Analyse Numérique - TP n° 2
Reconstruction d'images par FFT et SVD

Groupe C06

Christian Ingouff
Pierre-Alexandre Tyndal

EISTI

2013/2014
Semestre 2

Table des matières

Introduction

Le travail présenté dans ce rapport[2] traite de la reconstruction approchée d'images grâce à deux méthodes : la transformée rapide de Fourier et la décomposition en valeurs singulières. L'intérêt de ces méthodes est d'effectuer une compression de données sur l'image, tout en prenant en compte les erreurs qui pourraient être induites par la reconstitution.

Ce rapport présente l'état de l'art des deux méthodes ainsi qu'une application de celles-ci grâce au logiciel *Scilab*. Les résultats obtenus sont étudiés et seront conclus par un récapitulatif critique des méthodes, présentant leurs avantages et leurs inconvénients.

Partie 1

Description mathématique

1.1 Transformée rapide de Fourier (FFT)

La transformée de Fourier[3] est basée sur la découverte que toute fonction peut être exprimée par une fonction périodique dépendant du temps s(t). Dans notre cas les signaux sont numériques et nous ne discuterons que le cas de la transformée de Fourier discrète sur un intervalle de temps fini correspondant à N échantillons. Elle s'écrira donc :

$$s(t) = \sum_{k=0}^{N-1} (c_k \exp(\frac{k}{N} 2ki\pi))$$

Si le signal à étudier est échantillonné à intervalles régulièrement espacés et si l'erreur statistique est la même pour tous les échantillons, on peut utiliser la méthode du produit scalaire pour obtenir les coefficients de Fourier. Le signal s'écrit $S_r \in \mathbb{C}$ avec $r \in [0; N[$ dans l'espace direct et les composantes de Fourier s'écrivent :

$$c_k = \frac{1}{N} \sum_{r=0}^{N-1} (S_r \exp(\frac{k}{N} 2ki\pi))$$

On obtient ainsi une représentation spectrale discrète du signal échantillonné s(n). La transformée de Fourier discrète faite en appliquant directement la première équation nécessite N multiplications pour calculer un coefficient de Fourier c_k. Comme il y en a également N, le temps de calcul de la transformée de Fourier serait de N^2 multiplications.

La transformation de Fourier rapide consiste à définir N de telle sorte qu'il soit puissance de deux ainsi, il est possible de réduire le nombre d'opérations à $N \log_2 N$. La transformée de Fourier nous permet donc d'obtenir une représentation spectrale discrète du signal échantillonné c_k.

La compression d'un signal sous cette forme viendrait à ne prendre qu'une partie des composantes de Fourier. Il s'agirait en l'occurrence des signaux de modules proches de 0, que l'on pourrait approximer pour augmenter le taux de compression.

1.2 Décomposition en valeurs singulières (SVD)

Le procédé de décomposition en valeurs singulières (ou SVD)[3] d'une matrice est un outil de factorisation des matrices rectangulaires réelles ou complexes. Le théoreme affirme : Soit M une matrice $m \times n$ dont les coefficients appartiennent au corps K, où $K = \mathbb{R}$ ou $K = \mathbb{C}$. Alors il existe une factorisation de la forme :

$$M = U\Sigma V^*$$

avec U une matrice unitaire $m \times m$ sur K, Σ une matrice $m \times n$ 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 $n \times n$ sur K.

On considère une image en nuances de gris contenant m lignes et n colonnes. Cette matrice sera donc la matrice considérée pour ce procédé.

La procédure pour obtenir cette décomposition peut s'apparenter à la décomposition en valeurs propres pour les matrices carrées : le procédé consiste donc à "diagonaliser" la matrice de données. Ceci est possible algorithmiquement par la méthode de l'échelonnage par exemple.

La compression d'un signal sous cette forme est possible en prenant une SVD "tronquée", de telle manière à ce que :

$$\tilde{M} = U_t\Sigma_t V_t^*$$

en ne gardant uniquement les t vecteurs colonnes de U et les t vecteurs lignes de V^* correspondants aux t plus grandes valeurs singulières Σ_t.

Partie 2

Réalisation informatique

2.1 Description

La réalisation informatique se base sur une image importée au préalable dans le logiciel *Scilab*[1]. Elle est stockée dans une matrice A de dimensions $m \times n$, m étant le nombre de colonnes (longueur) et n étant le nombre de lignes (hauteur).

