INTRODUCTION A OCTAVE

C. CHARLES*

RÉSUMÉ

Cette note technique est une initiation à GNU Octave, logiciel de haut niveau pour le calcul numérique. Son objectif est d'apprendre au lecteur à utiliser ses fonctionnalités de base. Ainsi, après la lecture de cette introduction, il sera capable de manipuler dans Octave des vecteurs, matrices, chaînes de caractères, tableaux et structures. Il sera également à même de compléter le large panel des fonctions d'Octave, en réalisant ses propres programmes et fonctions.

SUMMARY

This note aims to introduce GNU Octave, high level language, primarily intended for numerical computations. The objective is to make known its basic functionnalities. Therefore, after reading this note, the reader will be able to handle with vectors, matrices, chains, tabulars and structures. He also will be able to conceive his own scripts and functions.

1. INTRODUCTION

1.1. Historique

Aujourd'hui, le recours à l'analyse numérique s'est généralisé. Dans ce domaine, le logiciel Matlab (**Mat**rix **Lab**oratory), dont l'origine remonte à la fin des années 70, est devenu presque incontournable. Son succès s'explique en particulier par ses nombreuses fonctions dans des domaines variés ainsi que par son langage de programmation simple basé sur les matrices. Parallèlement à cette évolution, le mouvement en faveur des logiciels libres a pris une grande ampleur. C'est ainsi que des logiciels analogues à Matlab ont vu le jour, GNU Octave étant celui qui offre actuellement le meilleur degré de compatibilité par rapport à Matlab. Conçu en 1988 en tant que logiciel spécifique dans le cadre d'un cours

*Chargée de cours à la Faculté universitaire des Sciences agronomiques de Gembloux (Unité de Statistique, Informatique et Mathématique appliquées)

sur les réacteurs chimiques, Octave porte le nom d'un professeur, auteur de manuels basés sur ce logiciel [LEVENSPIEL, 1962]. Il a été repensé dès 1992 par J. W. Eaton pour être un logiciel de calcul numérique plus général et plus flexible. La version 1.0 de GNU Octave est sortie en 1994.

1.2. Caractéristiques

Octave n'est pas un logiciel de calcul algébrique ou symbolique mais est un logiciel de calcul numérique, de visualisation et de programmation très performant.

Ses données de base sont des matrices, pouvant évidemment se réduire à des vecteurs et des scalaires. Ceci veut dire que, pour être rapide, un programme doit privilégier les commandes matricielles plutôt que les boucles. Par exemple, si A est une matrice de nombres à 1.000 lignes et 1.000 colonnes, la commande `A=A/2` divisera par deux chaque élément de la matrice A en 100 fois moins de temps que les deux boucles imbriquées

```
for i=1 :1000 for j=1 :1000 a(i,j)=a(i,j)/2 ; end, end
```

utilisées de façon similaire dans les autres langages de programmation.

Octave possède un langage interprété, ce qui veut dire qu'il n'y a pas besoin de compilation avant l'exécution d'un programme Octave. De plus, Octave ne demande pas de déclarations. Ces deux caractéristiques facilitent l'apprentissage d'Octave.

Octave est aussi un environnement de développement à part entière : son langage de haut niveau, doté de boucles et tests, de fonctions, de visualisation 2D et 3D, etc., permet à l'utilisateur d'élaborer ses propres fonctions, ainsi que de véritables programmes.

Octave est un logiciel libre. Il n'occasionne aucun coût. Il est disponible sur les trois plates-formes majeures que sont Windows, Linux et Mac. Octave fait partie du projet GNU et est distribué sous licence GPL (General Public License).

Développé sous Unix/Linux et dans l'esprit de ce système d'exploitation, Octave ne présente pas le caractère très exclusif de Matlab. Au contraire, plutôt que d'intégrer toutes les fonctionnalités dans un logiciel, Octave coopère avec d'autres outils complémentaires. Ainsi, il ne dispose pas de son propre moteur graphique mais s'appuie sur Gnuplot.

1.3. Plan

Après cette introduction, le paragraphe 2 concerne l'installation d'Octave ainsi que ses notions de base (espace de travail, chemin d'accès, caractères spé-

ciaux et opérateurs). Le paragraphe 3 s'intéresse aux différents objets manipulés par Octave : matrices, chaînes, tableaux et structures. Le paragraphe 4 explique les rudiments nécessaires pour concevoir ses propres scripts ainsi que ses propres fonctions. Enfin, avant de conclure, le paragraphe 5 présente les principales commandes graphiques et le paragraphe 6 illustre la large gamme de sujets traités par Octave.

