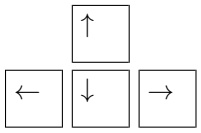
UNIVERSITÉ DE RENNES 1 **U.F.R. Mathématiques**

T.P. d'introduction à Matlab

1) LANCER Matlab
Ouvrez une fenêtre "Terminal" et tapez y `matlab`.
Dans la fenêtre "Command Window", notez le prompt de Matlab : `>>`
C'est là que les commandes doivent être tapées.

Ergonomie : Dans ce contexte, vous pourrez utiliser le pavé de touches  pour reprendre les commandes précédentes et les modifier.

2) QUITTER Matlab
Taper `>> quit`

PREMIERS PAS – COMMANDES ÉLÉMENTAIRES

Remarque : on peut taper plusieurs commandes Matlab sur une même ligne, en les séparant par une virgule. Voici des exemples variés à essayer :

OPÉRATIONS NUMÉRIQUES (remarquer les priorités)

```
>> 5*6, 2^5, 3+5*2^5
```

Essayez de deviner les résultats des deux commandes suivantes :

```
>> 3+5*2^5/5
>> (3+5*2^5)/5
```

Exemples d'affectation de variables (signe =)

```
>> x=2, y=x^5
>> y/x
```

La dernière réponse est appelée `ans`, à défaut de lui avoir donné un nom. On peut l'utiliser ainsi (deviner les résultats) :

```
>> ans, z=3*ans, z=4*z
```

Regardez la liste actuelle des variables de votre espace de travail :

```
>> who
>> whos
```

La commande `clear` supprime toutes les variables créées :

```
>> clear, who
```

Les calculs sont toujours exécutés en double précision. L'*epsilon machine* est une constante qui dépend de la machine et caractérise la précision maximale que l'on peut atteindre au cours des calculs. Sa valeur est mémorisée dans la variable `eps`. Que vaut-elle ?

```
>> 1+eps-1
>> 2+eps-2, 2+2*eps-2
```

Indépendamment de cela, l'affichage des valeurs numériques peut être modifié grâce à la commande `format` (voir `help format`) :

```
>> a=sqrt(3)
>> format long, b=sqrt(3)
>> a-b
>> format short   (retour au format standard)
```

Combien de décimales correctes peut-on espérer ?

MATRICES

En fait Matlab est essentiellement un outil matriciel : il considère un nombre réel comme un tableau 1×1. Tout est tableau, et Matlab est particulièrement adapté aux calculs numériques d'algèbre linéaire.

Voici différentes possibilités pour créer ou modifier une matrice et la façon de désigner ses coefficients :

```
>> a=[1,2,3;4 5 6]
```

Mais peut-être préfèrerez-vous :

```
>> a=[1 2 3
>> 4 5 6]
```

```
>> a(1,2), a(2,3)  (affichage de coefficients)
>> a(2,3)=10       (modification d'un coefficient)
>> a=[]            (création d'une matrice vide)
```

Quelques matrices particulières :

```
>> rand(1,3), rand(2,2)       (création de tableaux aléatoires)
>> zeros(1,3), ones(3,2)      (création de tableaux de 0, de 1)
>> eye(3,3), eye(2,3)         (création de matrices "identité")
>> b=[1,2,3], diag(b)         (matrice de diagonale donnée)
```

Nota : On peut supprimer l'affichage en terminant la commande par un point-virgule.

```
>> s=zeros(20,25);
```

C'est utile quand on travaille avec de grosses matrices, ainsi que dans les programmes.

DOCUMENTATION

Le résultat de la commande `help` s'affiche dans la fenêtre de travail, ce qui peut être gênant. On peut aussi utiliser les commandes `helpwin`, `doc` qui donnent accès à la documentation en format hypertexte (HTML) par l'intermédiaire d'un navigateur, dans une autre fenêtre.

Prenez le temps de découvrir cette aide en ligne pour qu'elle devienne un outil commode dans les TP suivants.

Enfin, quand on ne connaît pas le nom exact d'une commande, on peut utiliser l'instruction

```
>> lookfor mot_clef
```

qui effectue une recherche dans les en-têtes des rubriques d'aide (en Anglais).

QUELQUES ÉLÉMENTS DE PROGRAMMATION

Comme on vient de le voir, Matlab interprète des *commandes* (ou *instructions*) et manipule des *variables* qui sont des matrices. Il est souvent nécessaire de *structurer* la suite d'instructions qui permet de résoudre le problème posé. C'est possible avec Matlab : il possède un langage de programmation et reconnaît un certain nombre de *structures de contrôle* qui sont présentées dans la suite.

Dans ce contexte, il est peu pratique de taper les instructions de manière interactive comme on vient de le faire. À l'aide d'un éditeur de texte, on enregistre donc les instructions dans un fichier d'extension `.m`, que l'on appelle *script*, qu'on exécute ensuite.

Remarque : Vous disposez de plusieurs éditeurs de texte (`gedit`, `geany`, `nedit`, `emacs`). Il est également possible d'utiliser l'éditeur de Matlab.

