MATLAB : COMMANDES DE BASE

Note : lorsqu'applicable, l'équivalent en langage C est indiqué entre les délimiteurs / */ .*

Aide

help, help nom_de_commande

Fenêtre de travail (Command Window)

Ligne de commande : »
Exécution : ↵
Séparateur d'instructions : virgule, point-virgule et ↵ /* ; */
Le point-virgule inhibe l'affichage du résultat de l'instruction
Rappel d'instructions précédentes : ↑ ou *premières lettres* ↑

Variables numériques

Par défaut, toutes les variables sont de type $\mathbb{C}$: $x+iy$. Si $y=0$, seule la partie réelle sera affichée.
Par défaut, les lettres *i* et *j* représentent $\sqrt{-1}=e^{i\pi/2}$.
Par défaut, les lettres *pi* représentent π.

Chaînes de caractères

a='symboles_ascii' affecte la chaine de caractères *symboles_ascii* à la variable *a* (pour représenter l'apostrophe, il faut le doubler : *'aujourd"hui'*) /* " */

Affectation

Nom=valeur

- Si la variable *Nom* n'existe pas, elle est créée dans l'espace mémoire (workspace) et prend la valeur *valeur*.
- Si *Nom* existe, l'ancienne affectation est perdue et remplacée par *valeur*.
- Matlab est sensible à la casse dans le nom des variables.
- Pour être valide, le nom d'une variable doit commencer par une lettre et s'écrire en un seul mot. On peut utiliser le symbole _ , mais pas les autres symboles.
-

Affichage du résultat d'une opération à l'écran

1. Absence de point-virgule.
2. Instruction *disp.*

Message d'erreur

Dans le cas d'une commande illégale, Matlab retourne un message d'erreur généralement très précis et instructif.

Fonctions et opérateurs sur les nombres

Les fonctions de Matlab sont définies sur $\mathbb{C}$.

^ : exposant.
sqrt : racine carrée.
sin, cos, …, asin, acos, … : argument ou réponse en radian.
sind, cosd, asind, acosd, … : argument ou réponse en degré.
exp, log, log10 : exponentielle, logarithme népérien et logarithme en base 10.
real, imag, conj, abs, angle : parties réelle et imaginaire, conjugué complexe, module et phase.
floor, round, ceil : partie entière, entier le plus proche et entier supérieur ou égal à la valeur.

Vecteurs et matrices

1. Vecteur ligne

   - Liste de valeurs entre [ ], entrées séparées par des virgules ou par un espace.
   - Définition globale :
     $x = [m : h : M]$ donne un vecteur ligne formé des nombres *m, m+h, m+2h, … M',* ou $M - h < M' \leq M$. Si $h = 1$, on peut l'omettre. Pour un vecteur ligne, on peut aussi omettre les [ ].
   - Fonctions d'initialisation
     linspace(a,b,n) : *n* nombres espacés régulièrement entre *a* et *b*.
     zéros(1,n) : *n* zéros.
     ones(1,n) : *n* uns.

2. Vecteur colonne

   - Liste de valeurs entre [ ], entrées séparées par ; .
   - Fonctions d'initialisation *zéros(n,1), ones(n,1).*
   - Transposition d'un vecteur ligne avec l'apostrophe : $x = [m : h : M]'$.

3. Matrice $m \times n$

   - Écriture ligne par ligne : $A = [1\ 2\ 3; 4\ 5\ 6]$
   - Fonctions d'initialisations
     zéros(m,n)
     ones(m,n)
     eye(m,n) : uns sur la diagonale, zéros ailleurs.

- Concaténation
  Si A est une matrice $m \times n$ et B une matrice $p \times n$, alors $C = [A;B]$ est une matrice $(m+p) \times n$.
  Si A est une matrice $m \times n$ et B une matrice $m \times p$, alors $C = [A,B]$ est une matrice $m \times (n+p)$.

Accès à des éléments d'une matrice

$A(k,l)$ retourne A_{kl}. k et l doivent être entiers et **la numérotation commence à 1**. /* *A[k,l],* la numérotation commence à 0 */.

