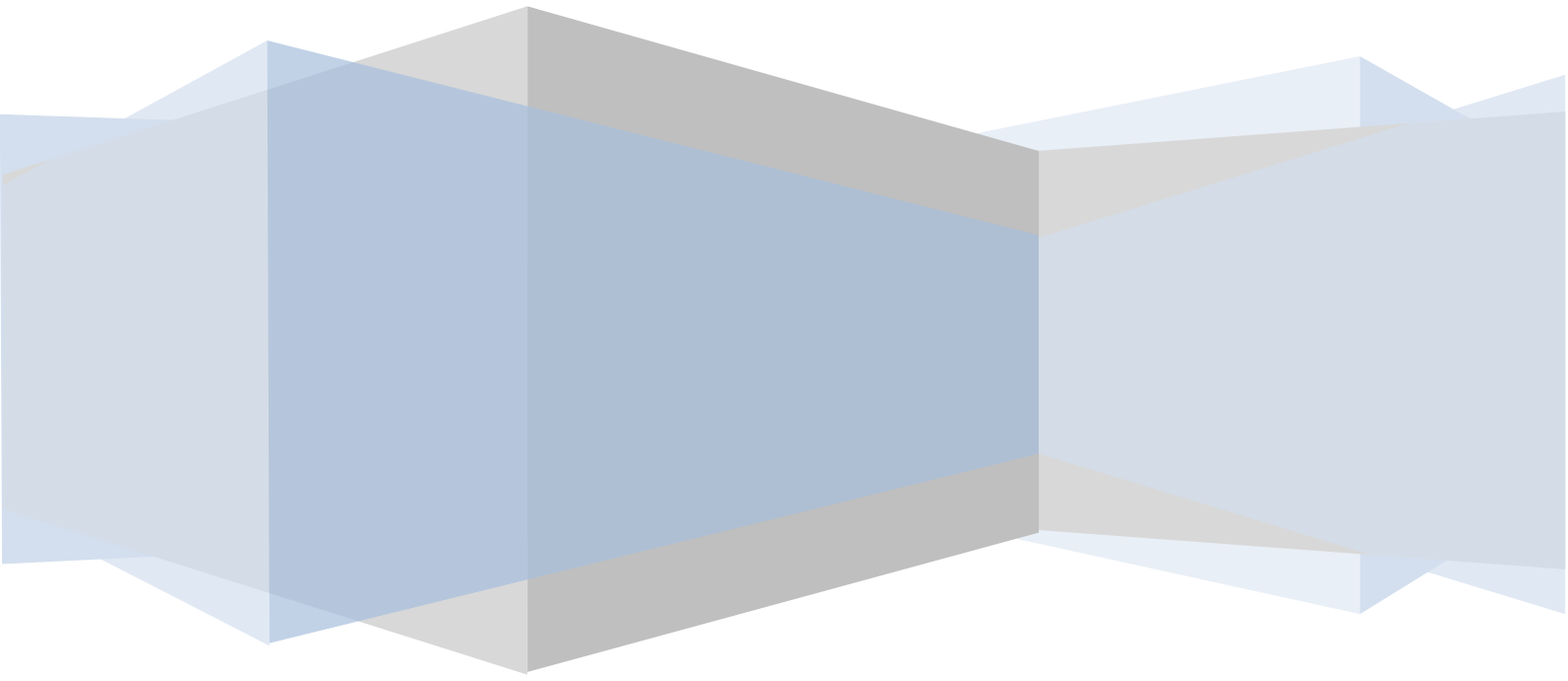
Université Batna2 Mostefa BENBOULAID
Faculté de Mathématiques et Informatique
Département de Mathématiques
Master1 Option: EDP **Matière : INF2**

INTRODUCTION AU LOGICIEL MATRIX-LABORATORY (SUITE)

COURS ET EXERCICES AVEC SOLUTIONS

Prof. Lakhdar DJEFFAL



Graphes et courbes

Le logiciel matrix- laboratory connu sous le nom matlab est un programme interactif de calcul numérique et de visualisation graphique des données comme des histogrammes, discrétisation des équations différentielles (grille), des images et des fonctions etc.