FIGURE 2.1 – Image d'exemple originale

Cette image est compressée à l'aide des deux méthodes décrites précédemment : la transformée rapide de Fourier (FFT) et la décomposition en valeurs singulières (SVD). Ces méthodes sont disponibles sur *Scilab* sous les fonctions *fft* et *svd*.

La compression consiste à ne garder qu'un nombre réduit de valeurs, en considérant les valeurs assez petites égales à 0.

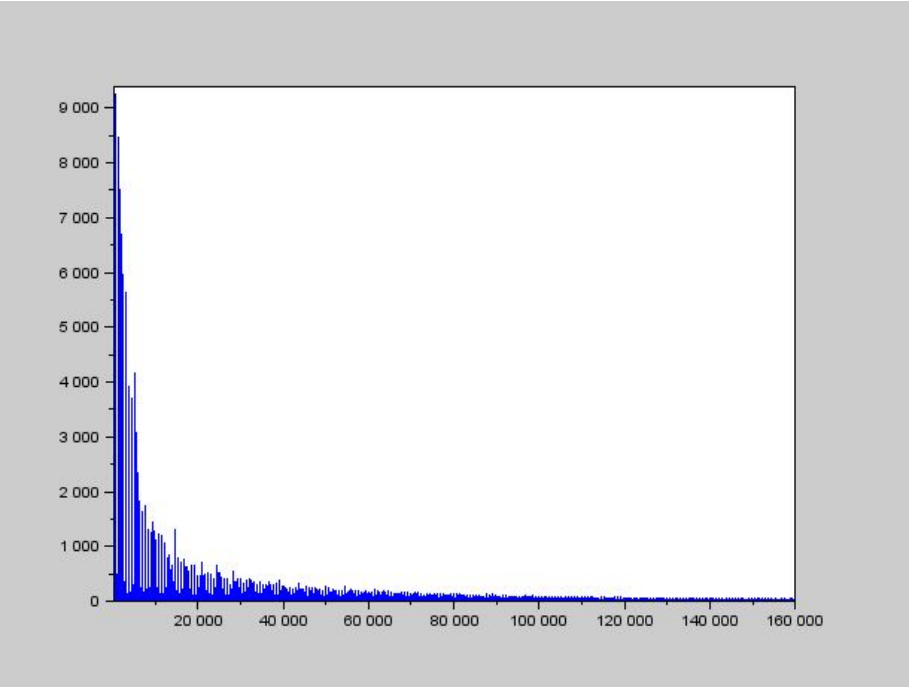


FIGURE 2.2 – Spectre de l'image

Le résultat de la FFT donne le tableau des résultats B, dont la deuxième moitié est le conjugué de la première, donc nous pouvons déjà éliminer la deuxième moitié du tableau. Le graphique ci-dessus représente le spectre de l'image, c'est-à-dire le module des valeurs restantes du tableau des résultats.

On constate que les valeurs décrivent un signal pseudo-périodique qui diminue d'intensité le long du tableau de résultats. Par conséquent, on peut ne prendre qu'une partie des premières valeurs pour représenter notre image et ainsi diminuer le nombre de valeurs stockées pour celle-ci dans un tableau réduit B'.

