UNIVERSITÉ ABOU BAKR BEL-KAID–TLEMCEN
FACULÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
Première année licence

Outils de Programmation pour les mathématiques

Mme HABRI née BENMAHDI Meryem Bochra

Année universitaire: 2022-2023

Présentation

- **Nom , Prénom: Mme HABRI née BENMAHDI Meryem Bochra**
- **Spécialité: Réseaux et Systèmes Distribués (RSD).**
- **Contact: benmahdibouchra@gmail.com.**
- **Disponibilité: Tous les lundis de 13 à 13:30 au département d'informatique.**

Présentation de la matière

- **Nom de la matière: Outils de Programmation pour les mathématiques.**
- **Unité : Méthodologique**
- **Découpage du cours: 1,5 de CM et 1,5 TP**
- **Coefficient: 1**
- **Nombre de crédits: 2**
- **Mode d'évaluation: Examen (60%) et contrôle (40%)**
- **Note du contrôle(non rattrapable): note de la présence en TP (/5)+ note du contrôle (15).**
  - **Chaque absence en TP est sanctionnée par un point, chaque étudiant a droit à 3 absences non justifier**
- **Prérequis recommandé: notion de base de programmation.**

Contenu de la matière

Chapitre1 : Maîtrise de Logiciels (Matlab, GNU Octave, Scilab, mathématica,..)

Chapitre 2 : Exemples d'applications et techniques de résolution

Objectif de la matière

- **Connaitre Matlab et GNU Octave**
- **Comprendre les techniques de résolution dans GNU Octave**
- **Programmer sous GNU Octave**

Règle de fonctionnement du cours

- **La présence en cours n'est pas obligatoire mais les étudiants qui assistent peuvent avoir des points bonus.**
- **Les étudiants qui ne sont pas intéressés par le cours, je leur demande de ne pas venir perturber mon cours.**
- **La présence à la séance de TP est obligatoire et les étudiants qui s'absentent seront sanctionner.**

Règle de fonctionnement du cours

- **L'engagement dans le cours : La participation des étudiants au cours et au TP peut être recomposer par des points en plus.**
- **Le retard.**
- **Quitter le cours.**
- **Les étudiants désirant poser des questions peuvent le faire de façon ordonnée.**

UNIVERSITÉ ABOU BAKR BEL-KAID–TLEMCEN
FACULÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
Première année licence

Outils de Programmation pour les mathématiques

Cours 1
Présentation et prise en main de GNU Octave

Mme HABRI née BENMAHDI Meryem Bochra

Année universitaire: 2022-2023

Plan du cours

- **Présentation de Matlab**
- **Présentation de GNU Octave**
- **Outils semblables à Matlab et à Octave**
- **Caractéristique de Matlab et de GNU Octave**
- **Type de Langages de programmation**
- **L'environnement de travail**
- **Modes de fonctionnement dans GNU Octave**
- **Quelques commandes utiles**
- **Opérateurs arithmétiques, logiques et relationnels**

Présentation de Matlab

- **Le nom de MATLAB vient de MATrix LABoratory car les éléments de données de base manipulés par MATLAB sont des matrices.**
- **Il a été développé par le Professeur de mathématiques Cleve Moler en 1977 et écrit à l'origine par le langage Fortran.**
- **C'est un environnement de programmation puissant, complet et facile à utiliser, destiné pour le calcul numérique / scientifique, la visualisation des données et la programmation.**
- **Il contient un mécanisme de toolboxes (boîtes à outils qui sont des collections de M-files) qui permet d'étendre les fonctionnalités de Matlab.**
- **Matlab est un logiciel avec une licence payante.**

Présentation de GNU Octave

- **GNU Octave est un alternative libre de Matlab, développé par John W. Eaton en 1992.**
- **Le nom Octave vient d'Octave Levenspiel (en), ancien professeur de génie chimique de John W. Eaton qui était connu pour son aptitude à donner de bonnes approximations à des problèmes numériques.**
- **Ce langage est considéré comme le meilleur clone de Matlab en terme de compatibilité.**
- **GNU Octave offre un mécanisme d'extension basé sur des packages téléchargeables qui sont distribués sur un dépôt appelé octave forge.**

Outils semblables à Matlab et à Octave

- **Python avec les librairies scientifiques NumPy, SciPy i MatPlotLib, Mayavi**

  Spyder, etc.

- **Julia**

- **Scilab jeune octave mais non compatible avec MATLAB/Octave**

Caractéristiques de Matlab et Octave

- **MATLAB et Octave sont "case-sensitive", c'est-à-dire qu'ils distinguent les majuscules des minuscules (dans les noms de variables, fonctions...);**
  - **Ex : les variables abc et Abc sont 2 variables différentes ; la fonction sin (sinus) existe, tandis que la fonction SIN n'est pas définie...**
- **Le typage est entièrement dynamique, c'est-à-dire que l'on n'a pas à se soucier de déclarer le type et les dimensions des variables avant de les utiliser;**
- **La numérotation des indices des éléments de tableaux débute à 1 (comme en Fortran) et non pas 0 (comme dans la plupart des langages actuels : Python, C/C++, Java...);**
- **Ces deux langages sont interprétés.**

La différence entre Matlab et Octave

Matlab	Octave
Beaucoup toolboxes	Moins de package
Éditer des graphiques en double cliques	Programmer pour la modification des graphiques
Licence payante	Licence libre