Lorsque l'on veut tracer et visualiser un graphe ou une courbe d'un objet (fonction mathématique f(x)) à 2D, il faut définir obligatoirement deux vecteurs : un pour l'axe des abscisses x et l'autre pour l'axe des ordonnées y=f(x). Comme Le logiciel matlab ne fait pas de calcul formel, il faut donc discrétisé l'axe des abscisses x en utilisant la commande' **linspace**' *dont la syntaxe matlab est :*

>>**linspace**(a,b,n) *;*

Cette commande génère un vecteur de ***n*** *éléments espacés linéairement entre les valeurs* ***a*** *et* ***b*** *tel que* ***a<b****.*

*Une autre méthode pour générer un vecteur espacé linéairement entre les valeurs a et b tel que a<b consiste à utiliser explicitement la commande entre crochet '***[ ]***' avec l'opérateur dit colon' : ' et on écrit*

>>x= [a: d: b] ; avec ***a*** *valeur initiale,* ***b*** *la valeur finale et* ***d*** *le pas de discrétisation ou bien le pas d'incrémentation/décrémentation.*

Exemple

>>x= [0 :4 :20] ; après validation de cette commande le résultat associé à cette ligne bien que il ne s'affiche pas parce que la commande se termine ici par le symbole ' ;' est

x= 0 4 8 12 16 20

>> [20:3 :35]

ans 20 23 26 29 31 34

Dans cette ligne, on remarque que la commande ne se termine pas par le caractère ' ;' et aussi n'est pas affecté à la variable indicée notée par x. Après sa validation et son interprétation par le logiciel matlab, le résultat doit être alors affiché et affectés implicitement à la variable prédéfinie et connue par matlab 'ans'.

Une autre remarque, on constate dans l'affichage du résultat 'ans' la valeur finale 35 n'est pas affichée car si on lui ajoute le pas d'incrémentation alors le résultat dépasse cette valeur finale.

Exercice 7.1

Soit un vecteur x qui représente l'ensemble des valeurs discrétisées d'une fonction continue allant de v1 jusqu'à v2 tel que v1=1, v2=10).

Ecrire un script matlab de deux manières différentes qui génère et affiche les valeurs du vecteur x, de donner sa dimension y et son transposé z.

Solution 7.1

```
>> clear all
>>v1=1 ; v2=10;
>>x=linspace(v1 ,v2 ,10);
>>y=length(x) ;
>>z=x';
>>save  session.mat
>>close all
```

Solution 7.1 bis

```
>> clear all
>>v1=1;
>>v2=10;
```

*>>x= **[**v1:1:v2**]** ou bien >>x=[v1:v2 :1] ça dépend de la version utilisée.*

x=1 2 3 4 5 6 7 8 9 10

*>>**length**(x)*

ans 10

>>z=[1;2 ;3 ;4 ;5 ;6 ;7 ;8 ;9;10];avec le séparateur ';'ou bien z = [1 2 3 4 5 6 7 8 9 10]'

>>save *session.mat*

>>close all

Exercice 7.2

Soit un vecteur v de dimension fini et de type entier.

Donner les solutions des différentes lignes matlab qui suivent après leurs validations et interprétations :

*>>v=*linspace*(70,790,10)*

???????

>>x=v(3 :7)

???????

>>v(4 :2 :8)

????????

>>close all

>>quit

Solution 7.2

```
>>v=linspace(70,790,10)

v=70  150 230 310 390 470 550 630 710 790

>>x=v(3 :7)

x= 230 310 390 470 550

>>v(4 :2 :8)

ans    310 470 630

>>close all
```

*Maintenant, on suppose qu'on sait bien comment générer des vecteurs, alors pour tracer une courbe ou un graphe à 2D (espace x et espace y par exemple) on utilise la commande matlab '***plot***' dont la syntaxe est :*

```
>>plot(x,y)
```

