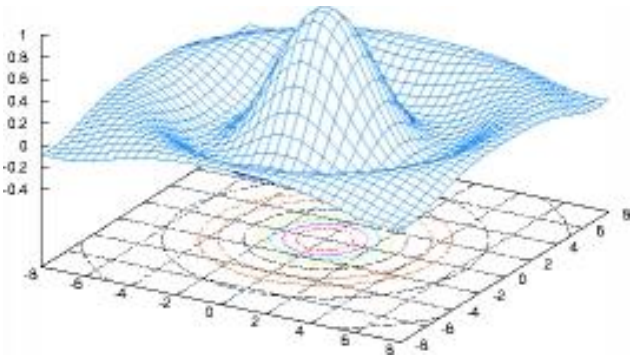
GNU OCTAVE



http://www.octave.org

http://wiki.octave.org/Main_Page

Notes de cours 1ère année GCGEO

Septembre 2019 – v.1.8

Philipe Truillet – Philippe.Truillet@irit.fr

1. INTRODUCTION

GNU OCTAVE, C'EST …

Matlab est un outil très utilisé dans les universités. Le faible temps nécessaire à l'écriture d'un programme, l'association du calcul et de la visualisation qui permet de tester en ligne ses modifications et la possibilité d'incorporer directement les figures dans un traitement de texte comme MS-Word ont fait le succès de Matlab. GNU Octave est la version libre (et gratuite) de Matlab. Les possibilités sont moindres mais pas inintéressantes.

Depuis Mai 2015, GNU Octave se présente avec une interface graphique qui le rend plus proche de Matlab. Le principal problème réside dans la partie graphique, traité par un autre logiciel **gnuplot**.

En résumé :

- Un logiciel open-source géré sous licence GNU
- Créé au départ pour un enseignement en chimie
- Un logiciel dont la syntaxe est compatible MATLAB

Il permet de résoudre numériquement des problèmes mathématiques (pas de solutions symboliques)

QUI UTILISE GNU OCTAVE ?

Les ingénieurs et les scientifiques, à la fois dans l'industrie et dans le monde académique pour des calculs numériques, du développement et des tests algorithmiques.

Par exemple, l'équipe Jaguar s'est servie de GNU Octave pour afficher et analyser les données transmises par ses Formule 1, l'université de Sheffield pour développer des logiciels de reconnaissance de cellules cancéreuses.

GNU Octave permet :

- D'écrire des programmes rapidement
- D'afficher les données de diverses manières facilement

DIFFERENCES AVEC UN LANGAGE DE PROGRAMMATION « HAUT-NIVEAU »

GNU Octave a été conçu spécifiquement pour résoudre des problèmes mathématiques et afficher les résultats.

L'écriture de programmes complexes prend du temps, surtout pour des langages qui ne supportent pas nativement les concepts mathématiques ou l'affichage graphique.

GNU Octave est **un langage interprété** ce qui signifie que chaque commande est convertie en code machine dès qu'il est tapé. Bien que les calculs soient plus longs qu'avec un langage compilé (comme C), il est plus simple de commencer avec un tel langage interprété.

En résumé, on prototype sous Octave et on développe ensuite dans un autre langage.

2. CALCULS SIMPLES

INSTALLER GNU OCTAVE

Rendez-vous à la page `https://www.gnu.org/software/octave` pour télécharger la version de GNU Octave adaptée à votre système d'exploitation. Au 1er septembre 2019, la dernière version est la 5.1.0

LANCER GNU OCTAVE

Tapez "Octave" dans une fenêtre de commande, lancez le via le menu « démarrez » | « programme » | Octave 5.1.0 ou directement depuis le répertoire d'installation en cliquant sur octave.vbs sous windows (cf . Figure 1)

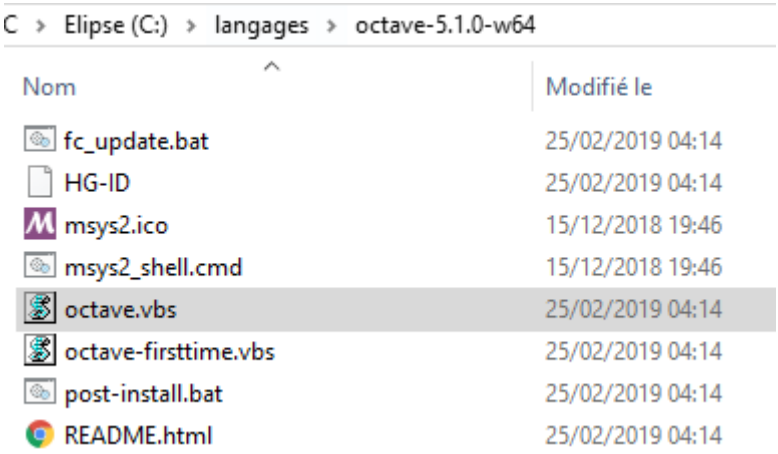


Figure 1 : Lancer GNU Octave

Une interface graphique s'affiche (cf. Figure 2)) dont dans l'onglet « Fenêtre de Commandes » un prompt (>>) qui permet de taper des commandes.

GNU Octave est d'abord une interface à ligne de commande, chaque commande étant tapée une par une après le prompt et terminée par l'appui de la touche « entrée ».