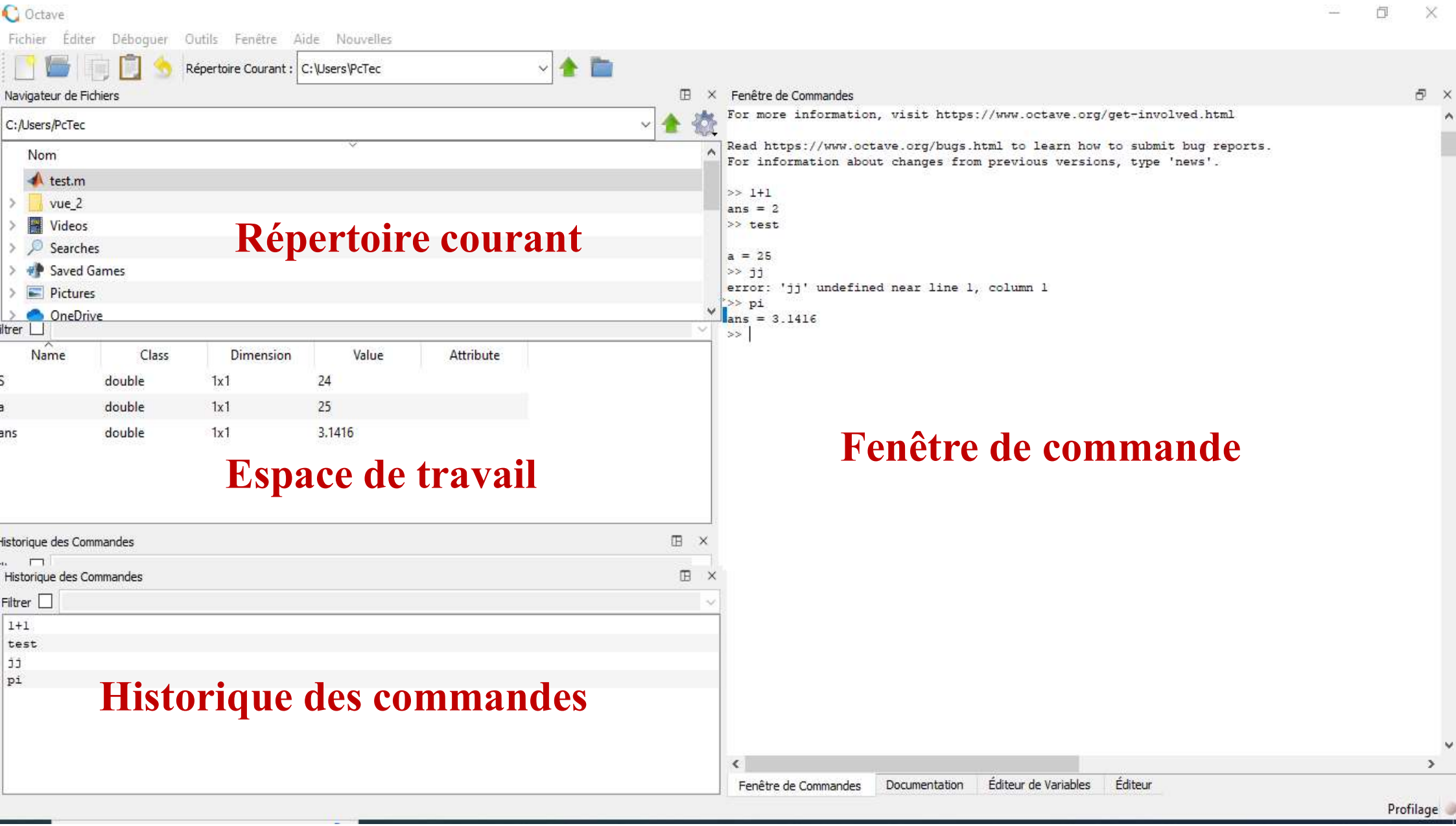
Type de Langages de programmation

Il existe deux grands types de langages:

Langage interprété: dans ce type de langage, le code source est interprété, par un logiciel qu'on appelle interpréteur. Celui-ci va utiliser le code source et les données d'entrée pour calculer le résultat. L'interprétation du code source est un processus «pas à pas»: l'interpréteur va exécuter les lignes du code une par une, en décidant à chaque étape ce qu'il va faire ensuite.

Langage compilé: le code source est tout d'abord compilé, par un logiciel qu'on appelle compilateur, en un code binaire qu'un humain ne peut pas lire mais qui est très facile à lire pour un ordinateur appelé l'exécutable. Ensuite le système d'exploitation va utiliser l'exécutable et les données d'entrée pour calculer le résultat.

L'environnement de travail: Octave


Octave
Fichier Éditer Déboguer Outils Fenêtre Aide Nouvelles
Répertoire Courant : C:\Users\PcTec
Navigateur de Fichiers
C:/Users/PcTec
Nom
test.m
vue_2
Videos
Searches
Saved Games
Pictures
OneDrive
Répertoire courant
Name Class Dimension Value Attribute
double 1x1 24
double 1x1 25
double 1x1 3.1416
Espace de travail
Historique des Commandes
Historique des Commandes
Filtrer
1+1
test
jj
pi
Historique des commandes
Fenêtre de Commandes
For more information, visit https://www.octave.org/get-involved.html
Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
>> 1+1
ans = 2
>> test
a = 25
>> jj
error: 'jj' undefined near line 1, column 1
>> pi
ans = 3.1416
>>
Fenêtre de commande
Fenêtre de Commandes
Documentation
Éditeur de Variables
Éditeur
Profilage

L'environnement de travail

Fenêtre de commandes

Fenêtre de commandes est l'une des plus importantes fenêtres d'octave, elle permet de traiter des instructions.

- **C'est après l'invite ou (Les caractères >>), qu'on tape les commandes qui seront exécutées par le logiciel après avoir appuyer sur la touche entrée .**
- **Le résultat s'affiche dans la fenêtre, mais il peut être représenté graphiquement dans une nouvelle fenêtre spécifique (avec possibilité de zoom, d'impression, etc).**

L'environnement de travail

Espace de travail et Le répertoire courant

- **Espace de travail:** **Permet de visualiser les variables (nom, dimension et type).**
- **Le répertoire courant:** **C'est le répertoire où sont enregistrés les fichiers -M. Il est fortement conseillé de se créer un répertoire autre que celui fournit par Octave afin de mieux gérer les fichiers. Pour utiliser un fichier-M, il doit être enregistré dans le répertoire courant**

L'environnement de travail

Historique de commandes et Barres de menus

- **Historique de commandes :Permet d'inscrire les commandes au fur et à mesure qu'elles sont appelées dans la fenêtre de commande.**
- **Barres de menus : Permet d'accéder aux commandes d'octave . Voici quelques exemples:**
  - **Editer → effacer la fenêtre de commande c'est-à-dire effacer les instructions et/ou les résultats visibles dans la fenêtre de commande.**
  - **Editer→ effacer l'historique des commandes: Effacer les commandes précédentes mises en mémoire.**
  - **Editer → effacer l'espace de travail : Effacer de la mémoire les variables stockées.**

Modes de fonctionnement dans Octave

Octave propose deux modes de fonctionnement :

- **Mode interactif : où octave interprète et exécute les commandes (instructions) directement après la saisie dans la fenêtre de commandes.**
- **Mode exécutif : il s'agit de l'exécution d'un programme (script) en langage octave (ligne par ligne) saisie dans un fichier avec l'extension "*.m« ce qu'on va voir par la suite).**

Premières commandes

- **Le moyen le plus simple d'utiliser octave est en mode interactif c'est-à-dire d'écrire directement dans la fenêtre de commande juste après le curseur >>**
- **Pour calculer une expression mathématique il suffit de l' écrire comme ceci :**

>> 7+8 **Puis on clique sur la touche Entrer pour voir le résultat**

ans =15

- **Si nous voulons qu'une expression soit calculée mais sans afficher le résultat, on ajoute un point virgule**
- **';' à la fin de l'expression comme suit :**

```
>>  7+8  ;
>>
```

Plusieurs expressions dans la même ligne

Il est possible d'écrire plusieurs expressions dans la même ligne en les faisant séparées par des virgules ou des points virgules. Par exemple :

```
>> x=5+6, y=2*5-1, 12-4
x =
11
y =
9
ans =
8
>> 5+6; 2*5-1, 12-4;
ans =
9
```

La sauvegarde

- **Pour sauvegarder l'ensemble des variables déjà manipulées en Octave, on utilise la commande « save »**

>> save tp1.mat permet de créer **un nouveau fichier portant le nom tp1.mat dans le répertoire courant.**

- **Pour sauvegarder un script**

>> save tp1.m

Quitter Octave et Restaurer un espace de travail

Quitter octave : Pour quitter Octave on peut utiliser soit le menu Fichier → Quitter ou d'utiliser directement la commande suivante

>> quit

Restaurer un espace de travail : (exemple : tp1.mat) , dans ce cas on tape la commande « load »

>> load tp1

L'aide d'octave

Octave met à la disposition des utilisateurs, des packages. On peut accéder à la documentation concernant ces packages de deux façons:

La commande help : permet d'afficher dans l'interpréteur toute la documentation d'une fonction.

