot(A_image), matrix_plot(A_approx)])
```

Out[23]:



Plutôt que de considérer 300*400 pixels pour stocker cette image, l'idée d'utiliser SVD est de n'utiliser que 50*(300+400)
Ce qui fait un gain en taille mémoire. Ici, c'est 70% de la taille mémoire de gagnée.

In [24]:
```
(50*(300+400))/(300*400).n()
```

Out[24]: 0.291666666666667