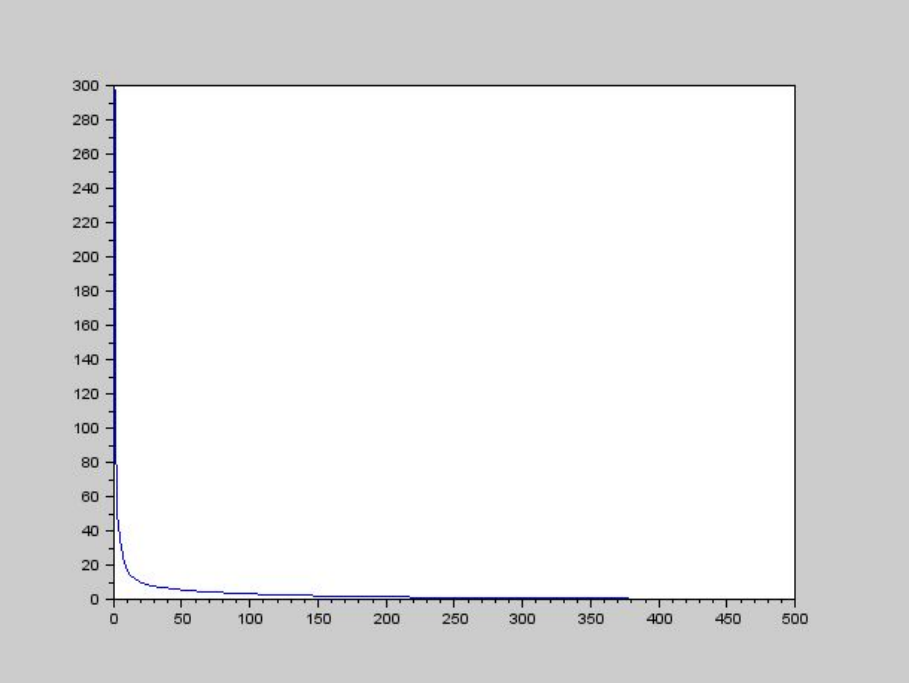


FIGURE 2.3 – Représentation des valeurs singulières de l'image

La décomposition en valeurs singulières exprime la matrice de l'image A à l'aide de trois matrices : U et V les matrices carrées unitaires et S une matrice diagonale telle que :

$$A = U \times S \times V^T$$

Le graphique ci-dessus trace le tableau des valeurs singulières, c'est-à-dire les valeurs de la diagonale de S. On remarque que les valeurs sont décroissantes, donc on peut ne prendre qu'une partie des premières valeurs singulières. Ainsi, si on ne prend qu'une partie des valeurs singulières, on peut également supprimer des colonnes de U et de V car elles ne rentreront plus en jeu dans le calcul de la reconstitution avec la matrice diagonale S' réduite. En résultent les matrices U' et V' telles que :

$$A_{\text{reconstituée}} = U' \times S' \times V'^T$$

L'image compressée est ensuite reconstituée à partir de la compression opérée par le processus inverse : la transformée rapide inverse de Fourier à partir de B' ou la recomposition à partir de valeurs singulières $U' \times S' \times V'^T$. Les images resultantes sont étudiées et comparées pour effectuer un bilan d'ensemble sur les méthodes.

2.2 Algorithmes

L'image de base est d'abord importée à l'aide de la lecture de fichiers implémentée sur *Scilab* sous une matrice $m \times n$ `img`.

2.2.1 Transformée rapide de Fourier

On vectorise d'abord la matrice de l'image dans un vecteur de longueur $m \times n$.
C'est sur ce vecteur que se déroulent les opérations suivantes :

```
Fonction compresserFFT(img : matrice, pourcentage : entier) : tableau
// img : Matrice (m * n) de l'image de départ
// pourcentage : Pourcentage de données à garder (de 0 à 100)
// Retourne un tableau de complexes de dimension (m * n) * pourcentage/100
var
   vect_img : tableau de réels // Vectorisation de img
   l : entier // Longueur de vect_img
Début
   vect_img <- vectoriser(img)
   l <- longueur(vect_img)
   B <- fft(vect_img) // Fonction de Scilab
   Retourner les l/2 * (pourcentage / 100) premières valeurs de B
Fin
```

On obtient ainsi un vecteur réduit `vect_fft` à partir duquel on peut reconstituer une image avec les opérations suivantes :

```
Fonction reconstituerFFT(vect_fft : tableau, m, n : entier) : matrice
// vect_fft : tableau de complexes, résultat de compresserFFT
// m : nombre de lignes de l'image (hauteur)
// n : nombres de colonnes de l'image (longueur)
var
   vect_img : tableau de réels // Reconstitution de img
   l : entier // Longueur de vect_fft
Début
   l <- longueur(vect_fft)
   vect_img.longueur <- m * n
   vect_img <- initialiser les valeurs à 0
   l premières valeurs de vect_img <- vect_fft
   l dernières valeurs de vect_img <- renverser(conjugué(vect_fft))
   vect_img <- fft_inverse(vect_img) // Fonction de Scilab
   vect_img <- module(vect_img) // Eliminer les parties imaginaires
   Retourner matrice(vect_img, m, n)
Fin
```