Utilisation de l'onglet documentation : il suffit de chercher dans l'espace de recherche.

Les opérateurs arithmétiques d'octave

Les opérations de base dans une expression sont résumées dans le tableau suivant :

Symbole	***Opération***
+	***addition***
-	***soustraction***
*	***multiplication***
/	***division***
\	***division gauche (ou la division inverse)***
^	***puissance***

Priorités des opérateurs arithmétiques

Priorité 1 : ^ (La puissance)

Priorité 2 : * et / (La multiplication et la division, même priorité)

Priorité 3 : + et - (L'addition et la soustraction, même priorité)

Il est possible de modifier cet ordre de priorité en intégrant les parenthèses ().

Quand deux opérateurs ont la même priorité (de la gauche vers la droite)

Les opérateurs Logiques

Symbole	*Opération*
&	*ET*
|	*OU*
~	*NON*

Les opérateurs relationnels

Symbole	*Opération*
<	*inférieur strictement*
>	*supérieur strictement*
<=	*inférieur ou égal*
>=	*supérieur ou égal*
==	*égal*
~=	*est différent*

UNIVERSITÉ ABOU BAKR BEL-KAID–TLEMCEN
FACULÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
Première année licence

Outils de Programmation pour les mathématiques

Cours 2
Variables et fonctions prédéfinies

Mme HABRI née BENMAHDI Meryem Bochra

Année universitaire: 2022-2023

Plan du cours

I. La création de variables
II. Règles pour le nommage d'une variable
III. Types de variables
IV. Liste des variables utilisées dans le programme
V. Lecture d'une variable
VI. Affichage
VII. Suppression de la variable
VIII. Sauvegarde d'une variable
IX. Variables spéciales
X. Fonctions prédéfinis
XI. Précision de calcul

La création de variables

Pour créer une variable il faut donner un nom à la variable et sa définition c'est-à-dire une valeur sans se préoccuper du type de la variable

'*variable = définition*'

Par exemple :

```
>> a = 10 ;
```

Règles pour le nommage d'une variable

- **Le nom d'une variable**
  - **ne doit contenir que des caractères alphanumériques ou le symbole '_' (underscore)**
  - **doit commencer par une lettre de l'alphabet.**
- **Il faut faire attention aux majuscules car Octave est sensible à la casse (A et a sont deux identifiants différents).**
- **Lorsque le calcul d'une opération n'est pas affecté à une variable, Octave crée automatiquement une variable ans (answer) qui contient le résultat de l'opération.**

Exemple >> 2+13

ans =15

Types de variables

- **En Octave, l'utilisation de variables ne nécessite pas de déclaration de type.**
- **Les types de variables sont déterminés d'une façon dynamique (c.à.d. durant le temps d'exécution).**
- **Les quatres principaux types de variables utilisés par Octave sont :**
  - **Le type réel**
  - **Le type complexe**
  - **Le type logique (Booléen) : Peut avoir deux valeurs (true=1 ou false=0)**
  - **Le type chaîne de caractères : Délimité par le symbole ‘**

Types des variables

>> a=23; b='MI'; c=a+4i; d=false;

4 variables crées :

- **a de type réel et vaut 23**
- **b de type chaîne de caractères et vaut 'MI'**
- **c de type complexe et vaut 23+4i**
- **d de type booléen (logique) et vaut 0=false**

Liste des variables utilisées dans le programme

Pour voir la liste des variables utilisées, soit on regarde directement l'espace de travail, soit on utilise les commandes 'whos' ou 'who'.

whos donne une description détaillée (le nom de la variable, son type et sa taille), par contre who donne juste les noms des variables.

Par exemple, dans ce cours on a utilisé quatre variables a, b, c et d :

```
>> who
Variables visible from the current scope:
a  b  c  d
```

Liste des variables utilisées dans le programme

```
>> whos
Variables visible from the current scope:

variables in scope: top scope

  Attr   Name        Size                Bytes  Class
  ====   ====        ====                =====  =====
         a           1x1                     8  double
         b           1x2                     2  char
         c           1x1                    16  double
         d           1x1                     1  logical

Total is 5 elements using 27 bytes
```

Lecture d'une variable

La fonction input : permet la saisie d'une valeur depuis le clavier:

- **Pour les valeurs numériques, n = input('message ') affiche message et affecte à la variable n la valeur numérique entrée au clavier.**
- **Pour les chaînes de caractères, str = input('message','s') affiche message et affecte à la variable str la valeur entrée au clavier qui est considérée alors comme une chaîne de caractères.**

Lecture

Exemple:

```
>> A=input('Entrer la valeur de A : ')

Entrer la valeur de A : 3

A =

3

>> S=input('Entrer une phrase : ','s')

Entrer une phrase : bonjour tout le monde

S =

bonjour tout le monde
```

Affichage

La fonction disp : permet d'afficher

Exemple

```
>> disp('bonjour')
bonjour
>> disp(‘Bonjour');
Bonjour
>> a=[1 2 3]; b=['O' 'P' 'M'];
>> disp(a)
1 2 3
>> disp(b);
OPM
```

Affichage avec format

printf(‘ ’) format de sortie sur écran

Exemples :

>> L=10;

>> printf(‘La longueur L=%f\n’,L)

La longueur L=10

Dans cette chaîne, on a :

% pour la sortie de la variable,

f format de sortie

\n retour à la ligne

L variable.

Suppression de variables

La commande clear permet de supprimer une variable, plusieurs variables, ou toutes les variables de l'espace de travail.

>> clear a , supprimer la variable a.

>> clear a b , supprimer les variables a et b.

>> clear all, supprimer toutes les variables.

Variables spéciales (prédéfinies)

Octave possède des variables prédéfinies, ces variables existent mais elles ne sont pas présentes dans le l'espace de travail

La variable pi :

```
>> pi
ans =
3.1416
```

Les variables i et j : les unités imaginaires des nombres complexes:

```
>> i
ans =
0 + 1.0000i
>> j
ans =
0 + 1.0000i
```

Variables spéciales (prédéfinies)

La variable Inf (L'infini) : Elle est obtenu par exemple en effectuant l'opération 1/0.

```
>> 1/0
ans = Inf
```

La variable NaN (Not -a -Number) : La valeur d'un NaN est obtenue des opérations n'ayant pas de valeur numérique en retour. Par exemple, multiplication de 0*Inf, l'addition Inf-Inf.

```
>> 0*Inf
ans = NaN
```

Changer la valeur d'une variable prédéfinie

Dans Octave , les variables prédéfinies sont réservées, mais, on peut également leur affecter une autre valeur (entières ou réelles).