2. INSTALLATION ET NOTIONS DE BASE

2.1. Installation

Le téléchargement se fait à partir du site

http ://www.gnu.org/software/octave/download.html.

A la base, le progiciel Octave a été conçu et développé sous Unix/Linux. Mais il peut également être installé sous Windows ou Mac.

Une fois installé, pour lancer Octave sur une machine Windows, on trouve le raccourci de démarrage dans le menu Démarrer > Programmes. Dans sa version de base, Octave n'offre qu'une fenêtre de commandes.

2.2. Notions de base

2.2.1. Exemples

Ci-dessous se trouvent deux exemples de programmation Octave. Ils vous permettront de mieux comprendre la suite de la note. Le premier calcule les dix premiers nombres de Fibonacci. Le deuxième illustre la formule $\mathbf{Ax}=\lambda\mathbf{x}$ liant les valeurs propres et les vecteurs propres d'une matrice $\mathbf{A}$.

Exemple 1

```
fib=ones(1,2) ;
i=3 ;
while (i<=10) fib(i)=fib(i-1)+fib(i-2), i=i+1 ; end
```

Résultat de l'exemple 1

```
fib = 1 1 2
fib = 1 1 2 3
fib = 1 1 2 3 5
fib = 1 1 2 3 5 8
fib = 1 1 2 3 5 8 13
fib = 1 1 2 3 5 8 13 21
fib = 1 1 2 3 5 8 13 21 34
fib = 1 1 2 3 5 8 13 21 34 55
```

Exemple 2

```
A=[2 -2 ; 1 -1]
[vecp,valp]=eig(A) ;
x=A*vecp( :,1) ;
y=valp(1,1)*vecp( :,1) ;
% affichage
disp("x"),disp(x)
disp("y"),disp(y)
```

Résultat de l'exemple 2

$$A = \begin{matrix} 2 & -2 \\ 1 & -1 \end{matrix}$$

x

0.89443
0.44721

y

0.89443
0.44721

Ces commandes peuvent être encodées directement dans la fenêtre de commandes ou dans un fichier texte d'extension ".m". Dans ce cas, il faut appeler le fichier depuis la fenêtre de commandes, comme nous le verrons en détails par la suite (paragraphe 4.3.).

2.2.2. Espace de travail

Les variables créées au cours d'une session Octave interactivement depuis la fenêtre de commandes résident en mémoire et constituent ce qu'on appelle le "workspace" c'est-à-dire l'espace de travail. A moins d'être sauvegardées sur disque dans un "mat-file", les variables sont perdues lorsque la session est terminée.

Les mat-files sont des fichiers binaires de variables qui n'ont pas d'extension. Il ne faut pas les confondre avec les "m-files" qui sont des fichiers texte de scripts ou de fonctions qui ont l'extension *.m. La commande save mat-file (save mat-file variables resp.) sauvegarde toutes les variables définies et présentes en mémoire (ou seulement les variables spécifiées resp.) dans le mat-file. La commande load mat-file (load mat-file variables resp.) charge en mémoire toutes les variables (ou seulement les variables spécifiées resp.) du mat-file spécifié. Lorsqu'il s'agit d'échanger des données entre Octave et d'autres logiciels, les mat-files ne conviennent pas car ce sont des fichiers binaires. Une solution consiste à utiliser les commandes save et load présentées ci-dessus avec l'option -text ou -ascii pour sauvegarder ou charger les variables sous forme de fichiers-texte (ASCII). Avec la commande variable=load('fichier-texte'), les données du fichier-texte sont chargées sur la variable de nom spécifié et non pas sur une variable de nom identique au nom du fichier-texte.

Octave ne nécessite aucune déclaration préalable de type de variable et de dimension de tableau. Lorsque Octave rencontre un nouveau nom de variable, il crée automatiquement la variable et y associe l'espace de stockage approprié dans l'espace de travail. Si la variable existe déjà, Octave change son contenu et, si nécessaire, lui alloue un nouvel espace de stockage. Un nom de variable valide consiste en une lettre suivie d'un nombre quelconque de lettres, chiffres ou soulignés.

