Pense-bête Matlab

1 Le plus important : la commande `help` (ou `doc`)

Grâce à la commande `help NomDeLaFonction`, il est possible d'obtenir à l'ecran des explications sur une fonction et son utilisation.

Exemple :

```
>> help sum

  SUM Sum of elements.
    For vectors, SUM(X) is the sum of the elements of X. For
    matrices, SUM(X) is a row vector with the sum over each
    column. For N-D arrays, SUM(X) operates along the first
    non-singleton dimension.

    SUM(X,DIM) sums along the dimension DIM.

    Example: If X = [0 1 2
                     3 4 5]
    then sum(X,1) is [3 5 7] and sum(X,2) is [ 3 12];

    See also PROD, CUMSUM, DIFF.
```

La commande `doc` affiche l'aide au format HTML dans le "Help Broswer".

2 Graphisme

2.1 Général

- `figure` permet d'ouvrir une nouvelle fenêtre "figure".
- `plot(x,y)` trace la courbe correspondant à l'ordonnée $y(i)$ pour l'abscisse $x(i)$
- `surf(M)` affiche la surface correspondant à la matrice `M` (le point de coordonnées (i, j) est à la hauteur $M(i, j)$)
- `subplot` permet de gérer le nombre de graphes crées sur une même figure
- `axis` permet de modifier la longueur des axes et par conséquent permet de ne visualiser qu'une portion d'un vecteur de données. Elle peut être utilisée de différentes manières (voir `help axis`)
  - `axis([ xmin xmax ymin ymax])`
  - `axis('square')` l'espace graphique prend une forme carré.
  - `axis('image')` respecte les ratios originaux.

2.2 Pour les images

2.2.1 Lecture et écriture

- `imread` lecture d'un fichier image. exple : `L = imread('lena.jpg');`. **Attention :** renvoie une image de `uint8`, à convertir en `double` pour la plupart des opérations (voir TP1)
- `imwrite` écriture d'un fichier image à un format donné. exple : `imwrite(L, 'lena_copie.bmp');`

2.2.2 Affichage

La fonction `imshow(i)` permet d'afficher l'image `I` dans une fenêtre "figure". Pour afficher une image, on doit d'abord choisir une dynamique. Voilà les principaux comportements de `imshow` en fonction du type de l'image et des paramètres passés :

- Pour une image contenant des `uint8` : `imshow` considère que la valeur 0 correspond à l'intensité la plus faible (du noir si l'image est en niveau de gris) et que la valeur 255 correspond à l'intensité la plus élevée (du blanc si l'image est en niveau de gris)
- Pour une image contenant des `double` ou `single` : par défaut `imshow` considère que 0 = noir et 1 = blanc, on peut préciser un intervalle de valeur en appelant `imshow(I,[low high])`. Enfin `imshow(I,[])` correspond à `imshow(I,[min(I(:)) max(I(:))])`.

3 Vecteurs et matrices

Pour définir une matrice sous Matlab, il faut respecter les conventions suivantes :
- séparer les éléments d'une même ligne par un blanc
- utiliser le symbole (;) pour indiquer la fin d'une ligne
- définir le début et la fin de la matrice par des crochets [ ]

Exemple :

```
>> A = [1  2;3  4]

A =
     1     2
     3     4
```

3.1 Initialisation et propriétés

- `M = zeros(n,p)` crée une matrice de taille $n \times p$ dont tous les éléments sont nuls.
- `M = ones(n,p)` crée une matrice de taille $n \times p$ dont tous les éléments sont égaux à 1.
- `M = rand(n,p)` crée une matrice aléatoire de taille $n \times p$.
- `length(v)` renvoie la longueur du vecteur `v`.

- `size(M)` renvoie un tableau indiquant les dimensions de la matrice `M`.
- `size(M,1)` (resp. `size(M,2)`) renvoie la taille de la première (resp. deuxième) dimension de la matrice `M`.

3.2 Manipulation

- `M(2,3)` renvoie la valeur de l'élément de la 2ème ligne et de la 3ème colonne de M.
- `M(i,:)` renvoie la i-ème ligne de M.
- `M(:,j)` renvoie la j-ème colonne de M.
- concaténation de deux matrices :
  - `B=[A C]` : concaténation horizontale, la matrice C est ajoutée à droite de la matrice A.
  - `B=[A;L]` : concaténation verticale, la matrice L est ajoutée au bas de la matrice A.

3.3 Opérations

- `M'` renvoie la transposé de `M`
- les opérations sur les matrices s'écrivent directement à l'aide des opérateurs `+`, `-`, `*`, etc. *exemple :*`M = A*B` est la **multiplication matricielle** de A par B
- pour effectuer une **multiplication terme à terme** on utilise l'opérateur `.*`. *exemple :*`M = A.*B`
- `M(:)` transforme une matrice à n lignes et p colonnes en un vecteur de taille $n * p$
- `max(V)` (`min(V)`) renvoient la valeur maximale (minimale) d'un vecteur `V`
- `mean(V)` renvoie la valeur moyenne d'un vecteur `V`

4 Fonctions

4.1 Définition d'une fonction

La syntaxe générale pour définir une fonction est la suivante :

```
function [sortie_1,sortie_2, ...]=nom_fonction(entrée_1,entrée_2, ...)
```

Exemple de définition d'une fonction :

```
function [somme,produit] = calcule_somme_produit(a,b)

somme = a+b;
produit = a*b;
```

Exemple d'appel de cette fonction :

```
[x,y] = calcule_somme_produit(1,2)
x =
     3
y =
```

```
2
```

4.2 Programmation

4.2.1 Boucles

exemple de boucle `For` *:*

```
s = 0;
for k = 1:2:11
  s = s+k;
end
% calcule s = 1 + 3 + 5 + 7 + 9 + 11

% même calcul mais plus adapté à matlab :
k = 1:2:11;
s = sum(k)
```

exemple de boucle `While` *:*

```
s = 1;
while ( s<1000 )
  s = s*2;
end
% à la sortie de la boucle : s = 1024
```

Important !
Matlab est un langage interprété, il faut éviter au maximum les boucles. En particulier pour les opérations vectorielles ou matricielles un gain de temps considérable peut être obtenu en préférant les fonctions et opérateurs Matlab à une écriture explicite via des boucles.

Exemple : pour trouver la différence maximale entre le vecteur x et le vecteur y définis par `x=rand(1,100000); y=rand(1,100000);` on vérifiera que `maxdif=max(x-y)` donne un résultat immédiatement tandis que `for i=1:100000; dif(i)=x(i)-y(i); end; maxdif=max(dif)` est beaucoup plus lent.

4.2.2 Structures conditionnelles

if(condition)...else...end

```
i = rand(); % renvoie un nombre aléatoire compris en 0 et 1
if (i<.5)
  disp('i est inférieur à 1/2')
else
  disp('i est supérieur à 1/2')
end
```

Il existe aussi `if (condition1) ... elseif(condition2) ... else ... end` et `switch ... case ... otherwise ... end`.