*Cette commande joigne les points d'intersections du graphe d'abscisse x_i et d'ordonnée y_i et ainsi la courbe s'affiche. Notons bien que matlab dispose de capacité énorme à réaliser des graphes de qualité présentant clairement les données en spécifiant des couleurs des styles de trait et l'insertion des légendes et des étiquettes et par conséquent, il est indispensable de procéder à l'amélioration de la lisibilité d'un graphe en mettant surtout des légendes. Les commandes '***xlabel***' et '***ylabel***' permettent d'insérer des textes en légende respectivement sous l'axe des abscisses* x *et l'axe des ordonnées* y *du graphe.*

*L'insertion de la commande '***title***' dans le script matlab permet de donner un titre expliquant le rôle du graphe et il peut être placé à n'importe quelle position grâce à la commande '***text***'.*

*Une autre chose que je pense fondamental à connaitre est bien le traçage d'un graphe histogramme (très utilisé en statistique) connu sous le nom bar-graph en matlab. Ce dernier associe une barre verticale et horizontale de taille proportionnelle à la quantité mesurée en fonction d'une variable descriptive. Les commandes utilisées ici sont '***hist***' et '***bar***'*

Les syntaxes matlab sont :

>>xlabel('legende en x'); >>ylabel('legende en y') ;

>>title('le titre du graphe') ; >>text('posit x, posit y,'l. intitulé du texte')

>>hist(x,n) ; avec x les données et n le nombre de zone de répartition.

>>bar(h); avec h=hist(x,n)/sum(hist(x,n))

Exercice 8.1

Tracer sur l'intervalle [-2,2] (avec un pas de 0.01) la fonction y1=xcos(x) et y2=sin(x)/x tout en insérant les légendes suivantes : sur l'axe des abscisses ' RADIANS' ; sur l'axe des ordonnées 'VALEURS DE FONCTION' et enfin le titre associé au graphe 'GRAPHE DES FONCTIONS y1 et y2'.

Solution 8.1

```
>>clear all
>>x=[-2: 0.01: 2]
>>y1=x*cos(x) ;
>>y2=sin(x)/x;
>>plot(x,y1) ; % pour voir le graphe, enlever le caractère ' ;'
>>plot(x,y2) ; % pour voir le graphe, enlever le caractère ' ;'
>>xlabel('RADIANS')
>>ylabel('VALEURS DE FONCTION')
>>title('GRAPHE DES FONCTIONS : y1 et y2)
>>gtext('avec la souris on place le titre là ou nous voulons') ;
>>close all
```

GRAPHISME A 3D (utilisé pour les fonctions à deux variables)

La manipulation des graphes pour des fonctions à plusieurs variables sont un peu plus compliqués que ceux des fonctions à une seule variable.

*Si on avait créer des vecteurs x qui discrétisait des fonctions continues à une variable en plusieurs valeurs suivant l'axe des abscisses x d'un graphe à 2D, maintenant on doit créer un maillage qui discrétise le plan (x y) associé au graphe à 3D en utilisant la commande '***meshgrid***' puis tracer la courbe bidimensionnelle grâce à la commande '***surf(x, y, f(x,y)***'. On verra bien par la suite lors de l'utilisation des matrices.*

Pour plus de détail sur la représentation graphique par le logiciel matlab, nous renvoyons le lecteur aux adresses électroniques suivantes (et ce ne sont pas les seules) :

http://www.ppur.org

http://mines.utah.edu/gg_computer_seminar/matlab/matlab.html

http://courses.washington.edu/css485/graphg.pdf

OPERATIONS VECTORIELLES

Le logiciel matlab recommande de prendre des mesures de précaution lors de l'utilisation des opérations vectorielles surtout les opérations algébriques. Les opérations d'addition et de différence des vecteurs sont des opérations termes à terme et nécessitent les mêmes dimensions. L'opération de multiplication des vecteurs est identique à celle de l'opération de produit matricielle (voir cours d'algèbre linéaire). Maintenant si les dimensions ne conviennent pas il faut faire alors l'opération de produit terme à terme en remplaçant le symbole de multiplication '' par le symbole '.*'*