Par exemple, la commande suivante crée une variable *pi* qui vaut *25* en écrasant la valeur *3.1416*

```
>> pi=25

pi =

25
```

Sauvegarde d'une variable

- **Dans Octave, toutes les variables de l'espace de travail sont effacées à la fermeture du logiciel, ainsi toutes les variables sont perdues.**

- **Afin de sauvegarder les variables dans un fichier avec l'extension « .mat », il est nécessaire d'utiliser la commande « save ».**

Sauvegarde d'une variable

Exemple

>> X=12 ; S='Programme Matlab' ; Y=true; Z = 25.3; %Création de variables

>> save('mes_variables') %crée un fichier mes_variables.mat dans le dossier courant contenant toutes les variables qui se trouve dans l'espace de travail

>> save('variable_X','X') %crée un fichier variable_X.mat contenant que la variable X

Les fonctions prédéfinies

- **Octave propose plusieurs fonctions prédéfinies pour le calcul arithmétique ou pour d'autres opération.**
- **Une fonction est appelée par son nom suivi de ces variables (paramètres) entre parenthèses () si elle a des paramètres. Sinon, elle est appelée par son nom.**

Fonctions sans paramètres

> > clock

ans =

2.0230e+03 2.0000e+00 1.3000e+01 1.1000e+01 1.2000e+01 6.9351e+00

> > date

ans = 13-Feb-2023

Fonction avec paramètres

- **Fonctions de calcul**
- **Fonctions trigonométrique**
- **Fonctions d'arrondi et de reste**
- **Fonction nombres premiers**

Fonctions de calcul

Fonction	Signification
exp	fonction exponentielle
log	logarithme naturel (à base) ln
log10	logarithme décimal (à base 10)
sqrt	La racine carrée
abs	la valeur absolue

Fonctions de calcul

```
>> sqrt(4)

ans = 2

>> abs(-3)

ans = 3

>> exp(1)

ans = 2.7183
```

```
>> log(exp(4))

ans = 4

>> log10(10^3)

ans = 3

>> z=1+i ;

>> abs(z)

ans = 1.4142
```

Fonctions de calcul

Fonction	Signification
conj	le conjugué d'un nombre complexe
imag	la partie imaginaire d'un nombre complexe
real	la partie réelle d'un nombre complexe
complex	cette fonction calcule un nombre complexe à partir de ces parties réelle et imaginaire
angle	l'argument d'un nombre complexe (en radian)

Fonctions de calcul

```
>> com=complex(2,4)

com =  2 + 4i

>> z=1+i

z =  1 + 1i

>> abs(z)

ans = 1.4142
```

```
>> real(z)

ans = 1

>> imag(z)

ans = 1

>> conj(z)

ans = 1.0000 - 1.0000i
```

Fonction du factorielle

Pour calculer la factorielle d'un entier n, il existe deux fonctions factorial(n) ou gamma(n+1)

> > gamma(7)

ans = 720

> > gamma(6)

ans = 120

> > factorial(6)

ans = 720

Fonctions trigonométriques

Fonction	Signification
sin	Sinus d'un angle (en radian)
sind	Sinus d'un angle (en degré)
cos	Cosinus d'un angle (en radian)
cosd	Cosinus d'un angle (en degré)
tan	TC d'un angle (en radian)

Fonctions trigonométriques

```
>> sin(pi/2)

ans = 1

>> cos(-pi)

ans = -1

>> sind(-90)

ans = -1
```

Fonctions d'arrondi et de reste

Fonction	signification
round	arrondir à l'entier le plus proche
floor	arrondir à l'entier le plus proche vers -œ
ciel	arrondir à l'entier le plus proche vers +œ
fix	arrondir vers zéro
mod	reste de division
lcm	plus petit commun multiple de m et n
gcd	plus grand commun diviseur de m et n
factor	décomposition en facteurs premiers de n

Fonctions d'arrondi et de reste

>> round(5.3)

ans = 5

>> round(-15.6)

ans = -16

>> floor(6.87)

ans = 6

>> floor(-7.08)

ans = -8

>> ceil(6.17)

ans = 7

>> ceil(-7.08)

ans = -7

Fonctions d'arrondi et de reste

>> fix(3.4)

ans = 3

>> fix(-3.4)

ans = -3

>> mod(13,7)

ans = 6

>> mod(14,3)

ans = 2

> > gcd(35,28)

ans = 7

> > lcm(122,30)

ans = 1830

> > factor(12)

ans = 2 2 3

Fonctions prédéfinies (nombres premiers)

Fonction	signification
isprime(n)	Renvoie vrai si le nombre est premier et faux s'il n'est pas
primes(n)	Revoie la liste des nombres premiers plus petits ou égale au nombre en paramètre
list_primes(n)	Revoies la liste des n premiers nombres premiers
factor(n)	Revoie la décomposition en facteurs premiers de n

Fonctions prédéfinies (nombres premiers)

```
>> a=isprime(3)
a = 1
>> b=isprime(4)
b = 0
>> list_primes(5)
ans =
   2   3   5   7   11
```

```
>> primes(5)
ans =
 2   3   5
>> factor(12)
ans =
  2   2   3
>> factor(18)
ans =
  2   3   3
```

Précision de calcul

- **La précision de calcul des nombres réels dans Octave peut aller jusqu'aux 15 chiffres significatifs après la virgule.**
- **Le résultat d'une opération de calcul par défaut est affiché avec quatre chiffres après la virgule.**
- **Il est possible de configurer le format d'affichage des nombres dans la fenêtre de commande à l'aide de la commande « format » :**

La commande	signification
format short	Affiche les nombres avec 04 chiffres après la virgule
format long	Affiche les nombres avec 15 chiffres après la virgule
format bank	Affiche les nombres avec 02 chiffres après la virgule
format rat	Affiche les nombres sous forme d'une ration (a/b)

Précision de calcul

Exemple

```
> > pi
ans = 3.1416
> > format long
> > pi
ans = 3.141592653589793
> > format rat
> > pi
ans = 355/113
> > format short
> > pi
ans = 3.1416
>> format bank
>> pi
ans = 3.14
```

Exercice

Ecrire un script qui permet de calculer la somme et le produit de deux nombres entrés par l'utilisateur.

Solution:

```
disp("Exercice qui calcule la somme et le produit de deux nombres")
A=input('entrer la valeur de A =' )
B=input('entrer la valeur de B = ')
S=A+B;
p=A*B;
printf("la somme de %d et de % d est de %d \n",A,B,S);
printf("le produit de %d et de % d est de %d \n",A,B,p);
```

UNIVERSITÉ ABOU BAKR BEL-KAID–TLEMCEN
FACULÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
Première année licence

Outils de Programmation pour les mathématiques

Cours 3
Programmation sous Octave (syntaxe)

Mme HABRI née BENMAHDI Meryem Bochra

Année universitaire: 2022-2023

Plan du cours

I. Les commentaires

II. Type de fichiers dans octave

III. scripts

IV. Fonctions

V. Structures de contrôle

1. Structures itérative
2. Structures conditionnelles
3. Instruction Switch

Commentaires

Un commentaire n'est pas évalué par le langage de programmation, il permet seulement de documenter un script ou une fonction. Un commentaire peut être sur une ligne ou peut s'étendre sur plusieurs lignes.