Octave possède une série de variables dites "built-ins". Celles-ci ont des valeurs par défaut (comme pi, inf pour l'infini ou nan pour **n**ot **a** **n**umber c'est-à-dire un résultat indéterminé du style 0/0). L'utilisateur peut modifier ces valeurs par défaut s'il le souhaite. La liste de ces variables built-ins s'obtient avec la commande who -builtins. La commande variable=expression affecte à "variable" le résultat de l'expression et affiche celui-ci. Par exemple, la commande i=3+2 affecte à i la valeur 5 et affiche i=5. La commande expression affecte à la variable de nom prédéfini "ans" (pour answer) le résultat de l'expression. Par exemple, la commande 3+2 affiche ans=5. La commande variable affiche le contenu de la variable spécifiée. Pour notre exemple, la commande i affichera i=5. La commande string = 'chaîne de caractères' enregistre "chaîne de caractères" dans la variable "string". Si la chaîne contient une apostrophe, il faut la dédoubler. La commande clear (clear variables resp.) efface tout (les variables spécifiées resp.) de l'espace de travail.

Octave travaille toujours en double précision (32 bits). Le format d'affichage des nombres dans la fenêtre de commandes se fait à l'aide des commandes suivantes. La commande format short affiche une notation décimale fixe à 5 chiffres significatifs. Avec l'option e (commande format short e), Octave utilise la notation décimale flottante avec exposant. Les 5 chiffres significatifs concernent la mantisse. La commande format long (e) est la même que la précédente sauf qu'elle utilise 15 chiffres significatifs.

2.2.3. Chemin d'accès

Octave procède de la façon suivante lorsqu'il évalue les commandes, fonctions et expressions passées par l'utilisateur depuis la fenêtre de commandes. Si on tape abc, Octave cherche d'abord s'il existe une variable nommée abc dans l'espace de travail. S'il n'a rien trouvé, il cherche si abc est une fonction d'Octave. Si tel n'est pas le cas, il recherche un m-file nommé abc.m (script ou fonction) dans le répertoire courant de l'utilisateur. Enfin, si cette recherche n'a pas donné de résultat, il parcourt dans l'ordre les différents répertoires définis dans le chemin d'accès, aussi appelé path de recherche, afin de trouver un m-file nommé abc.m. Cet ordre de recherche entraîne que les définitions de variables réalisées par l'utilisateur priment sur les fonctions d'Octave. Donc, pour avoir accès à toutes les fonctions d'Octave, il ne faut jamais créer de variables ayant le même nom qu'une des fonctions d'Octave.

Le path de recherche indique le chemin d'accès aux différents répertoires où se trouvent les scripts et fonctions (m-files) invoqués par l'utilisateur. Ce chemin d'accès est défini dans deux variables : DEFAULT_LOADPATH (répertoires où sont définies les fonctions d'Octave) et LOADPATH (répertoires où sont définies les fonctions de l'utilisateur). La commande path affiche le chemin d'accès. La commande addpath('chemins') ajoute au début du chemin d'accès les chemins spécifiés. La commande rmpath('chemins') supprime du chemin d'accès les chemins spécifiés.

2.2.4. Caractères spéciaux

Dans le paragraphe 2.2.1., le lecteur a pu constater que certaines commandes Octave sont suivies d'un point-virgule et d'autres pas. En réalité, non suivie du point-virgule, une commande sera normalement exécutée et son résultat sera affiché. A contrario, suivie d'un point-virgule, une commande sera normalement exécutée mais son résultat ne sera pas affiché.

Dans le premier exemple du paragraphe 2.2.1., la boucle while est écrite sur une seule ligne. On remarque que la commande fib(i)=fib(i-1)+fib(i-2) est suivie d'une virgule. En effet, la virgule est utilisée comme séparateur de commandes lorsque l'on souhaite passer plusieurs commandes sur la même ligne et qu'on ne souhaite pas mettre de point-virgule. Les trois points "..." sont utilisés en fin de ligne lorsque l'on veut continuer une instruction sur la ligne suivante.

Le symbole % indique que ce qui suit est considéré comme un commentaire. Ceci est illustré dans l'exemple 2.

Notons enfin que les espaces ne sont pas pris en compte par Octave et qu'Octave distingue les majuscules des minuscules.

2.2.5. Opérateurs et fonctions de base

Le tableau 1 reprend les opérations de base appliquées à des scalaires tandis que le tableau 2 liste les fonctions élémentaires appliquées aux scalaires.

+	addition
-	soustraction
*	multiplication
/	division
^	puissance
**	puissance

Tableau 1. Opérateurs de base (scalaires).

sqrt(var)	racine carrée de var
exp(var)	exponentielle de var
log(var)	logarithme népérien de var
log2(var)	logarithme en base 2 de var
log10(var)	logarithme en base 10 de var
cos(var)	cosinus de var (angle exprimé en radian)
acos(var)	arc cosinus de var
sin(var)	sinus de var
asin (var)	arc sinus de var
tan(var)	tangente de var
atan(var)	arc tangente de var
abs(var)	valeur absolue de var
round(var)	entier le plus proche de var
floor(var)	partie entière inférieure de var
ceil(var)	partie entière supérieure de var
mod(x,y)	reste de la division entière de x par y
sign(var)	vaut -1 pour les négatifs, 1 pour les positifs et 0 pour 0.