Exercice et solution 8.2

Effectuer les opérations algébriques entre les vecteurs_ lignes t et z

```
>>t= [1 3 5 6] ;
>>z= [10 13 15 16] ;
>>s=2*t+z
```

s=12 19 25 28

>>t-z

ans -9 -10 -10 -10

*>>t*z*

Erreur, il faut remplacer le symbole '' par le '.*' et écrire y=t .*z pour avoir le résultat y= 10 39 75 96*

Remarques

- *Une particularité du logiciel matlab est de permettre d'effectuer des opérations d'addition et de soustraction de manière globale sur les vecteurs et non élément par élément.*
- *L'opération algébrique globale '*' entre vecteurs doit se faire avec vérification de la conformité des dimensions. Comme elle peut être faite grâce a la commande 'cross(x, y)'.*
- *On peut aussi appliquer toutes les fonctions mathématiques usuelles et les plus courantes et d'autres opérations non algébriques (fusion des vecteurs par exemple) pour les vecteurs. Citons par exemple :*
- sum*(x), sum(x+y) : somme des éléments du vecteur entre eux.*
  prod*(x) : produit des éléments du vecteur entre eux.*

  mean*(x) : moyenne des éléments du vecteur x*

  sort*(x) : donne l'ordre croissant des éléments du vecteur*

  fliplr*(x) : renverse l'ordre des éléments du vecteur x*

  max*(x) et min(x) : donnent respectivement la valeur maximale et minimale du vecteur x*

cos*(x), sin(x), tan(x) : retournent les résultats de chaque fonction trigonométrique associée au vecteur x (toutes les opérations sont appliquées à tous les éléments du vecteur x en un seul clic, c'est-à-dire en même temps et non élément par élément).*

Exercice 8.3

Reprenons l'exercice 7.1 et calculer la racine carrée du vecteur x, la fonction trigonométrique cos(x) et aussi le carré du vecteur x.

Solution 8.3

```
>>clear all
>>x= linspace(1,10,10) ;
>>sqrt(x)
Ans  1.00  1.41  1.73  2.00  2.23  2.44  2.64  2.82  3.00  3.16
>> cos(x)
ans    0.54  -+0.41  -0.99  -0.65  0.2  0.96  0.75  -0.14  -0.91  -0.83
>>x^2
Ans   1 4 9 16 25 36 49 64 81 100
>>quit
```

*NB. Attention ne pas oublier que le **logiciel matlab** est un programme **interactif de calcul scientifique**. C'est-à-dire chaque opération (ligne matlab) écrite, après sa validation(en tapant sur le bouton du clavier ENTRER) la solution est automatiquement calculée et donnée. On n'attend pas l'écriture complète du programme puis on lance la compilation et l'exécution pour calculer et donner l'ensemble des solutions.*

MATR ICES en Matlab

Matlab est une abréviation de Matrix Laboratory. Un laboratoire de recherche scientifique qui a pour mission la résolution de nombreux problèmes de mathématiques appliquées notamment de calcul matriciel, traitement de signal, traitement d'image et visualisation graphique sous forme de logiciel de programmation. Il dispose de potentialité importante de commandes et d'instructions servant à manipuler et utiliser les matrices. Seules les fonctionnalités les plus

courantes sont présentées ici et nous supposons que les lecteurs sont bien familiers avec les notions de bases d'algèbre linéaire, d'analyse et de statistique.

Définition

Une matrice sous matlab est un ensemble de données contiguës de même type. Elle est définie explicitement à l'aide de la commande crochet '[. , .]' Comme *les vecteurs mais avec deux dimensions (dimension ligne et dimension colonne). C'est le cœur de matlab.*

>>A=[10 20 ;40 30] ; A est variable à deux indices. La première est appelée indice lignes notée généralement par une variable i de type entier et qui contiennent les valeurs 10, 20 et 40, 30, la deuxième est indice colonnes notée par la variable j de type entier et qui contiennent les valeurs 10, 40 et 20, 30.

>>B=[1 2 3;4 5 6 ;7 8 0] ; % ceci est équivalent à écrire :

```
>>B= [1 2 3
      4 5 6
        7 8 0] ;
```

Ceci génère deux matrices carrées : A de taille 4 et B de taille 9.