Sous-matrices

$A(k1:k2,\ l1:l2)$ est la sous-matrice comprenant les lignes *k1* à *k2* et les colonnes *l1* à *l2.*
$A([k1\ k2\ \cdots\ ],\ [l1\ l2\ \cdots\ ])$ est la sous-matrice formée des lignes *k1, k2, ...* et des colonnes *l1, l2, ...*

Raccourcis :

- *end* représente le dernier élément d'une ligne ou d'une colonne.
- $A(:,\ l1:l2)$ est équivalent à $A(1:end,\ l1:l2)$.
- $A(k1:k2,\ :)$ est équivalent à $A(k1:k2,\ 1:end)$.

Attention :

- Si A est de taille $m \times n$ et si $k > m$ ou $l > n$, $x = A(k,l)$ provoque une erreur.
- Par contre, $A(k,l) = x$ ne provoque pas d'erreur : A est agrandie et les éléments manquants sont initialisés à 0.

Si A est de taille $m \times n$, $C = sum(A,\text{dim})$ et $C = mean(A,\text{dim})$ retourne un vecteur $1 \times n$ (dim=1) ou $m \times 1$ (dim=2) qui contient la somme ou la moyenne des lignes ou des colonnes de A. Si dim=1, il peut être omis.

Opérations sur les matrices

1. Opérations élément par élément

   Si A et B ont le même nombre de lignes et de colonnes, alors

- $C = A + B$ additionne les matrices A et B.
- $C = A - B$ soustrait les matrices A et B.
- $C = A.*B$ multiple les éléments de A et B élément par élément
- $C = A./B$ divise les éléments de A par les éléments de B élément par élément
- $C = A.\hat{}\ x$ élève chacun des éléments de A à la puissance x.
- $C = A + x$ additionne x à chacun des éléments de A.

- $C = A * x$ multiplie chacun des éléments de *A* par *x*.
- Si *f* est une fonction de Matlab (*sin, exp, log, ...),* $C = f(A)$ applique la fonction *f* à chacun des éléments de *A*.

2. Calcul matriciel

   - Si *A* est de taille $p \times n$ et *B* de taille $n \times q$, alors $C = A * B$ effectue le produit matriciel de *A* et *B* : $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$.
   - Si *A* est de taille $n \times n$ et inversible et *B* de taille $n \times p$, alors $X = A \setminus B$ (division *à gauche*) retourne l'unique matrice *X* de taille $n \times p$ solution de l'équation $A * X = B$. En particulier, la solution du système d'équations linéaires
   $$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n \end{cases},$$
   si elle existe, s'écrit $X = A \setminus B$.
   - Si *A* est de taille $n \times n$ et inversible et *B* de taille $p \times n$, alors $X = B / A$ retourne l'unique matrice *X* de taille $p \times n$ solution de l'équation $X * A = B$.
   - Si *A* est de taille $n \times n$ et inversible, alors $C = A \,\hat{}\, (-1)$ ou $C = inv(A)$ affecte à *C* l'inverse de la matrice *A* : $A * C = \mathbf{1}$.

3. Transposée

   Si *A* est de taille $m \times n$, la matrice *A'*, de taille $n \times m$, est la transposée de la conjuguée complexe de *A* : $a'_{ij} = a^*_{ji}$.

Entrée/sortie de données

- *save nom_du_fichier.ext x y z ... -ascii* ou $save('nom\_du\_fichier.ext','x','y',\ldots,'-ascii')$ sauvegarde les variables *x, y, ...* dans le fichier *nom_du_fichier.ext* en format ASCII. Les variables doivent avoir le même nombre de colonnes.
- *load nom_du_fichier.ext* crée la variable *nom_du_fichier* qui contient les valeurs contenues dans *nom_du_fichier.ex*t .
- $x = load('nom\_du\_fichier.ext')$ met le contenu de *nom_du_fichier.ext* dans la variable *x*.

Taille des matrices

- *[m,n]=size(A)* retourne le nombre de lignes et de colonne de la matrice *A*.
- *n=length(A)* retourne la plus grande des valeurs entre le nombre de ligne et de colonnes.

