Introduction à Matlab

A. Perrut

Les commandes données ici correspondent à l'environnement de Matlab pour Windows, mais la plupart fonctionnent encore sous Linux.

Les instructions (ou commandes) peuvent être placées à l'invite >> dans la fenêtre de commandes ou dans un fichier avec l'extension `.m` (M-File) qu'on ouvre en cliquant sur `New/Open` dans le menu déroulant `File`. Pour compiler une M-File, il suffit de taper F5. Vous avez aussi la possibilité de fabriquer des fonctions (voir à la fin de ce document).

1 Matrices et tableaux

Une matrice peut être entrée directement, en ligne, en séparant les éléments d'une ligne par un blanc ou une virgule. Un point virgule permet de séparer deux lignes.
ex : $\mathtt{A} = [16\ 3\ 2\ 13;\ 5\ 10\ 11\ 8;\ 9\ 6\ 7\ 12;\ 4\ 15\ 14\ 1]$

`sum(A)` : vecteur-ligne, somme des colonnes de `A`.
`A'` : transposée de `A`.
`sum(A')'` : vecteur-colonne, somme des lignes de `A`.
`diag(A)` : vecteur-colonne des éléments diagonaux de `A`.
`A(i,j)` : élément de `A` sur la i-ième ligne et j-ième colonne.
`A(8)` : élément de `A`, situé en 8-ième position quand on lit en descendant les colonnes (ici, `A(8)`=15).

Matlab adapte la taille de la matrice suivant les besoins. Par exemple,

$$\left\{\begin{array}{l}\mathtt{X} = \mathtt{A};\\ \mathtt{X}(4,5) = 17\end{array}\right. \qquad \text{donne} \qquad X = \begin{pmatrix} 16 & 3 & 2 & 13 & 0\\ 5 & 10 & 11 & 8 & 0\\ 9 & 6 & 7 & 12 & 0\\ 4 & 15 & 14 & 1 & 17\end{pmatrix}$$

On peut utiliser `:` pour gagner du temps.
`1:10` est la suite (1,2,...,10).
`0:pi/4:pi` est la suite $(0, \pi/4, \pi/2, 3\pi/4, \pi)$.
`A(1:k,j)` donne les k premiers éléments de la j-ième colonne de la matrice `A`.
`:` signifie tous les éléments.
`end` signifie le dernier élément.

`sum(A(:,end))=sum(A(1:4,4))` calcule la somme de tous les éléments de la quatrième colonne de `A`.

Pour fabriquer des matrices, on peut utiliser (entre autres) ces 3 commandes :
`zeros(i,j)` génère une matrice $i \times j$ de 0,
`ones(i,j)` génère une matrice de taille $i \times j$ de 1,
`rand(i,j)` génère une matrice $i \times j$ de nombres uniformément répartis sur [0,1].

Ces constructions sont à utiliser sans modération, car préallouer de la place pour un tableau avant de le remplir fait gagner du temps.

On peut aussi construire des matrices par concaténation (par blocs) :

$$\mathtt{B} = [\mathtt{A}\ \mathtt{A}+32;\ \mathtt{A}+48\ \mathtt{A}+16] \qquad \text{donne} \qquad \left(\begin{array}{c|c} A & A+32\\ \hline A+48 & A+16\end{array}\right)$$

<u>Opérations sur les matrices :</u>
- somme de deux matrices : `A+B`
- produit de deux matrices (suivant les règles du calcul matriciel) : `A*B`
- multiplication terme à terme : `A.*B`
- calcul du déterminant, de l'inverse, du vecteur des valeurs propres, du polynôme caractéristique : `det, inv, eig, poly`
- extraction d'éléments d'une matrice : `A(cond(A))`.

Par exemple, `A(A>=0)` extrait les éléments positifs de `A`,
`A(isprime(A))` extrait les éléments premiers de `A`,
`A(~isprime(A))` extrait les éléments qui ne sont pas premiers de `A`,
Une autre manière d'arriver à de tels résultats : `k=find(isprime(A))` est le vecteur des numéros des éléments premiers de `A`. Puis `A(k)` donne les éléments premiers de `A`.

2 Graphiques

Deux méthodes pour tracer des fonctions : à partir d'un vecteur d'abscisses ou directement.

La commande `plot` trace une ligne brisée entre des points. Si `x` et `y` sont deux vecteurs de même longueur, `plot(x,y)` trace une ligne qui relie les points $[x(n), y(n)]$. Et `plot(y)` trace une ligne brisée de sommets $[n, y(n)]$. Pour tracer plusieurs telles lignes, on peut utiliser `plot(x1,y1,x2,y2,x3,y3)`.