Si on écrit B(i, j)=8 cela veut dire que la valeur prise en compte est celle qui appartient à l'intersection de la ligne i et la colonne j de la matrice B tels que i=3 et j=2.

La liste des éléments de la matrice sont insérés à l'intérieur de la commande matlab crochet et sont séparés soit par des blancs soit par des virgules. Le symbole ' ;' sépare les lignes de la matrice et on peut les séparées aussi en tapant le bouton ENTREE du clavier à chaque fin d'insertion des éléments d'une ligne. Notons bien que l'ordre d'insertion des éléments de la matrice doit être respecté.

Lors de l'utilisation de la structure de données matrice, alors il est inutile de déclarer ou définir sa dimension et son type au préalable comme pour les autres langages de programmation, ceci s'établie automatiquement à partir de l'expression mathématique définissant la matrice ou à partir de ses données insérées dans la commande matlab sinon si pour des raisons quelconques, la taille ou la dimension par exemple est exigée alors le logiciel

*matlab dispose des commandes qui répond à ceci. La commande '***size***' à pour rôle de déterminer la taille d'une matrice donnée et même en détaille.*

*>>B=***[***1 2 3 ; 4 5 6 ; 7 8 0 ; 0 0 1***]**

*>>***[***m,n***]=size***(B)***

m=4 n=3 % m désigne le nombre de lignes, n le nombre de colonnes ceci est équivalent à « écrire aussi par la syntaxe suivante :

*>>m=***size***(B,1)*

m= 4

>>n=size(B,2)

n=3

Encore une fois, une autre particularité qu'offre le logiciel matlab est la possibilité de manipuler des blocs de données en un seul coup (manipulation globale). Il existe des commandes qui permettent d'utiliser et manipuler un groupe de données d'une matrice en même temps grâce à l'opérateur colon ' **:** *', chose que les autres langages de programmation n'autorisent pas. Par exemple on peut accéder et lire une ligne ou colonne entière en une seule opération et/ou l'afficher sans répétition.*

>> B= [1 2 3 ; 4 5 6 ; 7 8 0 ; 0 0 1] ;

>>x=B (: , 3)

x= (3 6 0 1)' % En une seule opération d'accès, s'affiche la 3ème colonne de B

>>B (:) % Accès et affichage de toute la matrice B

B=1 2 3

4 5 6

7 8 0

0 0 1

>>B (2, :)

ans=4 5 6 % Accès et affichage de la 2ème ligne de B en un seul coup

Toutes les combinaisons d'accès, d'affichage et de manipulation sont possibles. Par exemple

>>B= [1 2 3 ; 4 5 6 ; 7 8 0 ; 0 0 1] ;

>>B ([1 3], :) %Accès et affichage des lignes 1 et 3 de B

ans 1 2 3

7 8 0

>>C=B (2 :4, 1:3)

4 5 6

C= 7 8 0

0 0 1

*>>**diag** (C)*

ans 4 8 1 % la diagonale principale de la matrice C.

*matlab ne se limite pas à la diagonale principale, la commande **diag** peut avoir un deuxième paramètre k qui indique l'accès et l'affichage de la kème sur- diagonale (si k>0)ou sous diagonale (si k<0)de la matrice.*

*>>y=**diag** (C,2)*

y= 6

*>> y1=**diag** (C,-1)*

y1= 7 0

*Le logiciel matlab dispose aussi des commandes qui permettent d'obtenir des matrices triangulaires inférieures et supérieures qui sont respectivement '**tril**' et '**triu**' d'une matrice carrée donnée.*

*>>MSup= **tril** (C)*

4 0 0

Msup= 7 8 0

0 0 1

*>>Minf= **triu**(C)*

$$Minf = \begin{matrix} 4 & 5 & 6 \\ 0 & 8 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Comme pour la structure de données vecteur, la transposé d'une matrice s'obtient de la même façon en employant soit le symbole ' ;' ou bien le symbole prime noté par « ' ».

