TD°1 : Structures de contrôle, tableaux/vecteurs/matrices

Elément de Cours : Structures de contrôle

GNU Octave permet d'utiliser les structures de contrôle de l'algorithmique, c'est-à-dire qu'il permet d'écrire des séquences d'action, d'effectuer des sélections et des répétitions d'action.

Sélection :

La forme générale de la sélection dans GNU Octave est :

```
if expression
      actions
elseif expression
      actions
else
      actions
end
```

Exemple :

```
a=0
b=2
if a>b
      c= a+b
else
      c= a*b
end
```

Répétition :

Si vous ne connaissez pas le nombre de répétitions à effectuer, il faut appliquer la boucle jusqu'à ce qu'une condition soit satisfaite.

```
while expression
      actions
end
```

Exemple :

```
while x > 1
      x = x/2
end
```

Attention : il existe d'autres formes pour décrire la sélection et la répétition !

Exercice 1 : Compteur

Ecrire un programme qui demande à l'utilisateur un nombre cible et qui affiche la liste des nombres en partant de la valeur 1 jusqu'à la valeur cible avec un pas de 1.

- Écrire l'algorithme en pseudo code.
- Le traduire en langage GNU Octave

Exercice 2 : Moyenne

Ecrire un programme permettant de déterminer et d'afficher le minimum, le maximum et la moyenne de 10 nombres fournis en entrée.

- Écrire l'algorithme en pseudo Code.
- Le traduire en langage GNU Octave

Elément de Cours : Tableaux et Vecteurs en GNU Octave

Beaucoup de problèmes mathématiques fonctionnent avec des séquences de nombres. Dans la plupart des langages, ces séquences sont manipulées au travers de tableaux (***arrays***). Dans Octave, on parlera de vecteurs (***vectors***).
Ces vecteurs sont communément utilisés pour représenter des positions dans un espace 3D, des vitesses, etc. mais ce ne sont en fait qu'une liste de données traitables avec Octave.

En fait, les vecteurs ne sont que des sous-ensembles de matrices (*matrix*) – grille à n-dimensions de données. **Un vecteur ne contient soit qu'une ligne, soit qu'une colonne de données.** On pourra distinguer les vecteurs-colonnes (***column vector***) des vecteurs-lignes (***row vector***).

Création de vecteurs

Il y a plusieurs façons de définir les vecteurs et les matrices. La façon la plus simple consiste à définir le vecteur entre « crochets » [] dans lesquels on place des données séparées soit par :

- des espaces ou des virgules → définira un vecteur-ligne
- des points-virgules ou des « retour-chariot » → définira un vecteur-colonne

Exemple :

```
➢ a = [1 4 8]
 a = 1 4 8
➢ a = [1, 4, 8]
 a = 1 4 8
➢ a = [1;4;8]
 a =     1
    4
    8
```

Il est aussi possible de définir un nouveau vecteur en incluant un vecteur déjà défini

Exemple :

```
➢ d = [a 6]
 d = 1 4 8 6
```

Elément de Cours : Création avancée de vecteurs

Il existe une autre façon de construire un vecteur de manière semi-automatique en utilisant les « : ».

Exemple :

- `v = 2:6` → construit un vecteur débutant par le 1er nombre et finissant par le nombre le plus proche (par défaut) du dernier avec un pas de 1

```
v= 2     3     4     5     6
```

La notation permet aussi d'introduire un 3ème argument : **le pas**

(premier nombre) : (pas) : (dernier nombre)

Exemple :

- `v = 2:0.2:3`

```
v = 2.0  2.2   2.4   2.6   2.8   3.0
```

Note : Il est possible de spécifier un pas négatif

Fonctions de création de vecteurs

`zeros(1,N)`	Créé un vecteur rempli de zéros (N = n éléments)
`ones(1, N)`	Créé un vecteur rempli de 1 (N = n éléments)
`linspace(x1, x2, N)`	Crée un vecteur de N éléments espacés entre les valeurs x1 et x2 (pas déterminé en fonction de N)
`logspace(x1, x2, N)`	Crée un vecteur de N éléments entre les valeurs de départ espacés logarithmiquement entre 10^{x1} et 10^{x2}

Nota : Les deux premières fonctions fonctionnent pour les matrices en jouant avec les valeurs des deux arguments M et N (lignes et colonnes respectivement)

Extraction d'éléments d'un vecteur

Chaque élément d'un vecteur est accessible individuellement en utilisant les parenthèses (), numérotés à partir de **1** (En langage C, la numérotation commence à zéro)

Exemple :

- `a = [1:4 6 8]`

```
a = 1   2     3     4     6     8
```

- `a(5)`

```
ans = 6
```

La notation « *deux-points* » peut aussi être utilisée pour spécifier une plage de données et récupérer plusieurs éléments en même temps

Exemple :

- `a (3:5)`

```
ans = 3  4  6
```

Elément de Cours : Manipulation de vecteurs

Contrairement à de nombreux langages, il est possible d'effectuer des calculs matriciels de haut-niveau avec GNU Octave.