2.2.2 Décomposition en valeurs singulières

On récupère une décomposition en valeurs singulières réduite à partir de la compression :

```
Fonction compresserSVD(img : matrice, pourcentage : entier) :
                                                (matrice, tableau, matrice)
// img : Matrice (m * n) de l'image de départ
// pourcentage : Pourcentage de données à garder (de 0 à 100)
// Retourne la décomposition réduite sous la forme suivante :
//   1. matrice unitaire réduite (hauteur)
//   2. vecteur réduit des valeurs singulières
//   3. matrice unitaire réduite (longueur)
var
   U : matrice unitaire (hauteur)
   S : matrice diagonale des valeurs singulières
   V : matrice unitaire (longueur)
   nb_val : entier, nombre de valeurs singulières à garder
Début
   (U, S, V) <- svd(img) // Fonction de Scilab
   nb_val <- min(m, n) * (pourcentage / 100)
   U <- nb_val premières colonnes de U
   S <- nb_val premières valeurs de la diagonale de S
   V <- nb_val premières colonnes de V
   Retourner (U, S, V)
Fin
```

A partir de (U, S, V), on peut reconstituer l'image avec la fonction suivante :

```
Fonction reconstituerSVD(U, V : matrice, S : tableau) : matrice
// U : matrice unitaire réduite (hauteur)
// S : vecteur réduit des valeurs singulières
// V : matrice unitaire réduite (longueur)
Début
   Retourner (U * S * transposée(V))
Fin
```

Partie 3

Etude des résultats

Nous appliquons les transformations sur une image de dimensions

$$m = 500 \text{ et } n = 600.$$

L'image initiale est par conséquent stockée dans

$$m \times n = 320000 \text{ valeurs.}$$

Le pourcentage définit la portion de valeurs initiales obtenues à garder. Les valeurs stockées désigne le nombre de valeurs prises par les résultats de la transformation. Le taux de compression est défini ainsi :

$$t_c = 1 - \frac{\text{nombre de valeurs stockées}}{m \times n}$$

La déviation absolue moyenne est définie ainsi :

$$D = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} |\text{image}_{i,j} - \text{image reconstruite}_{i,j}|$$

Le taux de déviation est défini ainsi :

$$t_d = \frac{D}{\text{moyenne pixels image}}$$

3.1 Transformée rapide de Fourier

3.1.1 Phase de calcul

Pourcentage	Valeurs stockées	Taux compression	Résultat
100%	160000	50%	
90%	144000	55%	
80%	128000	60%	
70%	112000	65%	
60%	96000	70%	

Pourcentage	Valeurs stockées	Taux compression	Résultat
50%	80000	75%	
40%	64000	80%	
30%	48000	85%	
20%	32000	90%	
10%	16000	95%	

Pourcentage	Déviation absolue moyenne	Taux déviation
100%	0,0001553	0,03%
90%	0,0061938	1,20%
80%	0,0091299	1,77%
70%	0,0121186	2,35%
60%	0,0156235	3,03%
50%	0,0200216	3,89%
40%	0,0258087	5,01%
30%	0,0333469	6,47%
20%	0,0442955	8,60%
10%	0,0629601	12,22%

3.1.2 Interprétation des résultats

La détérioration de l'image se constate par la présence de floutage et de taches noires plus ou moins marquée selon le pourcentage utilisé. Le flou n'est pas distinguable à l'oeil nu de 100% jusqu'à 40% et les taches noires ne se présentent que par groupes isolés de quelques pixels jusqu'à 20%.

Nous considérons alors les images de 100% jusqu'à 40% comme satisfaisantes dans leur reconstitution. Les images de pourcentages inférieurs sont considérées comme détériorées.

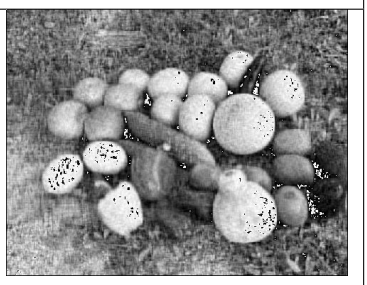
L'étude des taux de déviation nous indique effectivement qu'avec des pourcentages décroissants, la déviation augmente. La déviation dénote les taches noires et le floutage de l'image.

Les taux de compression observés sont satisfaisants : avec 40% de pourcentage, nous arrivons à réduire de 80% le nombre de valeurs stockées pour une image, tout en pouvant la reconstituer aisément.