Certaines matrices se construisent automatiquement à l'aide de commandes offertes gratuitement. On cite par exemple : matrice unité, matrice nulle, matrice identité dont les syntaxes sont respectivement :

*>>***ones***(m,n) ;* *>> **zeros**(m,n);* *>>**eye**(n);*

Exercice 9.1 ***(Devoir à la maison)***

Soit une équation aux dérivées partielles de transport (EDP à 1D prendre par exemple la dimension n=5). Ecrire un script matlab qui permet de construire une matrice correspondant à sa discrétisation par la technique de différence finie.

Même chose pour une équation aux dérivées partielles de Poisson (EDP à 2) prendre par exemple les dimensions n_lignes=3 ; m_colonnes=4.

OPERATIONS ET FONCTIONS PORTANT SUR LES MATRICES

Les opérations matricielles sont à la base du calcul sous le logiciel Matlab. Nous n'allons pas faire un cours complet de mathématiques mais de donner grosso modo les notions et les concepts mathématiques fondamentaux pour réaliser les opérations matricielles qui sont issue de l'algèbre linéaire tel que l'inversion, produit, diagonalisation etc.

Comme d'habitude, nous commençons par les opérations arithmétiques usuelles avec des opérandes de type matrice.

Addition et Soustractions de matrices

Comme pour le cas de la structure de données 'vecteur', la somme et la soustraction des matrices sont des opérations terme à terme et nécessitent des dimensions identiques. La syntaxe est la même que ceux des scalaires et vecteurs.

Exercice 9.2

Soit A une matrice de dimension (2,3) et B une autre matrice de même dimension données par $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$; $B = \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$. *Ecrire un script qui permet de déterminer la somme et la soustraction des deux matrices A et B*

solution 9.2

```
>>clear all
>>A=[1 2 3 ; 4 5 6] ;
>>B=[7 , 8,  9
      10  11  12] ;
>>Somme=A+B
```

$$Somme = \begin{pmatrix} 8 & 10 & 12 \\ 14 & 16 & 18 \end{pmatrix}$$ %Somme (i, j) =A (i, j) +B (i, j)

```
>>Souse=A-B
```

$$Souse = \begin{pmatrix} -6 & -6 & -6 \\ -6 & -6 & -6 \end{pmatrix}$$ % Souse (i, j) =A (i, j) +B (i, j)

```
>>close all
```

La somme et la soustraction des matrices se fait élément par élément et en ordre grâce aux indices par exemple i et j de types entiers et qui sont identiques aux deux matrices, c'est-à-dire

l'élément de A d'indice i et j avec l'élément de B de même indices (même valeurs des indices i et j.

Opération Produit matricielle

L'opération de produit matricielle peut se faire de deux manières différentes soit terme à terme comme les opérations arithmétiques de somme et soustraction et dans ce cas il faut utiliser l'opérateur de produit ' ***.****' , soit utiliser l'opérateur de multiplication '* ******* *' et dans ce cas il faut que la règle de dimensionnalité des matrices doit être obligatoirement vérifiée, c'est-à-dire le nombre de colonne de la première matrice soit égale au nombre de lignes de la deuxième matrice. Si A=* a_{ij} *et B=*b_{ij} *tel que m<=i<=n et n<=j<=p. Le produit A*B est une matrice appelée par exemple Mat de taille ou de dimension (m*p) tel que*

$$Mat = \sum_{0<=k<=n} a_{ik} * b_{kj} \quad pour\ m <= i <= n \quad et \quad n <= j <= p$$

Reprenons l'exercice 9.2 soit la matrice A = a_{ij} *et B =*b_{ij} *tel que 1<=i<=2 et 1<=j<=3. Le produit A*B ne peut pas se faire car les dimensions ne sont compatibles. Par contre, il peut être calculé terme à terme en changeant le signe de multiplication '* ******* *' par l'opérateur ''* ***. **** *'' et voici un exemple de script matlab suivant:*

```
>>clear all

>>A=[1 2 3 ; 4 5 6] ;

>>B=[7  89  ; 10  11  12] ;

>>prod1=A*B     %  Attention ici le produit '*' ne se fait pas élément par élément des mêmes indices i et j.
```

Erreur le nombre de colonnes de A est non égale au nombre de lignes de B

```
>>prod2=A.*B  %   ici le produit '.*' se fait élément par élément des mêmes indices
```

$$Prod2 = \begin{pmatrix} -6 & -6 & -6 \\ -6 & -6 & -6 \end{pmatrix}$$

```
>>close all

>>quit
```

Notons bien que le produit matriciel est non commutatif.