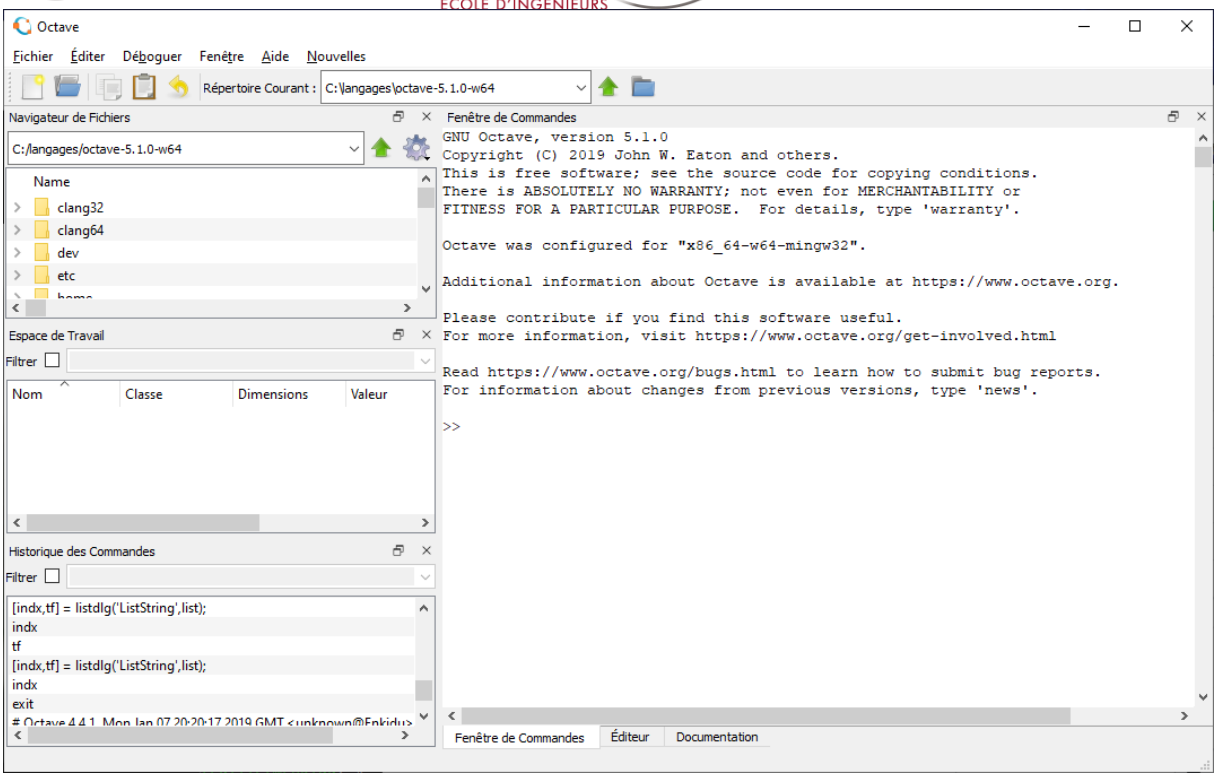


Figure 2 : interface graphique de GNU Octave (depuis la version 4.x)

Pour quitter, tapez la commande **quit** (suivi de la touche « entrée »)

GNU OCTAVE EST UNE CALCULATRICE

La façon la plus simple d'utiliser GNU Octave et de taper directement des commandes mathématiques.

Toutes les expressions mathématiques classiques sont reconnues.

Ex :

➢
```
2+2
ans = 4
```

Les opérateurs mathématiques sont : + - * / ^ (exposant). Par défaut, le résultat de chaque opération est stocké dans la variable `ans` (**ans**wer - réponse)

On peut aussi utiliser les parenthèses.

Les priorités des opérateurs sont les mêmes qu'en mathématiques c'est-à-dire :

- Les expressions « *parenthésées* » sont évaluées en premier
- Puis les exposants
- Puis les multiplication et divisions
- Et enfin les additions et soustractions

FONCTIONS USUELLES

Les fonctions sont invoquées comme dans le langage C, c'est-à-dire en utilisant le nom de la fonction suivi de ses arguments entre parenthèses.

Ex :

- ```
  exp(1)
  ans = 2.71183
  ```

On peut calculer des expressions plus complexes comme 1.5 cos(90° + ln(8.12²))

- ```
  1.5 * cos(90°*pi/180 + log(8.12^2))
  ```

Notez que :

- Le signe explicite de la multiplication est **absolument nécessaire** dans les équations
- Les fonctions trigonométriques fonctionnent en radians. Le facteur pi/180 doit être utilisé pour convertir les degrés en radians (**pi** est un exemple de variable nommée)
- Le logarithme népérien (ln) se nomme **log** sous GNU Octave

Fonctions usuelles

cos	Cosinus (en radians)
cosh	Cosinus hyperbolique
acos	Cosinus inverse
acosh	Cosinus inverse hyperbolique
sin	Sinus (en radians)
sinh	Sinus hyperbolique
asin	Sinus inverse
asinh	Sinus inverse hyperbolique
tan	Tangente (en radians)
tanh	Tangente hyperbolique
atan	Tangente inverse
atan2	Tangente inverse – forme à deux arguments
atanh	Tangente inverse hyperbolique
exp	Exponentielle
log	Logarithme népérien
log10	Logarithme en base 10
abs	Valeur absolue
sign	Signe d'un nombre (-1 ou +1)
round	Arrondi à l'entier le plus proche
floor	Arrondi par défaut (vers – infini)
ceil	Arrondi par excès (vers + infini)
fix	Arrondi autour de zéro
rem	Reste de la division entière
sqrt	Racine carrée

3. ENVIRONNEMENT OCTAVE

VARIABLES

Dans n'importe quel calcul, on a besoin de stocker des résultats pour une réutilisation future ou simplement pour un affichage.

GNU Octave permet bien entendu de créer de nouvelles variables et de les manipuler.

CREATION

Pour définir une variable, il suffit simplement de taper son nom (identificateur) sans **spécifier de type associé** et sa valeur.

Toutes les variables sont par défaut des nombres à virgule flottante (les caractères sont en fait une suite de nombres).

Ex :

- `deg = pi/180`

UTILISATION

Pour réutiliser une variable, il suffit de taper son identificateur dans une formule.