3.2 Décomposition en éléments simples

3.2.1 Phase de calcul

Pourcentage	Valeurs stockées	Taux compression	Résultat
100%	570500	-78,3%	
90%	513450	-60,5%	
80%	456400	-42,6%	
70%	399350	-24,8%	
60%	342300	-7,0%	

Pourcentage	Valeurs stockées	Taux compression	Résultat
50%	285250	10,9%	
40%	228200	28,7%	
30%	171150	46,5%	
20%	114100	64,3%	
10%	57050	82,1%	

Pourcentage	Déviation absolue moyenne	Taux déviation
100%	0,0000000	0%
90%	0,0015120	0,29%
80%	0,0032328	0,63%
70%	0,0054753	1,06%
60%	0,0083676	1,62%
50%	0,0121204	2,35%
40%	0,0171018	3,32%
30%	0,0240402	4,67%
20%	0,0345255	6,70%
10%	0,0523532	10,16%

3.2.2 Interprétation des résultats

La détérioration de l'image se constate notamment par la présente de taches blanches ou noires : les taches blanches se remarquent à partir de 90% par la présence de pixels blancs parsemés dans les coins sombres, tandis que des pixels noirs apparaissent à partir de 70% dans les coins clairs. A partir de 20%, les pixels commencent à se regrouper et à 10%, on observe un flou sur le restant de l'image.

Jusqu'à 20%, la qualité de l'image, en excluant la présence de taches blanches et noires, est satisfaisante, sans présence de flou observable contrairement à la FFT. Cependant, la présence de ces taches nuit beaucoup à la constitution de l'image.

Une solution proposable pourrait être de rendre les taches blanches noires et les taches noires blanches, mais il faudrait faire la différence entre les taches et les vrais pixels de l'image.

L'étude des taux de déviation nous indique effectivement qu'avec des pourcentages décroissants, la déviation augmente. La déviation dénote surtout les taches noires et blanches : le floutage amoindri place les taux de déviation pour la SVD en dessous de ceux pour la FFT.

La compression n'est rentable qu'à partir de 50% : au dessus, le nombre de valeurs stockées est supérieur au nombre de valeurs stockées pour l'image initiale. L'intérêt de cette méthode pour la compression d'images est donc limitée, car en plus des valeurs singulières, il faut aussi stocker les matrices unitaires carrées afin de pouvoir reconstituer l'image.

Conclusion

Nous avons vu deux méthodes de compression de données, chacune possédant ses intérêts et limites.

La méthode de la transformée rapide de Fourier permet ainsi de compresser des données efficacement, avec un bon taux de compression, en sacrifiant la qualité du rendu de l'image reconstruite (présence de flou).

D'un autre abord, la méthode de la décomposition en valeurs singulières permet une reconstitution plus minutieuse des données de l'image, mais ne couvre pas forcément l'ensemble des pixels de l'image. De plus, son taux de compression par rapport à la qualité du rendu est moins optimal qu'avec la FFT.

Il existe d'autres méthodes pour compresser une image en tolérant les pertes, qui utilisent également le concept de couleurs plus ou moins fréquentes. Notons par exemple la transformée en cosinus discrète, utilisée dans le format *JPEG*.

En conclusion, toutes ces méthodes utilisent une propriété de l'image : certaines couleurs, qui vont apparaitre comme des valeurs dans notre matrice de données, apparaissent plus fréquemment que les autres. Chacune de ces méthodes procède cependant différemment pour distinguer ces fréquences.

Ces méthodes de compression se confrontent à la compression sans pertes, par exemple avec le format d'image *PNG*. Toutefois, la contrainte de la non-perte de données s'instaurant, les taux de compression avec la compression sans pertes sont limités.

Références

[1] Ce programme a été conçu dans Scilab 5.4.1. Les graphiques proposés proviennent de la fonction "plot" de Scilab.
http ://www.scilab.org/

[2] Ce rapport a été rédigé en LaTeX
http ://www.latex-project.org/

[3] Les notions mathématiques introduites dans ce rapport proviennent de cours fournis à l'EISTI.
http ://arel.eisti.fr/
http ://sifoci.eisti.fr/