- **Commentaire sur une seul ligne, il commence par % ou par #**

  Exemple n=100 % nombre de nœuds dans un réseau

- **Commentaire sous forme de paragraphe, les séquences %{ et %} délimitent un commentaire s'étendant sur plusieurs ligne. Il ne faut rien avoir dans les 2 lignes contenant ces séquences %{ et %} (ni avant ni après).**

Remarque Lorsque le caractère pourcentage (%) se trouve dans la commande printf, c'est pour une définition de format

Exemple: printf('le résultats = %f', 25)

Type de fichiers sous octave

Dans octave, il y a quatre types de fichiers :

- **Les scripts avec l'extension .m,**
- **Les fonctions avec l'extension .m,**
- **Les figures avec l'extension .fig,**
- **Les espaces de travail enregistrés avec l'extension .mat**

La programmation dans octave se fait avec les scripts et les fonctions

Scripts

- **Script, programme, ou fichier de commandes est une suite d'instructions et de commandes octave.**
- **Pour exécuter un script, il faut évoquer son nom de fichier dans la fenêtre de commande d'octave ou l'exécuter avec l'icone d'exécution qui se trouve dans l'interface de l'éditeur d'octave.**
- **Les instructions du scripts s'exécutent l'une après l'autre comme si elles étaient saisies sur la ligne de commande ou la fenêtre de commandes octave.**
- **Les variables définies dans le scripts restent dans la mémoire d'octave (espace de travail) après l'exécution du script. C'est possible de les enregistrer avec la commande save nomfichier**

Fonctions

- **Une fonction octave permet d'exécuter un ensemble d'instructions selon des arguments d'entrée (paramètres) et de retourner un ou plusieurs résultats.**
- **L'implication de fonctions permet la réutilisation d'une partie ou des parties de programme pour ainsi éviter la répétition.**
- **Deux points à retenir, lors de la définition d'une fonction:**
  - **Il faut commencer par l'instruction « function » et terminer par « endfunction»**
  - **Il faut que le fichier porte le nom de cette fonction.**

Exemple

- **Exemple d'une définition de fonction sous octave fonction de la somme**

```
function s= somme(a,b)
 s=a+b;
endfunction
```

→ **Il faut l'enregistrer sous le nom somme**

- **L'appel de la fonction se fait en utilisant le script ou la ligne de commande**

```
e=12;
f=10;
a=somme(e,f)
```

→ **Dans la fenêtre de commandes, on aura: a=22**

- **Lors de l'appel de la fontion, il faut que la foction somme soit visible dans le répertoire courant.**

Structures itératives

Les boucles permettent d'exécuter une séquence d'instructions de manière répétée. Les structures itératives utilisées dans octave sont:

- **La boucle for**
- **La boucle while**
- **La boucle do until**

Structures itératives
Boucle for

Syntaxe

for indice = inf: sup

Instructions

endfor

- **indice : est une variable appelée l'indice de la boucle.**
- **inf (borne inférieure) et sup (borne supérieure) sont deux constantes réelles.**
- **Instructions est la suite d'instructions à répéter (On parle du corps de la boucle).**
- **Si *inf* ≤ *sup* , Instructions sont exécutées (sup-inf+1) fois, pour les valeurs de la variable.**
- **indice égales à inf, inf+1,...,sup , puis on passe à l'instruction qui suit immédiatement l'instruction de fin de boucle (*endfor*).**

Structures itératives
Boucle for

- **Si *inf* > *sup* , on passe directement à l'instruction qui suit immédiatement l'instruction de fin de boucle (*endfor*).**
- **L'indice de la boucle ne prend pas nécessairement des valeurs entières.**
- **On peut naturellement imbriquer des boucles for les unes dans les autres.**
- **Il est possible d'utiliser un incrément (pas) autre que 1 (valeur par défaut). La syntaxe est alors :**

***for* indice = inf:pas:sup**

Instructions

endfor

- **Le pas peut être négatif.**

Structures itératives
Boucle for

```
for n = 0:9
n.^2
endfor
```

Cette boucle va afficher le carré du nombre 0 jusqu'au nombre 9.

```
for n = 0:2:9
n.^2
endfor
```

Cette boucle va afficher le carré du nombre 0, 2, 4, 6, 8.

Structures itératives
While

while condition

Instructions

endwhile

- **Les instructions sont exécutées tant que la condition est vraie .**
- **condition : est une expression logique (ayant deux valeurs vrai ou faux).**
- **Instructions : est une suite d'instructions qui se répète tant qu'une condition a la valeur vrai.**
- **Lorsque la valeur de condition devient faux, on passe à l'instruction qui suit immédiatement l'instruction de fin de boucle (endwhile).**

Structures itératives

While

```
n=0;
while (n < 10)
n.^2
n=n+1;
endwhile
```

Cette boucle va afficher le carré du nombre 0 jusqu'au nombre 9. cette boucle se compose donc d'un compteur (ici n) initialisé à une valeur quelconque (ici 0), d'une condition de continuation (ici x < 10), d'une ou plusieurs commandes (ici d'afficher le carré de n) et d'une incrémentation (ici n = n + 1).

Structures itératives
do until

.

Syntaxe

do

instructions

until (condition)

Instructions : est une suite d'instructions qui se répète tant que condition est vraie.

condition : est une expression logique (ayant deux valeurs vrai ou faux).

Les instructions sont exécutées au moins une fois même si la condition est fausse.

Lorsque la valeur de condition devient faux, on passe à l'instruction qui suit immédiatement l'instruction until(condition).

Structures itératives

do until

```
i = 0;
do
n=i.^2
i++;
until (i == 10)
```

Cette boucle va afficher le carré du nombre 0 jusqu'au nombre 9. cette boucle se compose donc d'un compteur (ici i) initialisé à une valeur quelconque (ici i=0), d'une condition de continuation (ici x == 10), d'une ou plusieurs commandes (ici d'afficher le carré de i) et d'une incrémentation (ici i=i+1).

Exercice

Ecrire un script permettant de calculer les n premiers termes d'une suite arithmétique en utilisant la formule de récurrence: u(n+1)=u(n)+1

Et de calculer leur somme, sachant que le premier terme, la raison et n sont entrés par l'utilisateur.

Remarque: n doit être un nombre positif.