Ex :

- `cos(90°)` → `cos(90*deg)`

A noter

- ce n'est pas forcément plus rapide et facile à définir des variables et les utiliser MAIS cela permet de faire moins d'erreurs et permettre une lecture plus aisée des programmes
- Quand on tape une formule sans l'assigner à une variable, GNU Octave assigne quand même le résultat à une variable : la variable **ans**.

SUPPRESSION

Pour supprimer une variable, il suffit de taper la commande **clear** suivie du nom de la variable

Attention !

- Taper **clear** sans nom de variable associé supprime toutes les variables !

NOMBRES ET FORMAT

FORMATS

Par défaut, GNU Octave affiche les nombres avec au moins **5 chiffres significatifs**. La commande **format** permet de modifier cet affichage (**ET seulement l'affichage !**)

Par exemple, la commande **format long** permet d'afficher au moins 15 chiffres significatifs (**format short** permet de revenir à 5 chiffres significatifs)

Il existe de nombreux autres formats d'affichage possibles accessibles via la commande **help format**

Nous verrons par la suite comment spécifier plus finement l'affichage des contenus de variables.

Ex : deg = pi/180

- `deg = 0,017453`

format long

- `deg = 0,0174532925199433`

format short

- `deg = 0,017453`

NOMBRES RECONNUS

- *Notation à virgule flottant* : Octave permet d'afficher des nombres très petits ou très grands en utilisant la notation à virgule flottante.
  Ex :
  `1312.65 = 1,31265 p 10³` → `1.31265e+03`

- *Nombres complexes* : Ils sont compris totalement par Octave (utilise deux variables réservées i et j)

- *Infini* (**Inf**) : le résultat d'une division par zéro. Pour Octave, c'est un résultat valide d'une opération et peut être affecté à une variable

- *NaN* (**Not a Number**) : le résultat de zéro divisé par zéro ou par des résultats indéfinis. Pour Octave, c'est un résultat valide et peut être affecté à une variable

NOMBRES COMPLEXES

CREATION

GNU Octave sait aussi manipuler les nombres complexes. Pour ce faire, il suffit de les entrer dans GNU Octave de la même manière qu'en mathématiques.

Ex :

- `Z = 4 - 3i`

Octave réserve deux noms de variables : **i** et **j** pour représenter les nombres complexes (i et j représentent la valeur $\sqrt{-1}$)

Ex :

- ```
  Z = 1 + 3*j
  Z = 1+3i
  ```

MANIPULATION DE NOMBRES COMPLEXES

Les opérations arithmétiques (+, - , *, /, ^) sont possibles sur les complexes tout comme elles le sont sur les réels.
  ```

Ex :

- `Z =1 + 3i`
- `Z^2`
  `ans = -8 +6i`

FONCTIONS SUR LES COMPLEXES

GNU Octave propose aussi une série de fonctions pour manipuler les nombres complexes : nombre conjugué, module et argument, partie entière et partie imaginaire.

Fonction	Signification	Définition (z = a + bi)
`imag`	Partie imaginaire	b
`real`	Partie réelle	a
`abs`	Module	r = |z|
`conj`	Nombre complexe conjugué	$\overline{z}$ = a - bi
`angle` ou `arg`	Argument	$\theta = tan^{-1}(\frac{b}{a})$

COMMANDES USUELLES GNU OCTAVE

CHARGEMENT/SAUVEGARDE DE DONNEES

A chaque fois que l'on quitte l'environnement GNU Octave, toutes les données manipulées sont perdues. Si vous devez réutiliser vos données, vous devez au préalable les sauver puis les recharger à la prochaine ouverture de session Octave.

- Pour sauver : utiliser la commande **save** suivie d'un nom, éventuellement de noms de variables
  Ex : **`save`** `mesdonnees` → sauve toutes vos données dans un fichier nommé *mesdonnees*.mat (ajouté automatiquement par Octave) dans votre répertoire courant.
  Ex : save *convert deg* → sauve la variable deg dans le fichier *convert*.mat

- Pour charger : utiliser la commande **load** suivie d'un nom de fichier
  Ex : **`load`** `mesdonnees` → charge l'environnement sauvegardé dans le fichier *mesdonnees*.mat

Nous verrons par la suite comment sauvegarder des programmes écrits avec Octave ou charger des données, contenant des mesures à afficher.

REPETITION DE COMMANDES

GNU Octave mémorise toutes les commandes tapées durant une session. Il suffit d'utiliser les flèches « haut » et « bas » pour revenir sur une commande déjà tapée (la plus récente s'affiche en premier). Dès que la commande souhaitée s'affiche, taper sur « entrée » pour la rappeler.

Pour éditer la commande en cours, on peut utiliser les flèches « droite » et « gauche » pour bouger le curseur, et la touche « *suppr* » pour changer le contenu. Ceci est particulièrement utile si Octave trouve une erreur dans l'expression.

AIDE

GNU Octave intègre un système d'aide intégré. La forme la plus simple d'utilisation est :

- **help** commande

Ex :

- `help sqrt`

```
sqrt is a built-in function
- Mapping Function: sqrt (X)
Compute the square root of X. If X is negative, a complex
result is returned. To compute the matrix square root, see

*Note Linear Algebra::.
```

ANNULATION DE COMMANDES

Si une commande met beaucoup de temps à s'exécuter, il est possible d'arrêter les calculs en tapant sur les commandes « CTRL+C ». Normalement, cette action permet de revenir au prompt GNU Octave.

POINT-VIRGULE ET RESULTATS CACHES

Dans l'algorithmique en général et la plupart des langages, le point-virgule permet de spécifier une séquence à exécuter.

Dans GNU Octave, l'usage du point-virgule est différent. Si à la fin d'une séquence, on ajoute un point-virgule, le résultat ne sera pas affiché. Cet usage peut être (parfois) intéressant si on n'a pas besoin de connaître le résultat ou bien si le résultat est très long à afficher

4. TABLEAUX ET VECTEURS

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 GNU 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 GNU 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*).

CREATION 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 :

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

Ex :