Opérations Division et inversion matricielles

*L'opération de division entre matrices carrées se définit à partir de la fonction inverse matricielle et qui s'écrit par la commande matlab ' **inv(**nom de la matrice**)** '*

Soit la matrice A= a_{ij} tel que m<=i, j<=n (matrice carrée), on définit A^{-1} l'inverse de A si elle existe bien sûr par le produit entre A et A^{-1} qui vérifie l'égalité suivante $A*A^{-1}=A^{-1}*A=I$ où I est la matrice identité.

Le logiciel matlab donne aussi la possibilité d'utiliser l'opérateur anti-slash noté par ' \' appelé aussi la division à gauche pour effectuer l'opération de division matricielle.

Cette opération est très importante et elle est utilisée surtout pour résoudre des problèmes d'analyse numérique qui utilisent des systèmes d'équations.

Exercices 9.3

Soit la matrice $A = \begin{pmatrix} 4 & 2 \\ 0 & 1 \end{pmatrix}$ *et* $B = \begin{pmatrix} 1 & 2 \\ 5 & 4 \end{pmatrix}$

Calculer la division matricielle entre A et B.

Solution 9.3

La division doit être définit à partir de la fonction inverse de la matrice B. Il faut que la matrice B soit inversible. Aussi il faut vérifier la compatibilité de dimensionnalité des deux matrices A et B. Le script est donnée par

```
>>clear all

>>A=[4 2, 0 1] ;

>>B=[1 2, 5 4] ;

>>D=A*inv(B)
```

$$D = \begin{pmatrix} -1 & 1 \\ 0.88 & -0.16 \end{pmatrix}$$

```
>>close all
```

Fonctions portant sur les matrices

Il est bien entendu possible d'utiliser le logiciel matlab pour calculer directement des fonctions mathématiques appliquées aux matrices.

Nous présentons ici seulement des fonctions spécifiques qui sont extrêmement utiles :

*Commençant par exemple à présenter le cas le plus couramment rencontré dans les problèmes matriciels qui est la résolution d'un système d'équations linéaires auquel correspond un formalisme matriciel. Comme c'est indiqué auparavant le logiciel matlab permet de d'effectuer des inversions de matrices. Par conséquent il peut résoudre directement et rapidement un système d'équations linéaires en un seul clic ou opération à condition que la matrice associée est inversible (déterminant de cette matrice est non nul). Par exemple, pour résoudre un système d'équations de type A*X=B, Avec A matrice carrée quelconque de taille (m*n), m lignes et n colonnes, X et B deux vecteurs de n colonnes chacun, la solution est donnée par* $X=A^{-1}*B$*, avec* A^{-1} *inverse de la matrice A. En matlab on écrit tout simplement la ligne suivante :*

*>>X=**inv**(A)*B ; ou bien >>X=A\B ; avec **inv(A):** fonction mathématique appliquée à la matrice A et l'opérateur générique **' \'** représente la division à gauche.*

Maintenant, et par applications des méthodes numériques, si la matrice associée au système d'équations à des propriétés particulières, le logiciel matlab dispose de fonctions qui donne rapidement la solution. Par exemple,

- *Si la matrice A est symétrique, hermitienne, définie positive*
- *Si la matrice A est creuse ou dense, de grande taille*
- *Si la matrice A n'est pas carrée*
- *Etc*

*Pour chacun de ces cas et tout dépend de la structure de la matrice A du système, des algorithmes sont mis en œuvres et qui sont représentés par des simples fonctions offerts par matlab tel que **chol**, **lu**, **qr**.*