Solution

```
do
  n=input('entrer un nombre positive = ');
until(n>0)
r=input('entrer la raison de la suite = ');
u1=input('entrer la valeur du premier terme = ');
s=0;
for i=1:1:n
  u1=u1+r;
  printf("la valeur du terme %d est de % f \n",i,u1);
  s=s+u1;
endfor
printf("la somme des %d premiers termes est de % f\n",n,s);
```

Structure conditionnelle

Symbole	*Opération*	*clavier*
&	*ET*	1
|	*OU*	Alt + 6 (ou se trouve -)
~	*NON*	Alt + 2(ou se trouve é)

Structure conditionnelle

Une structure conditionnelle permet d'exécuter une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée. Différentes formes de structures conditionnelles existent sous octave entre autres:

- **Forme simple**
- **Forme alternative**
- **Forme imbriquée**

Structure conditionnelle
Forme simple

Syntaxe

if **condition**

Instructions

endif

- **Condition : est une expression logique dont le résultat peut être vrai ou faux.**
- **Instructions : est une suite d'instructions.**
  - **Si le résultat de l'évaluation de la condition est vraie, on exécute les instructions, puis l'instruction qui suit le mot clé *endif*.**
  - **Si le résultat de l'évaluation de condition est faux on passe directement à l'instruction qui suit le mot clé *endif*.**

Structure conditionnelle
Forme simple exemple

Ecrire un script qui permet à l'utilisateur de vérifier si la somme de 5 et 4 est égale à 9

Structure conditionnelle
Forme simple exemple

```
somme = input('La somme de 5 et 4 est : ');
if (somme==9)
disp('Réponse correcte')
endif
```

Enregistrer sous **formesimpleif**.m

Après exécution

```
>> formesimpleif
      La somme de 5 et 4 est : 2
>> formesimpleif
      La somme de 5 et 4 est : 9
      Réponse correcte
```

Structure conditionnelle
Forme alternative

Syntaxe

If condition

Instructions_1

else

Instructions_2

endif

- **Condition** est une expression logique dont le résultat peut être vrai ou faux.
- **Instructions_1** et **Instructions_2** sont deux suites d'instructions.
  - Si le résultat de l'évaluation de condition est **vraie**, on exécute **Instructions_1**, puis l'instruction qui suit le mot clé ***endif***.
  - Si le résultat de l'évaluation de condition est **faux** , on exécute **Instructions_2**, puis l'instruction qui suit le mot clé ***endif***.

Structure conditionnelle
Forme alternative exemple

Ecrire un script qui permet de calculer 1/x sachant que x est entrée par l'utilisateur ; faites attention à la division par 0

Structure conditionnelle
Forme alternative exemple

```
x = input('entrez la valeur de x : ');
if (x ~= 0)
y = 1/x
else
error('Division par zero’);
endif
```

Le script est enrengistrer sous EIF2

Après exécution

```
>> EIF2

entrez une valeur a x : 4
y = 0.2500
>> EIF2
entrez une valeur a x : 0
error: Division par zero
error: called from
    EIF2 at line 5 column 1
```

Structure conditionnelle
Forme imbriquée

Syntaxe

If condition_1

Instructions_1

elseif condition_2

Instructions_2

elseif condition_3

Instructions_3

...

else

Instructions_n

Endif

- condition_i : **est une expression logique dont le résultat peut être vrai ou faux..**

Structure conditionnelle
Forme imbriquée

Instructions _i : est une suite d'instructions

Si le résultat de l'évaluation de la condition_i est vraie, on exécute Instructions _i , puis l'instruction qui suit le mot clé *endif*.

Si aucune des expressions condition_1, condition_2,...,n'est vraie, on exécute l'instruction Instructions_n (suite d'instructions par défaut), puis l'instruction qui suit le mot clé *endif*.

- **Il n'est pas nécessaire de prévoir un cas par défaut (bien que cela soit préférable). S'il n'y a pas de cas par défaut, et si aucune des expressions condition_1, condition_2**
- **,..., n'est vraie , alors on continue à la première instruction suivant le mot clé *endif*.**

Structure conditionnelle
Forme imbriquée exemple

Ecrire un script permettant d'entrer un âge et selon l'âge afficher un message comme suit:

- **Age < 2 Vous êtes un bébé**
- **Age < 13 Vous êtes un enfant**
- **Age < 18 Vous êtes un adolescent**
- **Age < 60 Vous êtes un adulte**
- **Et au delà de 60 Vous êtes un vieillard**

Structure conditionnelle
Forme imbriquée exemple

```
age=input('Entrez votre âge : ');
if (age < 2)
disp('Vous êtes un bébé')
elseif (age < 13)
disp('Vous êtes un enfant')
elseif (age < 18)
disp ('Vous êtes un adolescent')
elseif (age < 60)
disp ('Vous êtes un adulte' )
else
disp ('Vous êtes un vieillard')
endif
```

Après exécution →

```
>> exifimbr

Entrez votre âge : 18
Vous êtes un adulte
>> exifimbr
Entrez votre âge : 8
Vous êtes un enfant
```

Exercice d'application

Écrire un script qui demande à l'utilisateur d'entrer une note sur 20 et affiche l'appréciation correspondante :

- **note < 10 → "Insuffisant"**
- **10 ≤ note < 12 → "Passable"**
- **12 ≤ note < 14 → "Assez bien"**
- **14 ≤ note < 16 → "Bien"**
- **16 ≤ note ≤ 20 → "Très bien«**
- **En prenant en considération la note invalide**

Solution

```
note = input("Entrez votre note sur 20 : ");
if ((note < 0)|(note > 20))
disp(" note invalide ");
elseif note < 10
 disp("Insuffisant");
elseif note < 12
disp("Passable");
elseif note < 14
disp("Assez bien");
elseif note < 16
 disp("Bien");
else disp("Très bien");
endif
```

Instruction switch

Syntaxe

```
switch var
  case const_ 1
instructions_1
  case const_2
instructions_2
  ...
  otherwise
Instructions_n
endswitch
```

var est une variable numérique ou une variable chaîne de caractères.

const_i est une constante numérique ou des constantes chaînes de caractères de même type que var.

Instruction switch

Instructions_i est une suite d'instructions.

- **Si la variable var est égale à la constante const_i , on exécute la suite d'instructions correspondante (c'est -à -dire Instructions_i , puis l'instruction qui suit le mot clé *endswitch*.**
- **Si var n'est égale à aucune des constantes const_1, const_2,..., on exécute l'instruction Instructions_n (suite d'instructions par défaut) , puis l'instruction qui suit le mot clé *endswitch*.**
- **Il n'est pas nécessaire de prévoir un cas par défaut (bien que cela soit préférable).**
- **S'il n'y a pas de cas par défaut, et si var n'est égale à aucune des constantes const_i, alors on continue à la première instruction suivant le mot clé *endswitch*.**

Instruction switch

```
x = input ('Entrez un nombre : ') ;
switch (x)
case 0
disp(x)
case 10
disp('x = 10')
case 100
disp(x)
otherwise
disp('x n''est pas 0 ou 10 ou 100')
endswitch
```

```
>> sriptswitch

Entrez un nombre : 10
x = 10
>> sriptswitch
Entrez un nombre : 100
100
>> sriptswitch
Entrez un nombre : 2
x n'est pas 0 ou 10 ou 100
```