- ```
  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

Ex :

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

LA NOTATION « DEUX-POINTS »

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

Ex :

- 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)

Ex :

- ```
  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 CREATION DE VECTEURS

GNU Octave fournit des 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'ELEMENTS D'UN VECTEUR

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

Ex :

- ```
  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

Ex :

- ```
  a (3 :5)
  ans = 3 4 6
  ```

MANIPULATION DE VECTEURS

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

OPERATIONS 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 /

Ex :

- ```
  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

Ex :

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

OPERATIONS SUR LES MATRICES ELEMENT PAR ELEMENT

Pour effectuer des opérations de multiplication **élément par élément** (idem pour la division et la puissance) avec GNU Octave, il faut utiliser un opérateur « spécial » : .* (./, .^)

Le point avant l'opérateur signifie ici « opération élément-par-élément »

Ex :

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} .* \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_2 b_2 \\ a_3 b_3 \end{pmatrix}$$
  ```

```
➢ a = [1      3      5      7      0]
➢ b = [1      2      3      4      5]
➢ a.*b
  ans = [1    6      15     28     0]
```

L'opérateur « puissance » est intéressant dans la mesure où il devient très facile de mettre des vecteurs à une puissance donnée.

Ex :

```
➢ b .^2
  ans = [1    4      9      16     25]
```
→ chaque élément de b est élevé à son carré

```
➢ 2 .^b
  ans = [2    4      8      16     32]
```
→ vecteur de données de 2 à la puissance de chaque élément de b

Nota :

Ces opérateurs ne peuvent être utilisés entre deux vecteurs que s'ils ont la même taille ! Dans le cas contraire, une erreur est signalée.

5. 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.

Ex :

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

CREATION 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)

Ex :

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

LIGNE PAR LIGNE – 2EME 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)

Ex :

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

UNE SEULE LIGNE SEPAREE PAR DES POINTS-VIRGULES

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

Ex :

```
➢ 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

Ex :

```
➢ 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 ouvrant et fermants

Ex :

```
➢ E = []

E = [](0x0)
```

MULTIPLICATION DE MATRICES

Le signe * s'utilise avec les vecteurs et les matrices et permet d'effectuer des produits matriciels.

Rappel :

C (c_{ij})= A (a_{ij}) * B (b_{jk})

$$c_{ij} = \sum_{k=1}^{n} a_{ij} b_{jk}$$

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 (1xn) ne peut pas être multiplié par une matrice m x n

FONCTION DE CREATION 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

FONCTIONS DE MANIPULATION DE MATRICES

Plusieurs fonctions sont disponibles pour manipuler des matrices déjà existantes

EXTRACTION DE MATRICE DIAGONALE

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

Ex :

```
➢ 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.

Ex :

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

DETERMINANT

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 TRANSPOSEE

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 OPERATEUR UTILE …

Pour résoudre l'équation **Ax = b** (n équations à n inconnues), quand A est inversible, on calcule $x = A^{-1}b$

Matlab (et 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

Ex :

- `x = A\b`

MANIPULATION ELEMENT PAR ELEMENT

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.

Ex :

```
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.

6. AFFICHAGE DE DONNEES

AFFICHAGE BASIQUE

Octave a des fonctionnalités avancées d'affichage graphique de données grâce à un module externe : GNUPLOT (http://www.gnuplot.org).

La commande de base est **`plot(x,y)`** où x et y sont des coordonnées ou des vecteurs passés en paramètres.

Ex :

- `angles = linspace(0,2*pi,100)` // créé un vecteur de 100 valeurs espacées entre 0 et 2π
- `y = tan(angles)` // calcul des résultats de la fonction y = tan(x)
- `plot(angles,y)` // affichage du graphique