Tableau 2. Fonctions de base (scalaires).

3. OBJETS

3.1. Série

Pour créer une série numérique linéaire débutant par la valeur début, autoincrémentée de 1 et se terminant par la valeur fin, il faut utiliser la commande début : fin. Elle crée donc un vecteur ligne. Pour créer une série numérique linéaire débutant par la valeur début, incrémentée ou décrémentée du pas spécifié et se terminant par la valeur fin, il faut utiliser la commande début :pas :fin. Par exemple, la commande p=0 :2 :10 retourne p = 0 2 4 6 8 10.

3.2. Vecteur

Octave ne fait pas vraiment de différence entre une matrice, un vecteur et un scalaire, étant donné que ces éléments peuvent être redimensionnés dynamiquement. Une variable vecteur n'est donc qu'une matrice dégénérée en une seule ligne ou une seule colonne. Les éléments du vecteur sont numérotés par des entiers débutant par la valeur 1. Les commandes intéressantes à propos des vecteurs sont reprises dans le tableau 3.

3.3. Matrice

Une matrice est un tableau rectangulaire à n lignes et m colonnes de nombres réels ou complexes ou encore de caractères. Les indices de ligne et de

vec=[v1 v2 v3]	création d'un vecteur ligne contenant les valeurs v1, v2 et v3
vec=[v1 ; v2 ; v3]	création d'un vecteur colonne contenant les valeurs v1, v2 et v3
vec'	transposée du vecteur vec
vec=début :pas :fin	affectation de la série linéaire à vec
vec(i)	désigne le ième élément du vecteur vec
vec(i :p :j)	désigne les élements d'indice i à j avec un pas de p
vec(i :j)=[]	destruction des éléments i à j du vecteur vec (qui est redimensionné)
length(vec)	retourne le nombre d'éléments du vecteur vec
vec1 + vec2	addition (vecteurs de même dimension)
vec1 - vec2	soustraction (vecteurs de même dimension)
vec1 * vec2	produit (nombre colonnes vec1 = nombre lignes vec2)
norm(vec)	calcule la norme du vecteur vec
dot(vec1,vec2)	calcule le produit scalaire des vecteurs vec1 et vec2
cross(vec1,vec2)	calcule le produit vectoriel des vecteurs vec1 et vec2
min(vec)	retourne le plus petit élément du vecteur
max(vec)	retourne le plus grand élément du vecteur
sum(vec)	retourne la somme des éléments du vecteur
prod(vec)	retourne le produit des éléments du vecteur
sort(vec)	retourne vec trié par ordre croissant
mean(vec)	retourne la moyenne arithmétique des éléments du vecteur
std(vec)	retourne l'écart-type des éléments du vecteur
var(vec)	retourne la variance des éléments du vecteur
median(vec)	retourne la médiane des éléments du vecteur
any(vec)	retourne 1 si au moins un élément de vec est différent de 0
all(vec)	retourne 1 si tous les éléments de vec sont différents de 0.

Tableau 3. Commandes appliquées aux vecteurs.

colonne sont des valeurs entières débutant par 1. Les commandes intéressantes à propos des matrices sont listées dans les tableaux 4 et 5.

3.4. Chaîne de caractères

Octave stocke les chaînes de caractères sous forme de vecteur ligne dans lequel chaque caractère est un élément du vecteur. La commande string='chaîne de caractères' enregistre la chaîne de caractères définie entre apostrophes sur la variable string qui est ici un vecteur ligne. Si la chaîne contient une apostrophe, il faut la dédoubler.