UNIVERSITÉ ABOU BAKR BEL-KAID–TLEMCEN
FACULÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
Première année licence

Outils de Programmation pour les mathématiques

cours 4
Tableaux(vecteurs et matrices) sous octave

Mme HABRI née BENMAHDI Meryem Bochra

Année universitaire: 2022-2023

Plan du cours

- **Introduction sur les tableaux**
- **Déclaration d'un tableau**
- **Utilisation des éléments du tableau**
- **Les matrices**
- **Opérations sur les tableaux**
- **Constructeur de tableaux**
- **Extraction de données à partir de tableaux**
- **Concaténation des matrices**
- **Opération sur les tableaux et les matrices**
- **Fonctions prédéfinies**

Tableaux

- **Un tableau est une collection d'objets de données du même type, généralement stockés séquentiellement en mémoire.**
- **Un tableau est une structure de données élémentaire.**
- **Presque tous les langages de programmation prennent en charge les tableaux.**
- **Octave est un langage spécialisé dans le support des tableaux.**
- **Un tableau est la manière la plus évidente de représenter un vecteur ou matrice.**

Déclaration d'un tableau

- **Dans octave, nous pouvons construire un tableau et l'associer un identifiant très facilement. Par exemple :**

  »a=[4, 1.5, -5]

  a = 4.0000 1.5000 -5.0000

- **Les virgules sont facultatives et peuvent être omises :**

  »a=[4 1.5 -5]

  a = 4.0000 1.5000 -5.0000

- **Les crochets indiquent à octave que le contenu représente un tableau.**

Les opérations sur un tableau

- **Ce processus de construction d'un tableau implique un segment de mémoire alloué associé au nom de la variable.**
- **Dans la plupart des langages de programmation, vous devrez déclarer un tableau et attribuer les valeurs une par une. Dans octave, c'est automatique.**
- **Vous pouvez ensuite effectuer de l'arithmétique sur des tableaux aussi simplement que possible avec des scalaires. Par exemple :**

```
»b = a*2

b =

8 3 -10
```

Utilisation des éléments du tableau

- **Vous pouvez extraire des valeurs individuelles d'un tableau en spécifiant l'index dans le tableau à l'aide de parenthèses. Par exemple :**

```
»c= b(1)
 c = 8
```

- **Vous pouvez également attribuer de nouvelles valeurs à des éléments individuels d'un tableau. Par exemple :**

```
»b(3) = 6
 b =
 8 3 6
```

Utilisation des éléments du tableau

- **Dans octave l'indice du premier élément d'un tableau est toujours 1.**
- **Octave garde une trace de la taille des tableaux et s'assure que vous n'essayez pas d'aller au-delà de leurs limites. Par exemple :**

```
»b(4)
error: b(4): out of bound 3 (dimensions are 1x3)
```

Exercice 1 tableau

Écrire un script qui crée un tableau (vecteur) de 5 éléments et l'affiche.

Solution:

```
A = [1, 2, 3, 4, 5];  % Déclaration d'un vecteur ligne

disp("Le tableau A est :");

disp(A);
```

Exercice 2 tableau

Écrire une fonction qui permet à un utilisateur de lire les éléments d'un tableau (vecteur) et un script qui permet de tester cette fonction, d'afficher le vecteur et d'accéder à ses éléments.

Solution:

```
function A = lecteurV()
n = input(" entrer la taille du vecteur ");
A = zeros(1,n);
for i=1:n
printf (" entrer l élément  numéro %d du tableau ", i);
A(i)= input(" ");
endfor
endfunction
```

Script

```
A=lecteurV();
disp("Le tableau A est :");
disp(A);
printf("le premier element du vecteur est %d \n" , A(1));
```

Les matrices

- **Un tableau est une collection d'objets de données du même type.**
- **Les objets de données du tableau peuvent eux-mêmes être des tableaux.**
- **Une matrice est généralement représentée par un tableau de tableaux ou un tableau 2D. Octave prend en charge les matrices de la même manière qu'il prend en charge les vecteurs.**
- **Octave utilise l'operateur point-virgule (;) pour distinguer les différentes lignes d'une matrice. Par exemple :**

```
» a = [1 2 3 ; 4 5 6]
a =
1 2 3
4 5 6
```

Les matrices

- **Une autre manière pour saisir une matrice sous Octave est possible en saisissant des lignes sur différentes lignes.**
- **Une fois qu'un tableau débute par un crochet ([), Octave suppose qu'une nouvelle ligne signifie une nouvelle ligne de la matrice. Par exemple :**

```
»m = [1 2 3
4 5 6] ; % Une matrice composée de deux lignes.
m =

   1   2   3
   4   5   6
```

Les opérateurs matriciels et vectoriels

Les opérateurs mathématiques standard peuvent être appliqués aux vecteurs et aux matrices. Tous est géré automatiquement.

- **Par exemple, supposons que nous ayons deux matrices définie comme :**

```
»m1= [1 2 ; 3 4]
m1 =
   1   2
   3   4
»m2= [5 6 ; 7 8] ;
```

Les opérateurs matriciels et vectoriels

- **L'operateur de transposition change les lignes et les colonnes d'une matrice.**
  - **L'operateur de transposition est défini par le symbole d'apostrophe unique (‘).**

»m1’

ans =

1 3

2 4

- **La transposition a une priorité plus élevé que la multiplication.**

Addition et soustraction

- **L'addition et la soustraction des matrices sont identiques à l'algèbre linéaire. Par exemple :**

```
»ma=  m1+ m2
ma =
6 8
10 12
»ma2= m1 + 2
Ma2 =
3 4
5 6
```

- **Les opérations de l'addition et de la soustraction des matrices fonctionne, à condition que les deux matrices ont la même dimension.**

Multiplication matriciel

- **La multiplication matricielle est définie comme dans l'algèbre linéaire standard.**
- **Pour que la multiplication matricielle fonctionne, le nombre de colonnes de la première matrice doit être égale au nombre de lignes de la deuxième matrice.**

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} \times \begin{vmatrix} x & y & z \\ r & s & t \\ u & v & w \end{vmatrix}$$

$$= \begin{vmatrix} ax + br + cu & ay + bs + cv & az + bt + cw \\ dx + er + fu & dy + es + fv & dz + et + fw \\ gx + hr + iu & gy + hs + iv & gz + ht + iw \end{vmatrix}$$

Multiplication matriciel

La multiplication matricielle est définie comme dans l'algèbre linéaire standard. Par exemple :

```
»m1 = [1 2 ; 3 4] ; m2 = [5 6 ; 7 8] ;
» p= m1 * m2
p =
19 22
43 50
» d = m1 * 3
d =
3 6
9 12
```

Multiplication ponctuelle

- **Un certain nombre d'opérateurs spéciaux sont associés aux matrices et aux vecteurs, dont beaucoup sont uniques à octave.**
- **L'operateur .* effectue une multiplication point par point sur chaque paire correspondante d'éléments de deux matrices (parfois appelée multiplication des tableaux).Par exemple :**

```
»pp= m1.*m2
pp =
5 12
21 32
```