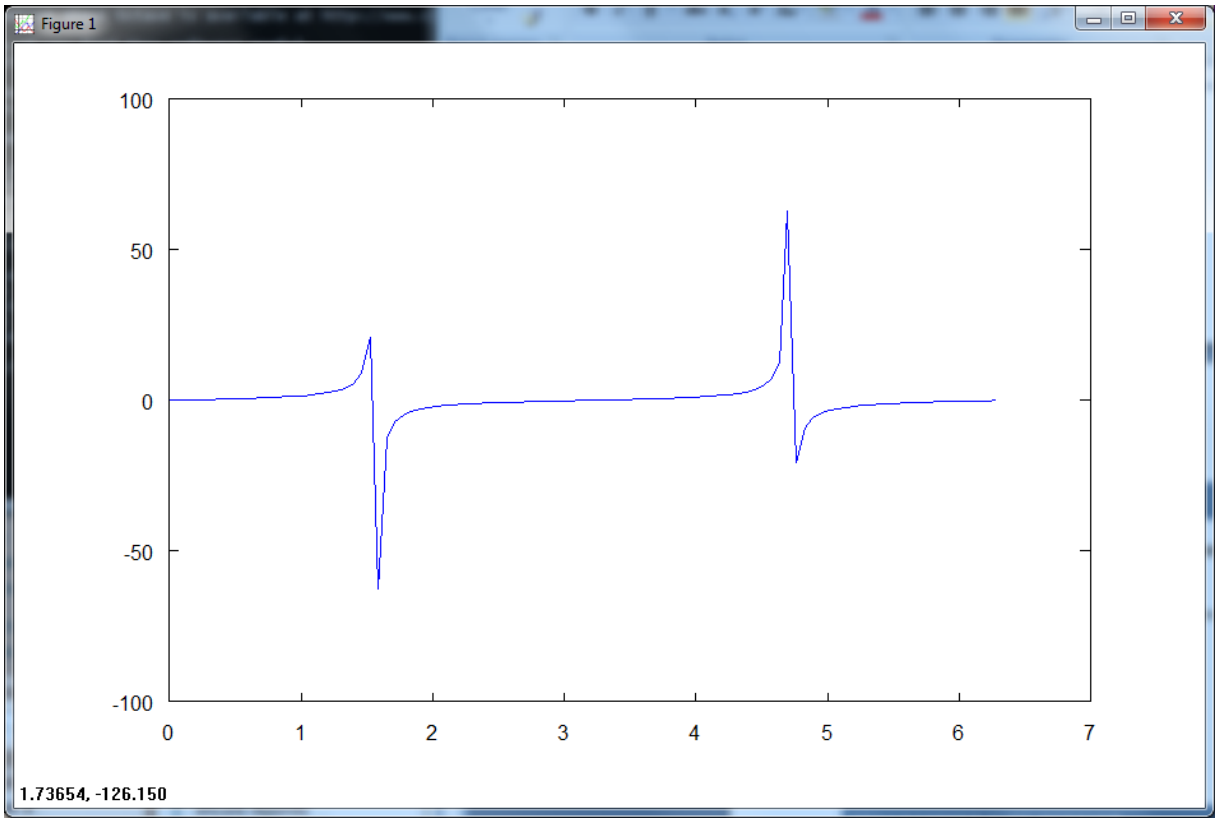


Figure 3 : affichage du graphique

AFFICHAGE EVOLUE

Bien évidemment, il est possible d'améliorer le graphe de base pour y ajouter des labels, titres voir afficher des graphes en 3D !

TITRE

`title` → ajoute un titre au graphique

Ex : `title('Tangente(x)')`

LABELS

`xlabel` → ajoute un label sur l'axe x

`ylabel` → ajoute un label sur l'axe y

Ex :

- `xlabel('x')`
- `ylabel('tangente(x)')`

PLUSIEURS GRAPHES DANS UNE MEME FENETRE

La commande `subplot` permet de partager la fenêtre pour afficher plusieurs graphes différents.

```
subplot(nombre_lignes, nombre_colonnes, selection)
```

L'argument `selection` permet de sélectionner le graphe courant dans la fenêtre. Les sous-fenêtres sont numérotées à partir du haut à gauche avec un parcours ligne par ligne.

Ex :

- `subplot(2,2,1)` // créé quatre sous-fenêtres et sélectionne la seconde (1ère ligne, 2ème colonne)
- `plot(x, cos(x))` // affiche le graphe dans la fenêtre sélectionnée

PROPRIETES D'AFFICHAGE

La commande plot permet d'ajouter des propriétés pour modifier l'affichage des données

Ex : `plot(z,'o')` → affiche la donnée z sous forme d'un cercle

w	blanc	**.**	point	**-**	solide
m	magenta	**o**	cercle		
c	cyan	**x**	croix		
r	rouge	**+**	plus		
g	vert	*	étoile		
b	Bleu				

Attention :

A chaque commande plot, l'affichage est réinitialisé ! Une commande permet d'éviter ce (léger) problème : `hold on` pour empêcher la réinitialisation et `hold off` pour la permettre à nouveau.

COMMANDES DIVERSES

Afin d'afficher les résultats avec un quadrillage utilisez la commande `grid on`

Bien entendu, il est aussi possible d'afficher plusieurs courbes sur un même graphique, rajouter des légendes, zoomer/dézoomer les axes manuellement, etc.

`legend` → ajoute une légende au graphe

`axis` → modifie les axes (x_{min}, x_{max}, y_{min}, y_{max}))

`figure` → permet de produire un graphe dans une nouvelle fenêtre

Ex :

```
legend('tangente')
```

`axis(-2 2 0 10)` → affiche la courbe entre -2 et 2 sur l'axe x et 0 et 10 sur l'axe y

Il est aussi possible d'interagir directement avec les données sur le graphe à l'aide de touches ou d'actions avec la souris.

Action	Résultat
Bouton Milieu souris	Annote le graphe
Bouton Droit souris	Permet de zoomer sur le graphe (en marquant la zone à zoomer)
a	Permet de recalibrer le graphe et de le réafficher
g	Affiche le quadrillage (un nouvel appui permet de l'enlever)
r	Affiche la règle
u	Dézoome le graphe
…	Pour les autres commandes, voir l'aide en ligne de GNUplot

AFFICHAGE 3D

Enfin, GNU Octave via GNUplot permet d'afficher des graphes en 3 dimensions. Là encore, plusieurs commandes sont disponibles.

- `plot3` : l'équivalent en 3D de la commande plot. Prend (x,y,z) en arguments.
- Avec `plot3`, on peut ajouter `zlabel` qui fonctionne comme `xlabel` et `ylabel`

Il est possible de créer une grille de points dans lequel une fonction `z=f(x,y)` sera affichée en utilisant la commande `meshgrid`.

Ex :

```
x = [2:0.2:4]        % créé un ensemble de valeurs x € [2,4]
y = [1:0.2:3]        % et y € [1,3]
[X,Y] = meshgrid(x,y)
```

L'affichage peut se faire avec la commande `plot3` (vue plus haut), `surf` ou `mesh`

Ex :

```
Z = X^2 + Y^2
surf(X,Y,Z)
mesh(Z)
```

Il est aussi possible d'interagir directement avec les données sur le graphe à l'aide de touches ou d'actions avec la souris.

Action	Résultat
Bouton Gauche souris + déplacement	Tourne la vue
<CTRL> + Bouton Gauche souris + déplacement	Tourne les axes (la vue est mise à jour quand le bouton de la souris est relâché)
Bouton Milieu souris	Mise à l'échelle de la vue
<CTRL> + Bouton Milieu souris	Mise à l'échelle des axes (la vue est mise à jour quand le bouton de la souris est relâché)
…	Pour les autres commandes, voir l'aide en ligne de GNUplot

SAUVER ET IMPRIMER DES FIGURES

Visualiser un graphe c'est bien, pouvoir l'imprimer et la sauvegarder, c'est mieux.