Opérations avec un scalaire

Multiplier un vecteur par un scalaire est aussi simple que multiplier un scalaire par un autre scalaire. Il suffit d'utiliser l'opérateur *.

De la même façon, il est possible de diviser le vecteur par un scalaire avec l'opérateur /

Exemple :

```
➢ a = [1:5]
 a = 1   2     3     4     5
➢ a * 2
 ans =   2     4     6     8     10
➢ a / 2
 ans = 0.50000      1.00000     1.50000     2.00000     2.500000
```

Enfin, il est possible d'utiliser les opérateurs d'addition et de soustraction à un vecteur alors qu'il n'existe pas d'équivalent en mathématiques. Le résultat consiste en l'addition/soustraction **pour chacune des valeurs** du vecteur du scalaire donné
Exemple :

```
➢ a + 2
 3 4     5     6     7
```

Exercice 3 : Chiffres

Ecrire un programme qui détermine tous les nombres situés entre deux bornes fournies en entrée du programme. Déterminez ensuite la multiplication de tous ces nombres.

- Écrire l'algorithme en pseudo code.
- Le traduire en langage Octave

Exercice 4 : Factorielle

Ecrire un programme qui calcule la factorielle d'un nombre fourni en entrée.

- Écrire l'algorithme en pseudo code.
- Le traduire en langage Octave

Exercice 5 : Devinette

Ecrire un programme permettant à la machine de deviner un nombre entre 1 et 100 choisi par l'utilisateur. Pour trouver le nombre, la machine fera des propositions successives à l'utilisateur. À chacune de ces propositions, l'utilisateur devra répondre en indiquant si le nombre à rechercher est inférieur, égal ou supérieur au nombre proposé.

La machine doit deviner le nombre en au plus 7 coups. Si au bout de 7 coups le nombre n'a pas été trouvé c'est que l'utilisateur a triché !

- Écrire le pseudo code.
- Le traduire en langage GNU Octave

Elément de Cours : Matrices

Les vecteurs ne sont que des cas spéciaux de matrices. Une matrice est un tableau à deux dimensions de nombres. Sa taille est décrite usuellement par la notation *m x n*, signifiant que la matrice comporte *m* lignes et *n* colonnes.

Exemple :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 0 \end{bmatrix}$$

Création de matrices

Plusieurs manières de précéder sont disponibles pour créer une matrice avec GNU Octave

Ligne par ligne :

Il suffit d'entrer ligne par ligne la matrice (« retour chariot » pour changer de ligne)

Exemple :

```
➢ A =    [1   2     3 <CR>
➢        1     0     0]
```

Ligne par ligne – 2ème façon

Une deuxième manière de faire est construire pas à pas la matrice en utilisant les « ; » à la fin de chaque ligne (particulièrement utile dans les programmes)

Exemple :

```
➢ A = [1     2     3] ;
➢ A = [A ;   -1    0     0]
 A =     1     2     3
    -1    0     0
```

Une seule ligne séparée par des points-virgules

D'une façon assez similaire, il est possible d'utiliser les « ; » pour séparer les lignes de la matrice.

Exemple :

```
➢ A = [1 2 3 ; -1 0 0]
 A =     1     2     3
    -1    0     0
```

Notation « deux-points »

Dernière notation possible, en utilisant les « : » comme pour les vecteurs

Exemple :

```
➢ B = [1 :3 ; 0 :2]
 B =     1     2     3
    0     1     2
```

Matrice vide

Enfin, il est possible de créer une matrice vide (avec ajout d'élément plus tard) en spécifiant le nom de la matrice suivi de crochets ouvrants et fermants

Exemple :

```
➢ E = []
E = [](0x0)
```

Manipulation élément par élément

Comme pour les vecteurs, il est possible de manipuler les éléments d'une matrice isolément en utilisant la même syntaxe que pour les vecteurs avec les parenthèses.

Exemple :

```
➢ J = [
      1  2     3     4
      5  6     7     8
      9  10    11    12]
➢ J(1,1)          // extrait l'élément ligne 1 colonne 1
 Ans = 1
➢ J(2,3)          // extrait l'élément ligne 2 colonne 3
 ans = 7
➢ J (1:2,4)       // extrait l'élément en colonne 4, ligne 1 et 2
➢ J(2,:)          // extrait tous les éléments de la ligne 2
```

Nota : L'élément : permet d'extraire soit un nombre d'éléments déterminés (plage de données), soit un ensemble complet de données.