mat=[v11 v12 v13 ; v21 v22 v23]	création d'une matrice 2 lignes 3 colonnes contenant les valeurs mentionnées
mat=[vec1 vec2]	construit la matrice mat par concaténation des vecteurs colonnes vec1 et vec2
mat=[vec1 ; vec2]	construit la matrice mat par concaténation des vecteurs lignes vec1 et vec2
ones(n,m)	renvoie une matrice n lignes m colonnes dont tous les éléments valent 1
zeros(n,m)	renvoie une matrice n lignes m colonnes dont tous les éléments valent 0
eye(n,m)	renvoie une matrice identité n lignes m colonnes dont les éléments diagonaux valent 1 et les autres 0
mat'	transposée de la matrice mat
mat(i,j)	désigne l'élément de la matrice mat se trouvant à la ième ligne et à la jème colonne
mat(i :j,k :m)	désigne la partie de la matrice dont les éléments se trouvent dans les lignes i à j et dans les colonnes k à m
mat(i, :)	désigne la ligne i
mat(:,j)	désigne la colonne j
mat(i :j, :)=[]	destruction des lignes i à j de la matrice (qui est redimensionnée)
$[n, m]$=size(mat)	retourne le nombre de lignes et de colonnes de la matrice.
A+B	addition matricielle (A et B matrices de même dimension)
A-B	soustraction matricielle (A et B matrices de même dimension)
A*B	produit matriciel (nombre colonnes de A = nombre lignes de B)
A.*B	produit élément par élément (A et B matrices de même dimension)
A\B	division matricielle à gauche (X=A\B est la solution de A*X=B)
A/B	division matricielle à droite (X=A/B est la solution de X*A=B)
inv(mat)	inversion de la matrice carrée mat
det(mat)	retourne le déterminant de la matrice carrée mat
rank(mat)	retourne le rang de la matrice mat.

Tableau 4. Commandes générales appliquées aux matrices.

3.5. Tableau multidimensionnel

Les tableaux multidimensionnels concernent des matrices à plus de deux indices. Toutes les commandes vues pour les matrices se généralisent facilement aux tableaux multidimensionnels. Par exemple, mat(i,j,k) désigne l'élément de la

min(mat)	retourne un vecteur ligne contenant le plus petit élément de chaque colonne
max(mat)	retourne un vecteur ligne contenant le plus grand élément de chaque colonne
min(mat,2)	retourne un vecteur colonne contenant le plus petit élément de chaque ligne
max(mat,2)	retourne un vecteur colonne contenant le plus grand élément de chaque ligne
norm(mat,p)	retourne la p-norme de la matrice mat
sum(mat)	retourne un vecteur ligne contenant la somme de chaque colonne
mean(mat)	retourne un vecteur ligne contenant la moyenne de chaque colonne
std(mat)	retourne un vecteur ligne contenant l'écart-type de chaque colonne
sum(mat,2)	retourne un vecteur colonne contenant la somme de chaque ligne
mean(mat,2)	retourne un vecteur colonne contenant la moyenne de chaque ligne
std(mat,2)	retourne un vecteur colonne contenant l'écart-type de chaque ligne.
cond(mat)	retourne le conditionnement de mat
$[v,l]$=eig(mat)	retourne les vecteurs propres (v) et les valeurs propres (l) de mat
poly(mat)	renvoie un vecteur contenant les coefficients du polynôme caractéristique associé à mat
$[u,s,v]$=svd(mat)	retourne la décomposition en valeurs singulières de mat
$[l,u]$=lu(mat)	retourne la décomposition lu de mat
chol(mat)	retourne le facteur de la décomposition de cholesky de mat
$[q,r]$=qr(mat)	retourne la décomposition qr de mat.

Tableau 5. Commandes spécifiques appliquées aux matrices.

ième ligne, de la jème colonne et du kème tableau de mat, tableau tridimensionnel.

3.6. Structure

Une structure est un type d'objet d'Octave se composant de plusieurs champs qui peuvent être de types différents (chaînes, matrices,...). On accède aux champs d'une structure avec la syntaxe structure.champ.

Imaginons que nous voulions créer une nouvelle structure appelée personne avec trois champs, à savoir nom, prénom, âge. La création de la structure se fait

en définissant les attributs du premier individu. Par exemple,

```
personne.nom='dupont'
personne.prenom='pierre'
personne.age='30'
```

La commande personne permet alors de vérifier le contenu de la structure. Si on veut définir un deuxième individu, il faut utiliser les commandes précédentes en ajoutant (2) après personne. Par exemple, personne(2).nom='leclerc'. Avec personne(1), on récupère la structure complète correspondant à la première personne. Avec personne([1 3]), on retrouve un tableau de structures contenant la première et la troisième personne. Avec personne(1).age, on obtient l'âge de la première personne.

3.7. Tableau cellulaire

Le tableau cellulaire est un tableau bidimensionnel qui peut se composer d'objets de types différents (chaîne, matrice, structure,...). La différence par rapport au tableau ordinaire est l'utilisation d'accolades à la place de parenthèses.

Par exemple,

```
T{1,1}= 'bonjour'
T{1,2}=[1 2; 3 4]
```

définit un tableau cellulaire 1 ligne 2 colonnes. Pour accéder à l'élément 1,1, on utilise la commande T{1,1}.