Pour l'imprimer, c'est simple, il suffit de taper la commande `print` (sous unix) ou `saveas`.

Pour sauvegarder le graphe, la même commande `print` est utilisée. La commande générale est :

```
print('nom.extension','-dextension')
```

De nombreux formats sont disponibles : gif, jpeg, png, eps, …. La figure est sauvée dans le répertoire courant.

Ex :

```
print('graphe.jpg','-djpeg')
```

7. PROGRAMMER AVEC GNU OCTAVE

Autre point intéressant avec GNU Octave, on peut étendre les fonctionnalités du langage en écrivant des scripts et ou nouvelles fonctions du système.

Sous GNU Octave, les scripts sont des fichiers texte et possèdent l'extension « `.m` » (matlab). Ils sont régis par certaines règles pour qu'ils puissent être réutilisables.

CREER UN NOUVEAU SCRIPT

Pour créer un nouveau script, on doit utiliser un éditeur de texte (celui fournit dans l'interface graphique de GNU Octave – onglet « éditeur » ou emacs, kedit, gedit, notepad, …)

Dans GNU Octave, un éditeur de texte peut être appelé en tapant la commande « `edit` ».

Dans l'éditeur, il suffit de taper la suite des commandes qui constituent le script et sauver le fichier. Le nom de sauvegarde constitue la commande à activer dans Octave. Par exemple, le script sauvé sous le nom « `affiche_sinus.m` » sera activable dans octave en tapant la commande « `affiche_sinus` ».

Nota :

- Les commentaires commencent par le signe « `%` » dans un script
- Les scripts sont sauvés et activables dans le répertoire courant.
- Comme toute commande octave, il est possible d'avoir une aide en tapant le mot-clé `help` suivi du nom de la commande (cela nécessite toutefois de définir l'aide dans le script)

STRUCTURES DE CONTROLE ET GNU OCTAVE

Comme tout environnement de programmation évolué, Octave permet d'utiliser les structures de contrôle de l'algorithmique.

SEQUENCE

La séquence dans Octave consiste à taper une suite de commandes (une par ligne) dans le script.

SELECTION

IF … ELSIF … ELSE … END

La forme générale de la sélection dans Octave est :

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

Nota :

La séquence est légèrement différente du langage C. En effet, les parenthèses '()' ne sont pas nécessaires autour des expressions (mais il reste préférable de les utiliser) et le bloc des actions ne sont pas entourées par des accolades '{}'. A la place, le mot-clé 'end' est utilisé pour signaler la fin de la structure 'if'.

Ex :

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

La sélection repose essentiellement sur le résultat d'expressions logiques. Voici les principales expressions booléennes utilisables.

Symbole	Signification	Exemple
==	Egale	`if x == y`
~=	Différent	`if x ~= y`
>	Plus grand que	`if x > y`
>=	Plus grand ou égale	`if x >= y`
<	Plus petit que	`if x < y`

<=	Plus petit ou égale	`if x <= y`
&	Et	`if x ~= y & x > 4`
|	Ou	`if x ~= y | y <1`
~	Non	`if x = ~y`

SWITCH

La sélection peut se faire aussi par la commande `switch` qui permet de choisir une action parmi n choix possibles.

```
switch x
      case c1
            actions
      case c2
            actions
      otherwise
            actions
end
```

Dans ce cas, la valeur du 'switch', est comparée à tous les cas listés et si un des cas correspond à la valeur, les actions correspondantes sont exécutées jusqu'au prochain 'case' (pas de mot-clé `break` comme en C). Si aucun cas ne s'applique, les actions listées dans la partie 'otherwise' sont exécutées.

Ex :

```
switch a
      case 0
            disp('a vaut zéro')
      case 1
            disp('a vaut un')
      otherwise
            disp('a n'est pas un nombre binaire')
      end
```

REPETITION

Tout comme la sélection, il y a plusieurs façons d'exprimer la répétition dans Octave

WHILE

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
```

Ex :

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

FOR

Cette structure permet de répéter un nombre de fois déterminé à l'avance des opérations.

```
for variable = vector
      actions
end
```

Ex :

```
for n=1:5
      nf(n) = n*3
end
```

- `nf = 3 6 9 12 15`

FONCTIONS

Comme nous l'avons vu, Octave permet d'écrire des scripts mais aussi des fonctions définies par l'utilisateur qui permet d'enrichir la bibliothèque Octave. Ces fonctions peuvent être réutilisées en ligne de commande ou encore dans des scripts GNU Octave.

Dans les fonctions Octave, les arguments sont toujours passés par **valeur** et non par ***référence*** (cela signifie que les valeurs passées par argument ne seront pas modifiées par la fonction). Autre aspect important, Les fonctions dans Octave peuvent retourner plus d'une valeur.

En général, une fonction est définie dans un fichier texte (tout comme les scripts). La différence essentielle consiste en l'écriture de la première ligne d'une fonction qui s'écrit de la manière suivante :

```
function [sortie1, sortie2, …] = nom_fonction(entree1, entree2, …)
```

Chaque fonction est stockée dans un fichier .m différent où le nom du .m est le même que la fonction définie dans le fichier. Définir une fonction permet de rendre du code plus lisible et éviter de recopier plusieurs fois la même suite d'instructions.

Ex :

cosinus renvoyé en degrés (et non en radians comme par défaut)

```
function c= cosd(x)