Quitter Matlab. Créer un répertoire `introMatlab`, puis dedans un répertoire `tp1`, s'y positionner et relancer Matlab. (Commandes Unix : `cd; mkdir -p introMatlab/tp1; cd introMatlab/tp1; matlab`)

Exercice 1. Premier exemple de boucle.
À l'aide d'un éditeur de texte, créer le fichier `exo1.m` contenant les instructions suivantes :

```
for j=1:5
 for i=1:10
   t(i,j)=i/j;
 end
end
```

Devinez la matrice `t` construite par ces lignes. Vérifiez en exécutant le script :

`>> what` *(cette commande donne la liste des fichiers d'extension* `.m`*)*
`>> exo1`
`>> t`

Essayer aussi sans le point-virgule dans le script. Consulter l'aide concernant `for` : `>> help for`

Exercice 2. Initialisation des variables.
a. Modifier le fichier `exo1.m` de la façon suivante :

```
for j=1:5
 for i=1:5
   t(i,j)=2*i/j;
 end
end
```

Enregistrer le fichier modifié sous le nom `exo2.m`.
b. Exécuter les scripts `exo1.m` puis `exo2.m`, puis afficher la valeur de `t`. Normalement, la matrice calculée par `exo2.m` occupe les premières lignes dans la matrice `t` calculée précédemment par `exo1.m`. Ceci met en évidence la nécessité d'**initialiser** les variables, en particulier avant les boucles `for`. Modifier le script `exo2.m` en conséquence et l'exécuter à nouveau.

RETOUR AUX MATRICES

MANIPULATION DES MATRICES ET DES VECTEURS

1) Dans Matlab, `i` (`j` aussi par défaut) désigne le nombre imaginaire pur de partie imaginaire égale à 1.

`>> i=sqrt(-1)` *(réinitialisation puisque* `i` *a été modifié précédemment)*
`>> v=[1+i 2+2*i 3-3*i]` *(vecteur complexe)*
`>> v.'` *(transposé)*
`>> v'` *(adjoint)*

ou plus simplement :

```
>> w=[1+i 2+2i 3-3i]
>> size(v), size(w), length(w)  (la commande whos donne aussi cette information)
```

2) Construction à partir de tableaux donnés :

```
>> b=v'; a=[1,2,3;4 5 6]; c=[a; 7 8 9]
>> [a;a;a], [a,a,a]
>> [a',b]
>> [[a;a;a],[b;b]]
```

3) Utilisations de ":"

• construction de suites arithmétiques

`>> -3:3` *(ici le pas est implicitement égal à 1)*
`>> t=2:3:11` *(ici on précise que le pas doit être égal à 3)*
`>> t=2:3:13` *(partir de 2, aller de 3 en 3 jusqu'à 13, même si "ça ne tombe pas juste")*
`>> u=1:.25:4` *(le pas peut être fractionnaire)*
`>> y=2:-.3:-2.4` *(le pas peut être négatif)*

On peut l'utiliser pour construire des matrices,

```
>> a=[1:6 ; 2:7 ; 4:9 ; 30:35 ]
```

ou extraire des sous-vecteurs en générant des vecteurs d'indices (essayez de deviner les résultats),

```
>> y(2:12), y(9:-2:1)
>> x=10:100; x(2), x(10), x(40:5:60)
```

• joker

```
>> a(4,:)   (pour voir la 4ème ligne de la matrice a)
>> a(:,1)   (pour voir la première colonne de la matrice a)
```

On peut les combiner pour considérer des blocs dans une matrice ; par exemple

```
>> a(:,3:5)      (affiche les colonnes n°3 à 5 de la matrice a)
>> a(1:3,:)      (affiche les lignes n°1 à 3 de la matrice a)
>> a(1:3,3:5)    (bloc des colonnes n°3 à 5 et des lignes n°1 à 3)
```

CALCUL MATRICIEL

Après avoir entré les données :

```
>> a=[1 2 3; 4 5 6; 7 8 10] , b=[1 1 1]'
```

on peut essayer d'effectuer ces opérations

```
>> 2*a ,a/4
>> a+[b,b,b]
>> a*b, b*a , b'*a    (dimensions compatibles ou non ?)
```

mais aussi ces opérations étranges :

```
>> a+1, b+2
>> 1./a , 1./a.^2
```

Comparer les opérations suivantes :

```
>> a.*a , a*a' , a'*a
>> b.*b , b'*b , b*b'
>> a.^2 , a^2
```

Combinez ce que vous avez déjà découvert. Que vont donner les quatre lignes :

```
>> n=5;
>> v=ones(n-1,1);
>> M=2*eye(n,n);
>> M=M+diag(v,1)+diag(v,-1);
```

Exercice 3.

a. Écrivez une nouvelle version du programme mémorisé dans `exo1.m` en utilisant une opération vectorielle de manière à supprimer la boucle interne.

b. Obtenez la même matrice en utilisant uniquement des opérations vectorielles et la multiplication matricielle (plus de boucle).

c. Consultez la documentation de l'instruction `cputime` (`help cputime`). Remplacez 5 par 500 et mesurez le temps d'exécution des trois versions (`exo1.m` et les deux précédentes) :

```
T=cputime; exo?  ; cputime-T
```

Conclusion ?