Minimum, maximum, moyenne

- Si *A* est de taille $m \times 1$ ou $1 \times n$, *C=max(A)* contient la plus grande valeur de *A*.
- Si *A* est de taille $m \times n$ avec *m* et $n > 1$, *C=max(A)* est de taille $1 \times n$ et contient la plus grande des valeurs de chaque colonne de *A*.
- Si *A* et *B* sont de taille $m \times n$, *C=max(A,B)* est de taille $m \times n$ et contient, pour chaque élément, la plus grande des valeurs des éléments correspondants de *A* et *B*.
- *C=max(A, [],dim)* effectue l'opération selon la dimension *dim*.
- De la même façon, *min* retournent les minimums.
- Si *A* est de taille $m \times 1$ ou $1 \times n$, *C=mean(A)* contient la valeur moyenne des éléments de *A*.
- Si *A* est de taille $m \times n$, *C=mean(A,dim)* effectue la moyenne selon la dimension *dim*.

Instruction « find»

v=find(condition) ou *[k,l]= find('condition')* retourne les indices d'une matrice qui satisfont à la condition *condition*. Par exemple, *v=find(x>0)* retourne les indices des éléments de *x* plus grands que 0.

Graphisme 2D

Si *x* et *y* sont de taille $m \times 1$ ou $1 \times m$,

- *plot(x,y)* génère un graphique dans la fenêtre graphique et trace une ligne reliant les points de coordonnées (x_i, y_i).
- *semilogx(x,y), semilogy(x,y)* et *loglog(x,y)* génèrent respectivement des graphiques semi-logarithmiques et logarithmiques.
- *plot(x,y,'couleur')* trace une ligne de couleur *couleur* : *r* (rouge), *b* (bleu), *g* (vert), *c* (cyan), *k* (noir), …
- *plot(x,y,'type_de_ligne')* trace une ligne de type *type_de_ligne* : - (continu), -- (tireté), : (pointillé), …
- *plot(x,y,'symbole')* trace le symbole *symbole* à chaque point : *o* (cercle), . (point), *s* (carré), *+* (plus), *x* (×), …
- On peut combiner les instructions : *plot(x,y,'type_de_ligne symbole couleur').*
- *plot(*x_1, y_1*,'instructions',*x_2, y_2*,'instructions',…)* trace plusieurs courbes sur le même graphique. On peut obtenir le même résultat avec les instructions *hold on* et *hold off*.
- *h=plot(x,y,…)* retourne le code d'identification (« handle ») du graphique (voir les instructions *get(h) et set(h)*).
- *axis([*$x_{\min}, x_{\max}, y_{\min}, y_{\max}$*])* fixe manuellement les plages en *x* et *y*.

- *subplot(m,n,k)* divise la fenêtre graphique en $m \times n$ systèmes d'axes et se place dans la $k^{ième}$.
- *xlabel('descriptif') et ylabel('descriptif')* ajoutent les descriptifs sur les axes.

Expressions et opérateurs logiques

Comme dans le langage C, le nombre 0 représente dans Matlab la valeur logique *faux* et 1 la valeur logique *vrai*.

1. Expressions logiques

   Égalité : ==
   Inégalité : ~= /* != */
   Supérieur : >
   Supérieur ou égal : >=
   Inférieur ou égal : <=
   Inférieur : <

2. Opérateurs logiques

   et : & /* && */
   ou inclusif : | /* || */
   non : ~ /* ! */

Si *A* et *B* sont de taille $m \times n$, *C=(A expression_logique B)* retourne une matrice de taille $m \times n$ contenant des 1 et des 0 selon que la condition est vraie ou fausse.

Scénarios et fonctions

Un **scénario** (« script ») est un fichier qui contient des instructions exécutables comme si elles avaient été tapées directement dans la fenêtre de commande. Ce fichier doit avoir l'extension *.m* .

- Délimiteur de commentaires : % ↵ /* /* */ */
- Instruction continuée sur la ligne suivante : ...

