Saisie de matrices en Matlab

- Une matrice type
  - [*a11, a12, a13 … ; a21, a22, a23 …*]
  - [1, 4, 6 ; 2, -1, 0.5 ; 3, 6, -2 ; 4, -7, 18] ⟺ $\begin{bmatrix} 1 & 4 & 6 \\ 2 & -1 & 0{,}5 \\ 3 & 6 & -2 \\ 4 & -7 & 18 \end{bmatrix}$
  - [ ] signalisent qu'on va composer une matrice à partir de plusieurs éléments
  - , sépare les éléments d'une ligne (les colonnes) ; peut être remplacé avec l'espace
  - ; sépare les lignes
- Un vecteur ligne (matrice 1×n)
  - [*a1, a2, a3 …*]
- Un vecteur colonne (matrice n×1)
  - [*a1; a2; a3 …*]
- Un élément simple (matrice 1×1)
  - [*a*]
- Une matrice vide (0×0)
  - []

Création raccourcie de matrices en Matlab

- Si les éléments forment une suite arithmétique
  - Raccourci : [*premier* : *raison* : *dernier*] (produit toujours un vecteur ligne)
  - [2 : 0.5 : 4.5] équivaut [2 2.5 3 3.5 4 4.5]
  - Si *raison* est omise, 1 sera supposé
  - [2 : 5] équivaut [2 3 4 5]
  - Pour transformer une ligne en une colonne, la transposer (')
- Matrice de uns
  - ones(*nombre_lignes, nombre_colonnes*)
  - ones(2,3) ⟺ $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
  - ones(*taille*) produit une matrice carrée et non pas un vecteur !
  - ones(1,3) ⟺ $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$
  - ones(3) ⟺ $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
  - par analogie – matrice de zéros : zeros
- Matrice d'identité
  - eye(*taille*)
  - eye(3) ⟺ $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Manipulations de matrices en Matlab

- Combinaison
  - matrices placées l'une à droite de l'autre = ajout de colonnes
    [M1, M2, M3 …]
  - matrices placées l'une au dessous de l'autre = ajout de lignes
    [M1; M2; M3 …]
  - les *Mx* désignent des matrices en général ; ça comprend également – en tant que cas spécifiques – vecteurs (n×1 / 1×n) ou éléments simples (1×1)
- Extraction
  - un élément d'une matrice : *M(no_ligne, no_colonne)*
  - un élément d'un vecteur : *V(no_element)*
  - une colonne d'une matrice : *M(:, no_colonne)*
  - une ligne d'une matrice : *M(no_ligne, :)*
  - plusieurs colonnes : *M(:, premiere_colonne : derniere_colonne)*
  - plusieurs lignes : *M(premiere_ligne : derniere_ligne, :)*
  - une sous-matrice :
    M(premiere_ligne : derniere_ligne, premiere_colonne : derniere_colonne)

Taille de matrices

- Détermination du nombre d'éléments en Matlab :
  - numel(*M*) – nombre d'éléments total
    - > matrice = [3 5 ; 7 -2 ; -9 11];
    - > numel(matrice)
      ans = 6
  - size(*M*) – nombre de lignes et nombre de colonnes
    - > size(matrice)
      ans = 3 2
    - > [nl, nc] = size(matrice)
      nl = 3
      nc = 2
  - Cela fonctionne aussi avec des tableaux à plus de 2 dimensions
- Pour le cas standard de 2 dimensions on peut se servir également de :
  - columns(*M*) – nombre de colonnes
  - rows(*M*) – nombre de lignes

Dans les exemples d'instructions Matlab entrées en ligne, « > » va signaliser une entrée dans la ligne de commandes d'Octave

Types composés – chaînes de caractères

- **Chaînes de caractères**
  - introduites entre " ou '
  - beaucoup de langages (Matlab compris) discernent entre " et '
- Dans beaucoup de langages (Matlab compris) une chaîne de caractères est considérée un tableau d'une ligne (vecteur horizontal) à données du type alphanumérique
  - pour combiner plusieurs chaînes :

```
> t1 = "abc"; t2 = "def"
> t3 = [t1, t2]
t3 = abcdef
```

- Caractères spéciaux (communs à beaucoup de langages)
  - sont introduits avec la barre inversée (backslash) \
  - souvent (Matlab) sont reconnus seulement entre les " et non pas entre les '
  - \n fin de ligne (écran et fichiers Linux)
  - \r\n fin de ligne (fichiers Windows)
  - \t tabulation (fin de « colonne »)
  - \\ la barre inversée même

