Mémento Matlab

Conservez ce mémento, mais surtout, complétez-le au fur et à mesure des séances de TP !

Notions générales

- Le résultat d'une affectation `=` est affiché, sauf si cette affectation se termine par un caractère `;`
- Les commandes `format short` et `format long` permettent de modifier le format d'affichage des variables.
- Les commandes `who` et `whos` permettent d'afficher l'ensemble des variables utilisées.
- La commande `clear` efface le contenu de toutes les variables utilisées.
- Il est fortement déconseillé d'utiliser des mots-clés de Matlab comme noms de variables.
- La commande `help <fonction>` affiche la description de `<fonction>` (exemple : `help plot` décrit la syntaxe de l'affichage des graphiques 2D).

Manipulation de vecteurs et de matrices

- Les composantes d'un vecteur ligne sont séparées par des virgules ou des espaces : `v1 = [x1 y1 z1];`
- Les composantes d'un vecteur colonne sont séparées par des points-virgules : `v2 = [x2 ; y2 ; z2];`
- Vecteur à incrément constant : `v3 = x_min:dx:x_max;` (vecteur ligne de dimension variable, qui contient les valeurs `x_min+i*dx`, où `i` est un entier positif ou nul tel que `x_min+i*dx` est inférieur à `x_max`).
- Les matrices utilisent la même syntaxe que les vecteurs : `M = [m11 m12 m13 ; m21 m22 m23];`
- La sous-matrice de `M` constituée par les lignes de numéros pairs et les colonnes de numéros impairs s'écrit : `N = M(2:2:size(M,1),1:2:size(M,2));`
- Vectorisation d'une matrice (les colonnes de `M` sont concaténées) : `v = M(:);`

Quelques matrices utiles

- `zeros(m,n)` : matrice nulle de taille $m \times n$.
- `ones(m,n)` : matrice de taille $m \times n$ dont tous les éléments sont égaux à 1.
- `eye(m,n)` : matrice de taille $m \times n$ dont les éléments diagonaux sont égaux à 1, les autres à 0.
- `rand(m,n)` : matrice de taille $m \times n$ d'éléments tirés aléatoirement selon la loi uniforme sur $[0, 1]$.
- `randn(m,n)` : matrice de taille $m \times n$ d'éléments tirés aléatoirement selon la loi normale centrée réduite.
- Appeler ces fonctions avec un seul argument équivaut à les lancer avec deux arguments identiques.

Opérations sur les matrices

- Addition `A+B` ; soustraction `A-B` ; produit `A*A'` ; puissance `A^3` ; transposition `A'` ou `transpose(A)`.
- Inverse `inv(A)` ; pseudo-inverse `pinv(A)`.
- Multiplication **élément par élément** `A.*B` (chaque élément `A(i,j)` est multiplié par l'élément `B(i,j)`) ; division **élément par élément** `A./B` (chaque élément `A(i,j)` est divisé par l'élément `B(i,j)`) ; puissance **élément par élément** `A.^3` (chaque élément de `A` est élevé à la puissance 3).

Quelques fonctions utiles

- `length(V)` calcule la taille d'un vecteur (ligne ou colonne) ou la plus grande dimension d'une matrice.
- `indices = find(V==0)` : le vecteur colonne `indices` contient les indices `i` de la *matrice vectorisée* `W = V(:)` tels que `W(i)==0` (n'importe quelle expression booléenne peut être passée en paramètre à la fonction `find`).
- `[V_trie,indices] = sort(V,'ascend')` : si `V` a plus d'une ligne, alors `V_trie` est une version de `V` triée par ordre croissant, colonne par colonne ; sinon, le tri est effectué sur la ligne unique de `V` ; la matrice `indices` contient les indices correspondants ; pour `'descend'` au lieu de `'ascend'`, idem par ordre décroissant.