4. PROGRAMMATION

Les m-files sont des fichiers-texte, créés et édités avec n'importe quel éditeur, qui ont l'extension *.m et qui contiennent des instructions Octave. Octave étant un langage interprété, les m-files n'ont pas besoin d'être compilés pour être utilisés. On distingue deux types de m-files : les scripts (ou programmes) et les fonctions. Les scripts manipulent directement les variables de l'espace de travail (variables dites globales) alors que les fonctions agissent par défaut sur les variables locales à la fonction (mis à part les arguments d'entrée et sortie).

4.1. Interaction avec l'utilisateur

Dans un script ou une fonction, on peut avoir besoin de donner de l'information à l'utilisateur ou de lui en demander. Les principales commandes d'interaction avec l'utilisateur sont reprises dans le tableau 6.

disp(variable)	affiche le contenu de la variable
disp('bonjour')	affiche bonjour
variable=input('Donner un chiffre')	Octave affiche "Donner un chiffre", attend que l'utilisateur entre quelque chose au clavier et appuie sur enter. Octave met l'information donnée par l'utilisateur dans variable.
choix=menu('n est','pair','impair')	affiche le titre "n est", puis un menu avec deux options : (1) pair (2) impair. Il faut répondre en tapant 1 ou 2. Le numéro entré est renvoyé dans la variable choix.
error("n est pair")	affiche comme message d'erreur "n est pair"
return	quitte une fonction.

Tableau 6. Commandes d'interaction avec l'utilisateur.

4.2. Boucles et tests

Octave utilise des boucles et tests similaires à ceux des autres langages de programmation (tableau 7).

Dans la boucle while et le test if, apparaissent des expressions logiques. Dans ces cas, la syntaxe à utiliser s'inspire du tableau 8. L'exemple 1 de la section 2.2.1. en est une illustration pour la boucle while.

4.3. Scripts

Un script n'est rien d'autre qu'une suite de commandes sauvegardées dans un m-file. Par opposition aux fonctions, les scripts sont invoqués par l'utilisateur sans passer d'arguments, car ils opèrent directement sur les variables de l'espace de travail. Un script peut donc lire et modifier les variables préalablement définies, ainsi que créer de nouvelles variables qui seront accessibles dans l'espace de travail une fois le script exécuté. Pour exécuter un script, on frappe dans la fenêtre de commandes son nom (sans l'extension ".m") suivi de enter.

4.4. Fonctions

Egalement programmées sous forme de m-file, les fonctions se distinguent des scripts par leur mode d'invocation depuis la fenêtre de commandes qui est fondamentalement différent :

[var-sortie1,var-sortie2,...]=fonction(var1,var2,...)

boucle for for i=d :p :f commandes end	 pour i allant de d à f par pas de p exécution de commandes
boucle while while expression-logique commandes end	 tant que expression-logique est vraie exécution de commandes
test if expr-log-1 commandes1 elseif expr-log-2 commandes2 else commandes end	 si expr-log-1 est vraie, exécution de commandes1 autrement, si expr-log-2 est vraie, exécution de commandes2 sinon, exécution de commandes
switch-case switch var case (v1), commandes1 case (v2), commandes2 otherwise commandes end	 si var=v1, exécution commandes1 si var= v2, exécution commandes2 autrement, exécution commandes
sortie prématurée d'une boucle break	

Tableau 7. Boucles et tests.

==	test d'égalité
~=	test de différence
<	test d'infériorité
>	test de supériorité
<=	test d'infériorité ou égalité
>=	test de supériorité ou égalité
~expression	négation logique
expression1 & expression2	ET logique
expression1 | expression2	OU logique

Tableau 8. Expressions logiques.

Comme on peut le voir, on appelle une fonction par son nom en lui passant ses arguments d'entrée entre parenthèses. La fonction retourne une (des) valeur(s) de sortie que l'on récupère par la (les) variable(s) à laquelle la fonction est affectée lors de l'appel. Le nom du m-file doit être rigoureusement identique au nom de la fonction.

Le fichier fonction.m doit commencer en première ligne par la déclaration de la fonction qui est

function [arg-sortie1,arg-sortie2,...]=fonction(arg-entrée1,arg-entrée2,...)

Les commandes viennent après cette déclaration.

Le mécanisme de passage des paramètres se fait par valeur. Les variables créées à l'intérieur de la fonction sont dites locales car elles sont inaccessibles en dehors de la fonction. Si l'on tient cependant à ce que certaines variables de la fonction soient visibles et accessibles à l'extérieur, on peut les rendre globales en les définissant comme telles dans la fonction, avant qu'elles ne soient utilisées, par une déclaration

global variable

Cette déclaration doit venir après la déclaration de la fonction qui est en première ligne du fichier. Il faudra aussi faire une telle déclaration dans la fenêtre de commandes si on veut accéder à ces variables dans l'espace de travail.