%COSD(X) CALCULE COS(X) AVEC X EXPRIME EN DEGRES

      c = cos(x*pi/180)

end
```

ENTREES/SORTIES FORMATEES, MANIPULATION DE CHAINES DE CARACTERES

AFFICHAGE DE TEXTE ET DE VARIABLES

La fonction **disp** permet d'afficher du texte ou toute donnée à l'écran (sous la forme spécifiée soit par défaut, soit par la commande **format**).

disp s'utilise soit avec une chaîne de caractère comme argument

Ex :

`disp('chaine')` affiche `chaine`

Soit avec une variable comme argument

Ex :

`disp(chaine)` affiche le contenu de la variable `chaine`

Nota :

A la fin de l'affichage, le curseur revient automatiquement à la ligne

Les fonctions **sprintf** (**s**tring **print f**ormated), **fprintf** (**f**ile **print f**ormated) et **printf** sont plus intéressantes. Très proches de la syntaxe de langage C, elles permettent d'afficher à la fois du texte et une ou plusieurs variables sous le format désiré sur la même ligne (En valeur de retour, ces fonctions renvoient le nombre de caractères affichés)

Ex :

- `A = 123.456`
- `message = 'valeur de A'`
- `printf("%s = %6.1f\n", message, A)`

→ Valeur de A = 123.4 (suivi d'un retour à la ligne)

Format	Description
%d, %*n*d	Nombre entier signé (positif ou négatif) ((n nombre de chiffres maximum)
%f, **%*n*f,** **%*n.m*f**	Nombre réel sans exposant (n nombre de chiffres maximum avant la virgule, m, nombre après la virgule (forme {-}n.m)
%e %E **%*n*e, %*n*E,** **%*n.m*e, %*n.m*E**	Nombre réel en notation scientifique (forme {-}n.mE{+/-}x)

%c **%*n*c**	Affiche 1(n) caractère(s), y compris les espaces
%s	Affiche une chaîne de caractères
…	Bien d'autres ! (%u, %i, %o, %g…)

Certains caractères sont dits spéciaux (retour à la ligne, tabulation, ..) et sont encodés avec le caractère d'échappement « \ »

Ex :

\n → retour à la ligne

\t → tabulation

\\ → le caractère « \ »

ENTREE DE TEXTE

La manière la plus simple de gérer les entrées au clavier est d'utiliser la commande **input**.

input s'utilise en spécifiant un message à afficher à l'utilisateur (un paramètre optionnel est disponible pour éviter d'évaluer automatiquement les données entrées). Les données entrées doivent être terminées par la touche « entrée »

La forme générale d'utilisation est :

```
variable = input('message') ;
```

ou

```
variable = input('message', 's') ;
```

Ex :

```
Xa = input('Entrez la coordonnée X du point A') ;
```

Nota :

- Il est possible d'effectuer un post-traitement sur les données saisies pour éviter les problèmes de saisie avec différentes fonctions comme **isempty** (variable vide), ou **length** (longueur)
- Les données saisies peuvent être de toute nature : réel, complexe, vecteur, matrices ou tout simplement du texte (paramètre optionnel 's' [string] requis) !)

Deuxième possibilité (à postériori), utiliser les entrées formatées via les fonctions **scanf**, **sscanf** (string **scan** formated) et fscanf (**f**ile **scan** formated) qui permet de décoder le contenu d'une chaîne de caractère et d'en récupérer les données dans un vecteur ou une matrice.

Formes générales :

- `vec = sscanf(chaine, format)`

Décode la chaîne à l'aide du format spécifié et retourne le vecteur-colonne dont tous les éléments seront de même type.

- `mat = sscanf(chaine, format, taille)`

Décode la chaîne à l'aide du format spécifié et retourne une matrice remplie colonne après colonne. La syntaxe de 'taille' s'utilise comme suit : 1 valeur (nb) → permet de lire les n premières valeurs et retourne un vecteur colonne. 2 valeurs [lignes, colonnes] → permet de lire une matrice lignes x colonnes

- `[var1, var2, …] = sscanf(chaine, format,'C')`

Cette syntaxe permet de retourner une série de variables de types différents.

Ex :

```
vec = sscanf('1 2 a b', '%f %f %c %c')
vec = 1
      2
      97
      98
```

```
mat = sscanf('1 2 a b', '%f %f %c %c', [2,2])
mat = 1     97
      2     98
```

```
[a, b, l1, l2] = sscanf('1 2 a b', '%f %f %c %c', 'C')
a = 1
b =2
l1 = a
l2 = b
```

Troisième possibilité laissée par Octave, en utilisant la fonction **scanf** (comme en C) qui permet de lire au clavier des données formatées. (**Attention, cette fonction ne semble pas fonctionner sous Windows**)

- `var1 = scanf(format)`

Ex :

```
a = scanf('%d') ; % lire un entier
```

MANIPULATION DE CHAINES

Il existe de très nombreuses fonctions permettant de manipuler les chaînes de caractères (minuscules → majuscules et inversement, transformation de chaînes en données numériques, etc, etc.). En voici quelques-unes assez utilisées.

Nom de la fonction	Signification
`length(chaine)`	Retourne le nombre de caractères contenu dans « chaine »
`split(chaine, separateur)`	Découpe la chaine selon le séparateur spécifié
`strcmp(chaine1, chaine2)`	Compare les deux chaines entre elles. Retourne 1 (vrai) si elles sont identiques, 0 sinon
`lower(chaine)`	Convertit la chaîne en minuscules
`upper(chaine)`	Convertit la chaîne en majuscules
`str2num(chaine)`	Convertir en nombres le ou les nombres contenus dans la chaîne
…	Et bien d'autres !

ALLER PLUS LOIN : CREER UN MENU

Il est possible de créer un menu (notamment dans un script) pour guider le choix de l'utilisateur parmi plusieurs fonctions disponibles avec la fonction **menu**.

La forme générale d'utilisation est :