Annexes

Annexe A : Organigramme du programme

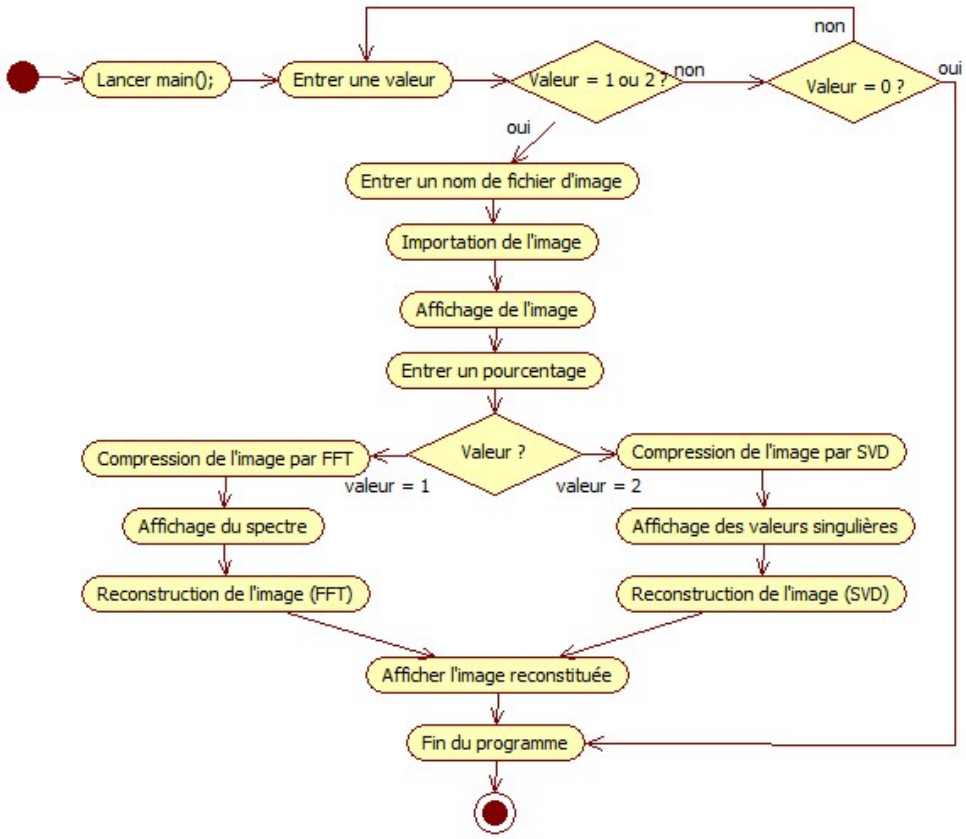


FIGURE 3.1 – Organigramme du programme Scilab

Annexe B : Guide d'utilisation du programme

Exécution du programme

Vous trouverez joint à ce rapport, dans le dossier `src`, le fichier `code.sce` qui peut être executé sous Scilab. Pour ceci, il faut accéder depuis la console Scilab au dossier contenant `code.sce` (par exemple : `cd ananu-tp2/src/;`), puis exécuter avec `exec("code.sce");`.

Le code ainsi executé, il est possible soit d'appeler la fonction `main();` pour faire appel à un invite de commandes interactif, soit de faire appel aux fonctions suivantes dans l'ordre :

- `img = getImage(nom_fichier);`
- `test = doFFT(img, pourcentage);`
  ou `test = doSVD(img, pourcentage);`
- `result = getFromFFT(test, size(img));`
  ou `result = getFromSVD(test, size(img));`

Utilisation du programme interactif

Le programme `main();` lancé propose à l'utilisateur plusieurs options :
- Démonstration de la FFT (option 1)
- Démonstration de la SVD (option 2)
- Quitter le programme (option 0)

L'utilisateur choisit son option en entrant un entier au clavier et en validant avec `Entrée`.

L'utilisateur est invité pour les deux méthodes à entrer le nom de l'image pour qu'elle puisse être importée. Une fois l'image importée, l'utilisateur est invité à préciser le pourcentage de valeurs à garder. Ces données sont entrées au clavier et validées avec `Entrée`.

Utilisation des fonctions individuelles

`img = getImage(nom_fichier)`
Importe l'image contenue dans `nom_fichier` dans la matrice de réels `img`.

`test = doFFT(img, pourcentage)`
Compresse l'image contenue dans `img` dans le tableau conteneur `test` à l'aide de la méthode par FFT.

`test = doSVD(img, pourcentage)`
Compresse l'image contenue dans `img` dans la matrice conteneuse `test` à l'aide de la méthode par SVD. Cette matrice contient dans l'ordre des colonnes :

- Colonne 1 : Valeurs singulières
- Colonnes 2 à $m + 1$: Matrice unitaire (hauteur)
- Colonnes $m + 2$ à $m + n + 1$: Matrice unitaire (longueur)

`result = getFromFFT(test, size(img))`
Reconstitue une image à partir d'un tableau issu de la compression par FFT.

`result = getFromSVD(test, size(img))`
Reconstitue une image à partir d'une matrice issue de la compression par SVD.