Voici un exemple de fonction, écrit dans le fichier factorielle.m :

```
function rep=factorielle(n)
rep=1;
if (n<0)
error("pas de factorielles pour les négatifs");
endif
if (n==0)
return;
else
for i=2:n
 rep=rep*i;
endfor
endif
end
```

Pour l'appeler, on tape par exemple f=factorielle(5) dans la fenêtre de commandes. Octave retourne f=120.

5. GRAPHIQUES

Alors que Matlab, logiciel commercial, ne compte que sur lui-même, Octave s'appuie davantage sur des outils externes, ce qui est notamment le cas pour les fonctionnalités graphiques. Octave n'intègre donc pas de moteur graphique mais utilise par défaut le logiciel de visualisation gratuit Gnuplot.

Nous décrivons ici les commandes à encoder dans Octave (sous Windows) pour obtenir un graphique dans la fenêtre des graphiques de Gnuplot (intitulée

"gnuplot graph") qui s'ouvre automatiquement. Ces fonctionnalités peuvent différer légèrement sous Linux. Lorsqu'un programme Octave contenant des commandes graphiques est lancé sous Windows, non seulement la fenêtre gnuplot graph apparaît mais également une fenêtre de commandes Gnuplot (console). Celle-ci est à l'état "réduit" dans la barre des tâches. Dans celle-ci défilent les commandes Gnuplot générées par les ordres graphiques du programme Octave. Ces deux fenêtres ne doivent pas être fermées manuellement. Pour fermer proprement ces deux fenêtres de Gnuplot, il faut utiliser la commande closeplot.

Tout d'abord, nous présentons les commandes graphiques relatives au graphe en dimension 2 (tableau 9).

plot(x1,y1,x2,y2)	dessine un graphique avec deux courbes. xi contient les abscisses de la ième courbe yi contient les ordonnées de la ième courbe. Chaque paire de vecteurs (xi,yi) doit avoir le même nombre d'éléments. Celui-ci peut être différent d'une paire à l'autre.
plot(vec)	équivaut à plot(x,vec) avec x=[1 2 3 4...]
fplot('f',[$xmin$ $xmax$])	trace la fonction f entre les limites xmin et xmax. La fonction spécifiée est une fonction d'Octave (par exemple 'sin') ou une fonction de l'utilisateur (par exemple 'sqrt(3x+2)+x').
stairs(x,y)	dessine une ligne en escalier pour la courbe définie par les vecteurs x et y
scatter(x,y,size,color)	dessine le semis de points définis par les coordonnées (x,y). size spécifie la taille des points et color leur couleur.
pie(val)	dessine un camembert 2d sur base du vecteur val
bar(x,y)	dessine les barres verticales définies par les vecteurs x (position) et y (hauteur)
hist(y,n)	détermine la répartition des valeurs de y selon n catégories de même largeur puis dessine cette répartition sous forme d'histogramme.

Tableau 9. Commandes graphiques en dimension 2.

Ces commandes graphiques se généralisent facilement pour les graphes en 3 dimensions (tableau 10).

Les graphiques obtenus par ces commandes en dimensions 2 ou 3 peuvent être améliorés en jouant sur les axes, en introduisant une légende, un titre, etc. Ceci justifie la présentation des commandes suivantes. Elles doivent être placées après la commande graphique.

Dans les commandes graphiques, on peut utiliser une option spécifiant le

plot3(x,y,z)	dessine une ligne passant par les points (x,y,z)
[xm, ym]=meshgrid(x,y) ; z=f(xm,ym) surf(x,y,z)	permet de calculer f(x,y) en tous les points du maillage défini par x et y sans utiliser de boucle for et de dessiner la surface correspondante.

Tableau 10. Commandes graphiques en dimension 3.

axis([$xmin\ xmax\ ymin\ ymax$])	Les limites inférieures et supérieures des axes sont déterminés automatiquement par les commandes graphiques. Cette commande-ci est utilisée lorsqu'on ne veut pas que ce soit automatique.
xlabel('labelx')	définit et affiche le texte de la légende de l'axe des x
ylabel('labely')	définit et affiche le texte de la légende de l'axe des y
titre('titre')	définit et affiche un titre de graphique
text(x,y,'mot')	définit l'annotation mot qui est placé sur le graphique aux coordonnées spécifiées
legend('mot1','mot2')	met la légende mot1 pour le premier graphe et la légende mot2 pour le deuxième graphe.