Pour tracer la fonction sinus, voici les deux solutions :

-
```
t = 0:pi/100:2*pi
y = sin(t)
plot(t,y)
```

- `fplot('sin',[0 2*pi])`. On peut aussi définir une fonction au préalable avec la commande `inline`.

On peut choisir parmi une grande variété de types de graphes en utilisant `plot(x,y,'`$\alpha\beta\gamma$`')` où

- α : couleur (c cyan, m magenta, y jaune, r rouge, g vert, w blanc, k noir).
- β : type de ligne (- continue, -- tirets, : pointillés, -. donne $- \cdot - \cdot - \cdot -$, none aucune).
- γ : type de points (+,o,*,x).

Ces champs sont facultatifs. Exemple : `plot(x,y,'y :+')`.

Quand on a déjà tracé une courbe, `plot` trace la nouvelle courbe dans la même fenêtre en effaçant le précédent graphe. Il faut utiliser `figure` pour ouvrir une nouvelle fenêtre et `hold on` pour tracer un nouveau graphe dans la fenêtre courante en conservant les graphes déjà tracés. On peut aussi ajouter toute sorte de légende, texte, titre, grille... Essayer :

```
t=-pi:pi/100:pi;
y=sin(t);
plot(t,y)
axis([-pi pi -1 1])
xlabel('-\pi\leq t\leq \pi')
ylabel('\sin(t)')}
title('Graphe de la fonction sinus')
text(1,-1/3,'Remarquez la parité')
```

3 Expressions, formats, aide, mise en forme...

Les variables n'ont besoin d'aucune déclaration de type ou de dimension. Le nom d'une variable commence par une lettre, puis n'importe quoi ou presque (pas d'opérateurs arithmétiques). Attention, Matlab est sensible à la casse : $\texttt{A} \neq \texttt{a}$.

La liste des variables utilisées pendant la session est obtenue par `who` ou `whos` (liste courte ou détaillée). Pour les effacer, on utilise `clear`.

Les nombres ont 16 décimales significatives et sont compris entre 10^{-308} et 10^{308}. On peut choisir plusieurs formats d'affichage : `format short` (5 chiffres), `format short g` (5 chiffres significatifs), `format long` (15 chiffres), `format long e` (15 chiffres significatifs). Enfin, `format compact` n'affiche pas les lignes blanches.

Si on ne veut pas afficher le résultat d'une ligne de commande, il faut la terminer par `;`. Si une ligne est trop longue, on peut la couper en utilisant `...` avant le renvoi à la ligne. Enfin, pour rappeler la ligne de commande précédente (au prompt), on peut utiliser la flèche ascendante du curseur.

L'instruction `helpdesk` permet de lancer l'aide de Matlab en format hypertexte. Pour obtenir le fichier d'aide d'une commande précise, il suffit de taper `help` suivi du nom de la commande. Pour faire une recherche sur un mot présent dans les fichiers d'aide, on utilise `lookfor` suivi du mot (essayer `lookfor inverse`). Pour plus de renseignements sur la liste des fonctions existantes sous Matlab, on peut demander `help elfun` (fonctions élémentaires), `help specfun` (fonctions spéciales), `help elmat` (fonctions sur les matrices), `help stixbox` (boîte à outils de statistique), `help datafun`...

Pour afficher un texte, `fprintf('blabla')`. On peut faire afficher des phrases plus complexes : `fprintf('on obtient les valeurs % g et % g\n', a, b)`.

4 Boucles, fonctions

La syntaxe des boucles est très simple, mais le temps de calcul est long : il ne faut donc pas en abuser.

Une boucle `if` s'écrit ainsi :

```
if cond
expr
elseif cond
expr
else expr
end
```

Une boucle `for` s'écrit par exemple :

```
for n = 2 : 30
expr
end
```

Une boucle `while` doit ressembler à :

```
while cond
expr
end
```

Il est possible de sortir rapidement d'une boucle `while`, aussi bien que d'une boucle `for`, grâce à la commande `break`, en insérant quelque chose comme

```
if cond
break
end
```

Quelques exemples de conditions : `==`,`~=` et `>=` comparent deux scalaires ; `isequal(A,B)` teste l'égalité de deux matrices ; `all(V)=1` vérifie que tous les éléments d'un vecteur V sont nuls ; `any(V)=1` teste si un élément d'un vecteur V est nul...

On peut fabriquer des fonctions qui correspondent aux procédures Maple ou aux fonctions numériques. Ces sous-programmes doivent être sauvés dans un M-File qui porte le nom de la fonction. Ce fichier doit contenir :

```
function y = mafonction(x)
...
y = ...
```

Il suffit ensuite d'appeler `mafonction(x)` pour les valeurs de x désirées.