TP Matlab 3

Sur la décomposition en valeurs singulières, les moindres carrés et la compression d'image.

3.1 Instructions.

Le sujet du tp est la SVD pour singular value decomposition. La première partie présente un algorithme de calcul, la seconde propose de faire le lien avec la méthode des moindres carrés. Enfin, la dernière partie montre un nouvel aspect des multiples avantages de la SVD.

La dernière section est importante : elle donne un certain nombre d'astuces en matlab. S'y reporter devrait simplifier quelques problèmes potentiels.

3.2 Au boulot !

3.2.1 SVD

Soit M une matrice de $\mathcal{M}_{n,m}$.

Reparlons de vecteurs propres et de valeurs propres

Dans cette partie du TP, posons $n = 200$ et $m = 2$. On fabriquera la matrice M comme suit :

```
M = zeros(n,2);
ind = 1;
while ind<=size(M,1),
    point = 2*(0.5-rand(1,2));
    if norm(point)<1,
        M(ind,:) = point;
        ind = ind+1;
    end
end

M = M*P;
```

avec P :

$$P = \begin{pmatrix} -1.5 & 3.2 \\ 1.2 & -0.4 \end{pmatrix}$$

Calculer les valeurs propres λ_i et vecteurs propres v_i de la matrice $M' \times M$ – avec `eig` par exemple. On pose $S = \text{diag}\left(\sqrt{\lambda_i}\right)$, et V la matrice des vecteurs propres.

Calculer $U = M \times V' \times S^{-1}$.

Vérifier ensuite que U, S et V correspondent bien au retour de la fonction matlab `svd` !

Tracer ensuite tous les points de M – cela devrait faire une ellipsoïde – et superposer les droites définies par les colonnes de V et le point 0. Penser à utiliser la commande

```
axis equal
```

pour que les axes soient bien normés ! Conclusions ?

Résolution de système hyperstatique et SVD

La SVD permet aussi de calculer la solution de système linéaire à grande dimension, c'est-à-dire trouver la solution $\boldsymbol{x}$ du système :

$$M\boldsymbol{x} = \boldsymbol{b}$$

avec M une matrice de donnée de $\mathcal{M}_{n,m}$, et $n >> m$.
Expliquer pourquoi en faisant le lien avec la résolution par la méthode des moindres carrés. En utilisant les données du TP précédent, calculer, par SVD, les coefficients du polynôme de degré 2 approximant les données. Faire la comparaison avec les résultats du TP précédent.

Compression d'image et SVD

Télécharger l'image en noir et blanc à l'adresse suivante :
`http://www.limsi.fr/~gueniat/teaching/black_and_white.jpg`
Importer là – par exemple sous le nom M – sous matlab avec la commande `imread`.

Réenregistrer la matrice résultante en double, avec la commande bien nommée `double`.

Faire une SVD de M, récupérant les matrices U, S et V.

Faire une boucle affichant le résultat du produit $U(:,i) \times S(1:i,1:i) \times V(:,1:i)'$. Pour l'affichage, utiliser la fonction `pcolor`.

Conclusions sur la compression d'image en utilisant la SVD ?

Annexes sur Matlab

Inversion de matrice

Pour inverser une matrice, on peut utiliser la fonction de matlab `inv`, ou encore l'opérateur /.

Plot et axes

La fonction `plot` gère les axes automatiquement pour les adapter "au mieux" aux données. Pour contraindre les axes, on doit utiliser l'argument `axis` :

```
plot(x,y,'.')
axis equal
```

Importation d'image

Il faut commencer par importer l'image avec la commande `imread`. Ensuite, il faut transformer l'image en double (pour pouvoir faire des calculs dessus) et récupérer l'intensité des couleurs, ce qui se fait grâce à la somme. Matlab voit les images comme des matrices : pour l'avoir dans le "bon" sens, il faut donc la renverser. C'est pourquoi on utilisera la commande `flipud`.

```
M = imread('le chemin/black_and_white.jpg');
M = double(sum(M,3));
M = flipud(M);
```

Utilisation de pcolor

```
myim2 = u(:,1:i)*s(1:i,1:i)*v(:,1:i)';
pcolor(myim2)
shading interp
colormap('bone')
pause(.1)
```

Utilisation diag

En pratique, `diag` réagit de deux façons différentes suivant l'argument :
- si l'argument est un vecteur, `diag` renvoit une matrice diagonale, dont la diagonale est le vecteur donné en argument.
- Si l'argument est une matrice, `diag` renvoit alors la diagonale.