Division ponctuelle

L'opérateur ./ effectue une division point par point sur chaque paire correspondante d'éléments de deux matrices. Par exemple :

»dp= m1./ m2

dp =

0.2000 0.3333

0.4286 0.5000

Puissance ponctuelle

L'opérateur .^ effectue une exponentiation point par point sur chaque paire correspondante d'éléments de deux matrices. Par exemple :

»ep= m1.^m2

ep =

1 64

2187 65536

Division Matricielle

- **La division matricielle implique la résolution des inverses matriciels. Octave gère cela automatiquement !**
- **Notez que parce-que la multiplication matricielle n'est pas commutative, nous avons besoin du concept de division à gauche et à droite.**
- **La division a droite est post-multiplication par l'inverse d'une matrice :**

» dd= m1 / m2 % dd= $m1 * m2^{-1}$

dd =

3 -2

2 -1

Division Matricielle

La division à gauche est pré-multiplication par l'inverse d'une matrice :

» **dg = m1 \ m2 % c = $m1^{-1} * m2$**

c =

-3 -4

4 5

13

Constructeurs de tableaux
L'opérateur deux-points

- **Il est facile de construire de petits tableaux en spécifiant explicitement tous les éléments, mais ce n'est pas pratique pour les grands tableaux.**
- **Octave fournit l'opérateur deux-points (:) pour construire des séquences de valeurs.**
- **L'opérateur deux-points (:) produit un tableau équivalent aux éléments d'une séquence arithmétique.**
- **Les séquences arithmétiques sont définies en fonction de la première valeur de la série, de l'incrément entre les valeurs successives et de la dernière valeur de la série.**
- **La syntaxe pour créer un tableau à l'aide de l'opérateur deux-points (:) est la suivante :**

tableau = premier : incrément : dernier

L'opérateur deux-points (:)

Par exemple :

X = 3 : 2 : 11

X =

3 5 7 9 11

Si l'incrément est 1, il peut être omis. Par exemple :

x = 1 : 5

x =

1 2 3 4 5

L'opérateur deux-points est extrêmement utile, non seulement pour la création de tableaux, mais aussi pour le contrôle des boucles.

Sous-Tableaux

En plus de sélectionner des éléments individuels à partir de tableaux, octave permet la sélection de sections d'un tableau.

Par exemple :

```
»a = [10 :-1 : 1]
a =
10 9 8 7 6 5 4 3 2 1
» a(2 :5)
ans =
9 8 7 6
```

Sous-Tableaux

- **On peut également attribuer des valeurs aux sous-tableaux.**

» a(1 :3) = [-5 -2 -4]

a =

-5 -2 -4 7 6 5 4 3 2 1

- **La taille du tableau attribue doit correspondre à la taille du tableau sélectionné.**

Extraction de données à partir de tableaux 2D

Les opérations de sélection de matrices sont étendues aux tableaux 2D de manière naturelle.

»a = [1 2 3 ;4 5 6 ;7 8 9] ; % définition d'un tableau 2D

» a(3, 2) ; % récupérer la valeur de la ligne 3 colonne 2

ans =

8

Extraction de données à partir de tableaux 2D

» a(2, 1 :3) ; % récupérer les valeurs de la ligne 2 des colonnes 1 à 3.

ans =

4 5 6

» a(2 :3, 1 :2) ; % récupérer les valeurs des lignes 2-3 dans les colonnes 1-2.

ans =

4 5

7 8

Extraction de toutes les lignes ou colonnes

Si vous souhaitez extraire toutes les lignes (ou toutes les colonnes) d'une matrice, vous pouvez utiliser l'opérateur deux-points vides pour spécifier la ligne ou la colonne.

```
»a( :, 2) % récupérer les valeurs de la colonne 2

ans =

2

5

8
```

Extraction de toutes les lignes ou colonnes

»a(2 :3, :) % récupérer les valeurs des lignes 2-3 pour toutes les colonnes

ans =

4 5 6

7 8 9

Concaténation de matrices

Octave a une syntaxe très pratique pour concaténer des matrices, il suffit de coller les matrices côte à côte, ou les unes sur les autres, dans un ensemble de crochets englobants.

» a = [1 2 3] ;

» b = [a 7 8] ; % fusionne 7 et 8 à la fin de a.

b =

1 2 3 7 8

» a = [a a(1 :2) ; b]

a =

1 2 3 1 2

1 2 3 7 8

Opérations sur les Tableaux et les Matrices

Opération ponctuelle	La forme octave
Division à droite	a./b
Division à gauche	a.\b
Produit	a.* b
Puissance	a.^b

Opération ponctuelle signifie point par point sur chaque paire correspondante d'éléments de deux matrices.

Opérations sur les Tableaux et les Matrices

Opération	Forme octave	signification
Addition (Soustraction)	a + b (a - b)	L'addition et la soustraction suivent le même principe.
Multiplication des matrice	a*b	Le produit des deux matrice a et b
Division à droite	a/b	Le produit de a et $\mathbf{b^{-1}}$
Division à gauche	a\b	Le produit de $\mathbf{a^{-1}}$ et b

Fonctions propres aux vecteurs

Fonction	Utilisation
length(A)	retourne la taille d'un vecteur A
sum(x)	somme des éléments du vecteur x
prod(x)	produit des éléments du vecteur x

Fonctions propres aux vecteurs

Fonction	Utilisation
max(x)	plus grand élément du vecteur x
min(x)	plus petit élément du vecteur x
mean(x)	moyenne des éléments du vecteur x
sort(x)	ordonne les éléments du vecteur x par ordre croissant

Fonctions propres aux matrices

Fonction	Utilisation
eye(n)	la matrice identité (carrée de taille n)
ones(m,n)	la matrice a m lignes et n colonnes dont tous les éléments valent 1
zeros(m,n)	la matrice a m lignes et n colonnes dont tous les éléments valent 0
rand(m,n)	une matrice a m lignes et n colonnes dont les éléments sont générés de manière aléatoire entre 0 et 1

Fonctions propres aux matrices

Fonction	Utilisation
diag(a)	permet d'extraire les éléments diagonaux de la matrice a
triu(a)	permettent d'extraire les parties triangulaire supérieure de la matrice a
tril(a)	permettent d'extraire les parties triangulaire inferieure de la matrice a
size(a)	permet de récupérer les dimensions de la matrice a
det(a)	déterminant de la matrice a
inv(a)	inverse d'une matrice carrée

Exercice d'application

Ecrire un script sous octave qui permet de trouver le nombre d'occurrence d'un nombre dans un vecteur et dans une matrice en suivant les étapes suivantes:

1. **La lecture de la taille d'un vecteur vec et des valeurs de ses éléments.**
2. **La lecture de la taille d'une matrice mat et des valeurs de ses éléments.**
3. **La lecture de nombre nbr.**
4. **Calculer le nombre d'occurrence de nbr dans le vecteur vec et dans la matrice mat.**
5. **Afficher le résultats**