```
Choix = menu('Titre du menu', 'libellé du choix 1', 'libellé du choix2', …)
```

Ex :

```
> choix = menu('Mon premier menu', 'Calculer', 'Afficher')

Mon premier menu

  [1] Calculer

  [2] Afficher

pick a number, any number:
```

FICHIERS

Nous avons vu précédemment qu'il était possible de charger et sauver des données dans des fichiers externes à GNU Octave.

D'autres fonctions (similaires à celles utilisées dans le langage C) peuvent permettre de lire et écrire très précisément des données dans un fichier. Les fonctions les plus utiles sont :

- **textread** : permet de lire facilement un fichier texte respectant un format homogène
- **fopen et fclose et feof** : ouvrir et fermer un fichier,
- **fgets, fgetl** : lecture par groupe de caractère (file get string) ou ligne par ligne (file get line)
- **fprinf et fscanf** : lecture et écriture dans un fichier

Pour plus de renseignements, reportez-vous à l'aide en ligne.

8. ENCORE PLUS LOIN AVEC OCTAVE

Comme nous venons de le voir, Octave est un véritable environnement de programmation évolué. Il permet de concevoir, de tester et d'afficher des solutions à des problèmes complexes. Nous allons voir qu'il est possible d'utiliser des types de données encore plus complexes que les vecteurs ou matrices.

TABLEAUX MULTI-DIMENSIONNELS

Au-delà des matrices (n x m), il est bien entendu possible de manipuler des tableaux à N dimensions tout simplement en manipulant un 3ème, 4ème, … Nème indice avec les matrices.

Ex :

- `C(3,3,3) = 1` `// Créé une matrice 3x3x3 initialisée à zéro`
  `// sauf l'élément C(3,3,3) qui vaut 1`

```
C =
ans(:,:,1) =
      0     0     0
      0     0     0
      0     0     0
ans(:,:,2) =
      0     0     0
      0     0     0
      0     0     0
ans(:,:,3) =
      0     0     0
      0     0     0
      0     0     1
```

Certaines fonctions vues pour les vecteurs ou matrices peuvent gérer les tableaux multidimensionnels comme : `zeros`, `ones` et `rand`

Ex :

- `ones(2,4,4)` : créé un tableau 3D de dimension 2x4x4 dont tous les éléments sont mis à 1

Les opérations dont l'un des deux opérandes est un scalaire, les opérateurs arithmétiques, logiques et relationnels ainsi quel opérateurs « élément par élément » fonctionnent avec les matrices multidimensionnelles.

Par contre, les fonctions qui opèrent sur la matrice (2D) et vecteurs [inversion, produit matriciel, …] ne s'appliquent que sur les sous-ensembles de la matrice multidimensionnelle (vecteurs ou matrices 2D). Il est donc impérieux de savoir manipuler les indices !

Quelques fonctions utiles pour manipuler ces tableaux :

- `size(tableau)` → retourne un vecteur-ligne dont le ième élément indique la taille de la ième dimension du tableau
  Ex : `size(C)` renvoie [3 3 3] // lignes, colonnes, « couches »
- `numel(tableau)` → retourne le nombre d'éléments du tableau
  Ex : `numel(C)` renvoie 27 (3x3x3)
- `ndims(tableau)` → retourne la dimension du tableau
  Ex : `ndims(C)` renvoie 3

Nota : Un scalaire et un vecteur est de dimension 2 pour Octave (considérés comme des matrices dégénérées) !

STRUCTURES

Une structure est un type d'objet (appelé aussi enregistrement) qui se composent de « champs » pouvant être de types différents (champs, matrices, …). Les structures sont utiles pour « rassembler » des éléments de types différents (Par exemple, une personne peut être définie par son nom, son prénom (caractères), son numéro de téléphone (nombre), … Ces structures sont fréquemment utilisées dans des tableaux de structures rassemblant plusieurs enregistrements.

Chaque champ peut être accédé en utilisant le séparateur « . ». Par exemple, si la structure « *personne* » est composée des trois champs « *nom* », « *prenom* » et « *telephone* », on accédera au nom en tapant « *personne.nom* » à la fois pour créer une nouvelle structure ou modifier celle-ci.

Ex :

- ```
  personne.nom = 'Truillet'
  personne =
  {
        nom = Truillet
  }
  ```

Autre possibilité, de manière directe, en définissant d'un seul coup tous les champs en utilisant la commande **setfield**.

Pour vérifier le contenu de la structure, il suffit de taper son nom (par exemple ici, en tapant « personne »)

Pour définir un tableau de structure, il suffit d'ajouter un indice à la structure. Par exemple, pour définir un deuxième enregistrement, il suffit de définir « *personne(2).nom* » et ainsi de suite.

TABLEAUX CELLULAIRES

Enfin, il est aussi possible de définir des tableaux cellulaires. Le tableau cellulaire se compose d'objets de types différents (scalaires, matrices, …).

Il est donc le type le plus polyvalent, chaque cellule pouvant être de type différent.

9. CONCLUSION

Comme nous venons de le voir, GNU Octave offre de multiples possibilités en tant que logiciel de calcul mathématique mais est aussi un projet « open-source » toujours vivace et multiplateforme. En perpétuelle évolution, grâce aux ajouts de fonctionnalités (voir `http://sourceforge.net/projects/octave` pour plus de détails), GNU Octave peut devenir rapidement indispensable pour tout ingénieur.

10. BIBLIOGRAPHIE

1. *Introduction to Octave*, Dr. P.JG. Long, version 0.1.0, september 2005, http://www-mdp.eng.cam.ac.uk/web/CD/engapps/octave/octavetut.pdf
2. Introduction à Matlab et GNU Octave, J-D Bonjour, 2008, http://iste.epfl.ch/cours_matlab
3. Wiki Octave, http://wiki.octave.org/Main_Page
  ```