Texte dans un programme

- Par défaut : considéré corps du programme
  - alors : nom d'une variable, nom d'une fonction, valeur du type numérique...
  - a = sin(3.14)
- Entre guillemets " ou apostrophes ' : valeur du type chaîne de caractères
  - alors : texte que l'interpréteur/compilateur n'essaye pas de comprendre, qui peut néanmoins être traité par l'ordinateur (mémorisé, transformé, affiché...)
  - t = "Message a afficher"
- Après ou entre les caractères spécifiques : commentaire
  - alors : note destinée au programmeur, donc pas à l'ordinateur qui ne peut pas s'en servir d'aucune manière et va l'ignorer entièrement
  - Matlab : à commencer du pour cent % (Octave : ou croisillon #) jusqu'à la fin de la ligne
  - a = sin(3.14) % 3.14=pi radians correspond a 180 degres
  - ou depuis une ligne %{ jusqu'à une ligne }%
  - dans d'autres langages (exemples) : après l'instruction Rem (Basic), après le caractère ' (Visual Basic), entre les caractères /* et */ (C)

Affectation

- **Opérateur de l'affectation =**
  - nombre = 0.56
  - texte = 'alpha'
  - v2 = [0 5 2]
  - matrice = [2 3 4; 5 8 10; -1 -5 9]
- Attention au symbole « = »
  - Trompeur si on compare Matlab avec pseudo-code
  - Il y a des langages ou l'affectation est marquée autrement tandis que = désigne la relation de l'égalité

Langage	Affectation (action)	Égalité (relation)
Pseudo-code	a ← 2	a = 2
Matlab, C	a = 2	a == 2
Pascal	a := 2	a = 2
Visual Basic	a = 2	a = 2

- C'est une action
  - Ce n'est pas donc une relation (déclaration/constatation d'un fait persistant, durable)
  - même si dans beaucoup de langages notée à l'aide d'un opérateur
- S'effectue une fois précisément au moment où l'ordinateur y arrive
- On ne peut pas associer deux variables de façon permanente
  - a = 2 # a devient égal 2
    b = a # b devient égal 2
    a = 3 # a devient égal 3
    # b reste égal 2 ≠ a

Gestion d'identifiants

- Si la variable avec le nom donné n'existe pas, elle est créée, c'est-à-dire :
  - son *nom* (« étiquette ») est ajouté au **tableau d'identifiants** (noms, symboles) reconnus
  - son *type* est aussi mémorisé dans ce tableau
  - une place (« boîte ») dans la mémoire est réservée pour garder sa *valeur*
  - l'*adresse* (emplacement, « numéro de la boîte ») de cet espace est aussi mémorisé dans le tableau
- Liste d'identifiants en Matlab
  - who
    affiche une liste de noms de variables définies
  - whos
    ajoute de l'information sur taille de matrice, occupation de la mémoire et classe de variable
- Suppression en Matlab
  - clear *nom*
    supprime la variable ou la fonction portant le nom *nom*
  - l'entrée correspondante est effacée du tableau d'identifiants
  - la place réservé dans la mémoire devient libre et peut ensuite être occupée par une autre variable

Opérateurs arithmétiques

- **Opérateurs arithmétiques** scalaires
  - addition + soustraction -
  - multiplication * division /
  - puissance ^ ou ** (il y a des langages où ça n'existe pas comme opérateur)
  - il y en a plus dans d'autres langages, p. ex. modulo (reste) mod
- Matriciels de Matlab (tailles doivent être cohérentes)
  - matriciels classiques : + - * / \ ^ ** ' (transposition)
    - A/B est équivalent à A * B^-1
    - A\B est équivalent à A^-1 * B (calculé plus vite avec \)
  - terme à terme : + - .* ./ .\ .** .^ – c'est ça qui est nécessaire dans beaucoup d'algorithmes de traitement de données
  - matrice inverse : A^-1 ou inv(A)
- Affectation + arithmétique (C, Matlab)
  - a+=4 est équivalent à a=a+4 (par analogie : -= *= /=)
  - a++ est équivalent à a=a+1 (par analogie : --)

Interaction avec l'utilisateur (ou avec soi-même lors du développement du programme)

- Affichage le plus simple : *nom_de_la_variable*
  - > a
    a = 5
  - utile pour vérifier le contenu d'une variable à l'étape donnée de l'exécution
- Affichage : disp
  - disp(a)
    affiche la valeur de la variable a
  - disp("Message")
    affiche le texte « Message »
  - Pour afficher un texte suivi d'une valeur, il faut deux *disp* séparés
    disp("Le prix moyen vaut :")
    disp(prix_moy)
- Demande d'une entrée : input
  - prix_achat=input("Entrez le prix d'achat de l'article : ");
    affiche le texte, attend l'entrée d'une valeur et l'enregistre dans la variable *prix_achat*
  - mot=input("Entrez un mot : ", "s");
    pour des chaînes de caractères (variables du type alphanumérique)