Exercice 6 : Stockage de données

Considérons un groupe de 30 étudiants. Chaque étudiant est associé à un identifiant. Les identifiant sont gérés de 1 à 30. Nous avons besoin d'associer à chaque étudiant un ensemble de notes, chacune correspondant à une matière différente.

Voici à quoi ressemblera cet ensemble de données :

N'étudiant	Matière 1	Matière 2	Matière 3	Matière 4
1	4	11	15.5	14
2	18	10	9	13.8
…				
30	12	14	19	7

- Développez un algorithme permettant de saisir et de stocker toutes les notes pour chacun des étudiants. Pour des raisons de simplicité lors de la saisie on souhaitera saisir toutes les notes matière par matière et non pas étudiant par étudiant.
- Traduisez cet algorithme en Octave
- On souhaite extraire et stocker dans un autre tableau toutes les notes de l'étudiant n°14. Proposez directement l'algorithme en Octave permettant d'extraire ce résultat.
- On souhaite calculer pour chaque étudiant sa moyenne générale (notes sans coefficient). Proposez l'algorithme GNU Octave permettant de la calculer et de la stocker dans un tableau séparé.

Elément de Cours : Quelques fonctions de traitement matriciel avec Octave

Addition matricielle :

Soit 2 matrices M, N. L'addition matricielle en octave s'écrit simplement :

$$M + N$$

Produit matriciel ordinaire :

Soit 2 matrices M, N. Le produit matriciel ordinaire en octave s'écrit simplement :

$$M * N$$

Nota : Attention de ne multiplier que des matrices à taille compatible ! Avec les vecteurs, il faut faire attention entre les vecteurs-colonnes et vecteurs lignes ! Par exemple, un vecteur-colonne (`1 x n`) ne peut pas être multiplié par une matrice `m x n`

Produit matriciel membre à membre :

Soit 2 matrices M et N. Le produit matriciel membre à membre en GNU Octave s'écrit simplement :

$$M .* N$$

Fonction de création de matrices

Octave fournit des fonctions de création de matrices.

`zeros(M,N)`	Créé une matrice remplie de zéros.
`ones(M, N)`	Créé une matrice remplie de 1.
`eye(N)`	Créé une matrice identité (matrice carrée de NxN éléments)
`rand(N)`	Créé une matrice carrée d'éléments aléatoires (matrice carrée de NxN éléments)
`diag([vecteur])`	Créé une matrice diagonale (0 partout sauf sur la diagonale où sont posés les éléments du vecteur). **Nota :** La matrice identité est un cas spécial de la matrice diagonale

Extraction de matrice diagonale

La fonction **`diag`** (déjà vue plus haut) permet aussi de créer une matrice diagonale à partir d'une matrice préexistante.

Exemple :

```
➢ B = [1     2     3] ;
➢ B = [B ;   4     5     6]
➢ diag(B)
 ans =   1
    5
```

Nota : Il n'est pas nécessaire que la matrice soit carrée, l'extraction s'achève dès qu'il n'est plus possible d'extraire d'éléments.

Taille d'une matrice

La fonction **size** permet de renvoyer un couple (M, N) [lignes, colonnes] correspondant au nombre de lignes et de colonnes que comporte la matrice.

Exemple :

```
➢ size(rand(4))
 ans = 4       4
```

Déterminant

La fonction **`det`** calcule le déterminant de la matrice. (On rappelle que si le déterminant est égal à zéro, la matrice est non inversible)

Matrice inverse

Pour calculer la matrice inverse A^{-1}, on utilisera la fonction **`inv`** si le déterminant de matrice est non nul. On rappelle que $A * A^{-1} = A^{-1} * A = I$

Matrice transposée

Enfin, il existe un opérateur permettant d'obtenir la transposée d'une matrice ${}^tA$, l'opérateur '. Si A est une matrice, A' renverra sa transposée.

Rang d'une matrice

Il est parfois important de connaître le nombre de lignes ou colonnes linéairement indépendantes dans une matrice notamment pour résoudre les équations du type Ax = b. La fonction **`rank`** permet de le savoir.

Et bien d'autres …

Il existe d'autres fonctions de manipulation de matrices que nous verrons lors d'exercices de manipulation d'Octave.

Un opérateur utile …

Pour résoudre l'équation **`Ax = b`** (n équations à n inconnues), quand A est inversible, on calcule

$x = A^{-1}b$

Matlab (et GNU Octave donc) définit un opérateur que l'on peut qualifier de « *division matricielle* », l'opérateur \ (qui utilise la méthode d'élimination gaussienne, plus efficace que la résolution classique). Il est à noter que cette notation n'est pas standard et n'a pas d'équivalent en mathématiques

Exemple : `x = A\b`