type et la couleur du trait ainsi que le symbole utilisé. Cette option se définit comme une combinaison de caractères définis ci-dessous.

m	magenta
c	bleu foncé
r	rouge
g	vert clair
k	brun
-	ligne continue
- -	petits traits
:	ligne pointillée
-.	ligne trait point
o	losange
*	étoile
+	plus
x	croix

Par exemple, plot(x,y1,'r-o',x,y2,'m :*') dessine la première courbe en trait continu de couleur rouge et avec un losange en les points et la deuxième courbe en ligne pointillée de couleur magenta et avec une étoile en les points. On peut bien sûr aller beaucoup plus loin et créer soi-même sa table de couleurs et modifier les attributs de lignes et symboles avec la technique "Handle Graphics" non vue dans cette introduction. Il est également possible de faire des animations, d'importer des images,... Pour cela, nous vous référons à la bibliographie.

Par défaut, Octave envoie tous les ordres graphiques à la même fenêtre graphique appelée figure. Chaque fois que l'on dessine un nouveau graphique, celui-ci écrase le précédent. Si l'on désire tracer plusieurs graphiques, les commandes utiles se trouvent dans le tableau 11.

hold('on')	superpose les ordres de dessin qui suivent dans la même figure
subplot(l,c,i)	découpe la fenêtre graphique courante en l lignes et c colonnes et sélectionne la ième comme espace de tracé courant
figure	ouvre une nouvelle fenêtre de graphique et en fait la fenêtre active
figure(i)	en fait la fenêtre active, si la fenêtre i existe ; sinon ouvre une nouvelle fenêtre portant ce numéro
clf	efface le graphique de la fenêtre courante.

Tableau 11. Commandes graphiques.

Les commandes graphiques décrites dans cette section agissent immédiatement sur le graphique courant si la variable automatic_replot vaut 1 (par défaut). Si cette variable vaut 0, il faut passer la commande replot pour voir l'effet de ces fonctions.

6. AUTRES COMMANDES

Octave possède encore de nombreuses commandes qui permettent de résoudre des problèmes dans des domaines tout à fait différents. Le nombre de commandes existantes est très important. Le tableau 12 présente un minuscule échantillon de commandes permettant d'illustrer la large gamme de sujets traités par Octave.

fsolve(function,x0)	trouve la racine de function avec pour point initial x0
quad(f,a,b)	intègre f entre a et b
lsode(f,x0,t0)	résoud l'équation différentielle dx/dt=f(x,t) avec comme points initiaux t0 et x0
ols(y,x)	moindres carrés
anova(y,g)	test statistique anova
interp1(x,y,xi)	interpole la fonction passant par (x,y) aux points définis par xi
...	...

Tableau 12. Autres commandes.

7. EN GUISE DE CONCLUSION

Le principal avantage d'Octave (et de Matlab) est sa rapidité due à son langage "matriciel". Il est donc important d'en tenir compte lors de la programmation et de privilégier les commandes matricielles plutôt que les boucles (for ou while). Pour un gain de temps, il est également utile de dimensionner les matrices en les initialisant. Les autres avantages rencontrés par Octave sont la facilité d'apprentissage du langage et sa richesse, ainsi que la gratuité du logiciel.

Cette très brève introduction à Octave a présenté les notions de base. Cependant, ce document est loin d'être complet. Pour obtenir de plus amples informations, l'utilisateur consultera les références se trouvant dans la bibliographie ou sur le site principal consacré à Octave :

http ://www.gnu.org/software/octave/docs.html.

Pour obtenir de l'aide en ligne, la commande help fonction affiche dans la fenêtre de commandes la description et la syntaxe de la fonction. Passée sans paramètre, la commande help liste les rubriques d'aides principales. Enfin, la commande help -i motclé donne des informations concernant le "motclé".

BIBLIOGRAPHIE

BONJOUR J.D. [1999]. *Introduction à Matlab et GNU Octave*. Support du cours "Informatique de l'ingénieur" pour les étudiants en Sciences et ingénierie de l'environnement de l'ENAC, EPFL à Lausanne.
(disponible sur http ://enacit1.epfl.ch/cours_matlab/ consulté le 21/03/2008)

EATON J.W. [2005]. *GNU Octave Manual*. Network Theory Limited. 324pp.

LEVENSPIEL O. [1962]. *Chemical Reaction Engineering*. New York Wiley. 573pp.

O'DONOVAN B. [2004]. *GNU Octave : une introduction*. Gazette Linux 109.

O'DONOVAN B. [2005]. *GNU Octave : fonctions et scripts*. Gazette Linux 112.