L'éditeur de Matlab permet d'écrire facilement des scénarios. Ceux-ci sont exécutés en tapant dans la fenêtre de commande l'instruction *nom_du_scénario*. Les variables utilisées dans le scénario sont automatiquement créées dans le *workspace*.

Une **fonction** est un scénario dont le premier mot de la première ligne est **function**.

function [a,b,c,...]=zozo(x,y,z,...) crée une fonction ayant les variables *x, y, z, ...* en entrée et les variables *a, b, c, ...* en sortie. Les fonctions sont également enregistrées dans des fichiers

portant l'extension *.m* . **Il est fortement recommandé de donner le même nom au fichier que celui de la fonction !**

Exemple :

```
function y=exposant(x,u)
% élève x à l'exposant u

y=x^u ;
```

crée la fonction *exposant*. Dans l'espace de commande, on peut taper *mon_nom_de_variable=exposant(2,3)*, qui retourne la valeur 8 dans la variable *mon_nom_de_variable*.

Les variables utilisées dans une fonction sont des **variables formelles** qui n'ont pas d'existence effective. Elles ne sont pas créées dans le *workspace*. Ces variables formelles ne servent qu'à indiquer le procédé de calcul. Elles ne sont reconnues qu'à l'intérieur de la fonction.

Structures de condition ou de répétition du langage Matlab

1. Alternative

```
if expression logique
        instruction
        instruction
        …
end
```

ou encore

```
if expression logique
        instruction
        instruction
        …
elseif expression logique
        instruction
        instruction
        …
elseif …
…
else …
…
end

/* if (  ) {  } else if (  )  {} … else {  }  */
```

2. Répétition

   while *expression_logique*
   instruction
   instruction
   …
   end

   se lit : tant que *expression_logique* est vraie, exécuter les instructions. Attention, comme en langage C, si *expression_logique* est une variable, elle est considérée comme vraie à moins qu'elle ne soit exactement égale à 0.

   /* do { } while ( ) . En langage C, l'instruction *do* est exécutée au moins une fois, même si la condition *while* est fausse, ce qui n'est pas le cas en Matlab. */

3. Boucle

   for k=début : pas : fin
   instruction
   instruction
   …
   end

   se lit : pour *k* égal *début* jusqu'à *fin* par pas de *pas*, exécuter les instructions.

   /* for (i=début ; expression_logique pour la fin ; i=i+pas) { } */

4. Sortie

   Lorsque le programme rencontre l'instruction *break* à l'intérieur d'une répétition *while* ou d'une boucle *for*, il va immédiatement après le premier *end* qui ferme la répétition ou la boucle.

Exemple

L'exemple simple suivant traduit en langage Matlab le programme de bissection écrit en langage C dans la section 5.2 des notes du cours PHY 1234.

Contenu du fichier *bissection.m*

```
function [rac,drac]=bissection(fonc,x1,x2,epsilon,intermax);
% function [rac,drac]=bissection(fonc,x1,x2,epsilon,intermax);
%
% Entrée :
%  fonc : nom de la fonction dont on veut trouver la racine ;
```

```
%  x1,x2 : intervalle dans lequel devrait se situer la racine ;
%  epsilon : précision requise pour la racine ;
%  nombre maximal d'itérations.
% Sortie :
%  rac : racine ;
%  drac : incertitude sur la racine

k=0;
delta=x2-x1;

while (delta>epsilon) & (k<intermax)
    xm=0.5*(x1+x2);
    fm=feval(fonc,xm); % équivaut à fm=contenu_de_fonc(xm)
    f2=feval(fonc,x2);
    if fm*f2>0
        x2=xm;
    else
        x1=xm;
    end
    delta=x2-x1;
    k=k+1;
end

rac=xm;
drac=delta;
```

Contenu du fichier zozo.*m*

```
function y=zozo(x)

y=x.*cos(x)-sin(x);
```

Instruction dans l'espace de travail

```
>> [racine,delta_racine]=bissection('zozo',1,6,1e-5,40) ;
```

Vous devriez aussi essayer

```
>> r=fsolve('zozo',2) ;
```