*La fonction **chol** : méthode numérique de Cholesky*

La fonction ***lu*** *: méthode de Factorisation LU en utilisant la méthode d'élimination de Gauss avec pivot*

La fonction ***qr*** *: méthode de résolution numérique de type QR*

Citons aussi quelques fonctions les plus fréquemment utilisées:

- ***det(A)*** *: calcul et renvoie le déterminant de la matrice A*
- ***eig(A)*** *: calcul et renvoie deux matrices, la première est une matrice dont les colonnes représentent, et la deuxième est une matrice dont la diagonale contient les valeurs propres de A*
- ***diag(A)*** *: fonction qui retourne la diagonale principale de la matrice A*
- ***rank(A) :*** *renvoie le rang de la matrice A*
- ***trace(A) :*** *renvoie la trace de la matrice A*
- ***expm(A) :*** *renvoie l'exponentielle de la matrice A*
- ***norm(A) :*** *renvoie la norme 2 de la matrice A*
- ***norm(A,1), norm(A,2), norm(A,inf)*** *renvoie respectivement les normes de type 1,2 et infinie de la matrice A.*

Le logiciel matlab dispose aussi des fonctions et commandes qui utilisent des méthodes itératives comme les méthodes de gradient conjugué pour résoudre des problèmes d'optimisations qui peuvent avoir des formalismes matriciels. Nous citons par exemple les fonctions ***cgs, bicg, bicgstab.***

*Aussi le problème des polynômes qui ont un formalisme matriciel (vectoriel) peut être résolu en utilisant la fonction '****polyval****' en des points donnés. Soit un polynôme* $P_n(x)$ *de degré n donné par*

$P_n(x) = \sum_{i=0}^{n} a_i * x^i$ *défini par un vecteur p de (n+1) points et dont les coefficients* a_i *,* $0<=i<=1$

sont rangés dans l'ordre suivant : $p(1)=a_n$, $p(2)=a_{n-1}$, … , $p(n)=a_1$, $p(n+1)=a_0$

Pour calculer et évaluer le polynôme $P_n(x)$ *donné, matlab utilise la fonction polynomiale en des points '****polyval****' dont la syntaxe est* ***polyval(****p,x****)*** *et la fonction* ***roots(p)*** *pour obtenir la solution du polynôme* $P_n(x)$*.*

La forme canonique du polynôme est représentée par la fonction ***poly(****variable_solution****)*** *avec 'variable_solution' représente les racines du* $P_n(x)$

(Pour tracer le graphe d'un polynôme on écrit la forme suivante:

*>>**fplot('polyval([a_n, … a_0], x)', [x_{min}, x_{max}])***

Remarque

*Les fonctions ou commandes usuelles dérivées, intégrales des fonctions mathématiques f(x) données sous forme de polynômes canoniques ou sous forme d'équations algébriques ou autres) sont aussi offert par matlab et dont les syntaxes sont respectivement **diff**(f,x) et **int**(f,x).*

Exercice 10.1

Soit le système d'équations linéaires

$$\begin{cases} -3X_1 + 4X_2 = 3 \\ 2.5X_1 - X_2 - 0.5X_3 = 1 \\ 7X_2 - 7X_3 + X_4 = 1 \\ 13.5X_3 - 17X_4 + 4.5X_5 = 1 \\ -X_4 + X_5 = 0 \end{cases}$$

Ecrire une suite de lignes matlab qui calcule la solution du système donné.

Solution 10.1

```
>>clear all

>>A=[-3 4 0 0 0, 2.5 -1 -0.5 0 0, 0 7 -7 1 0,0 0 13.5 -17 4.5, 0 0 0 -1 1] ;

>>B=[3 1 1 1 0] ;

>>det(A) ; %ici normalement on utilise le test de condition alternative if - else qui vérifie si A est inversible pour continuer sinon on affiche pas de solution)
```

```
>>X=inv(A)*B

1.609 1.957 2.131 2.222 2.222

>>close all

>>quit
```

Exercice 10.2

Soit le polynôme $P_n(x) = x^2 - 1$. *Ecrire un script matlab qui permet de :*

- *Définir et évaluer le polynôme* $P_n(x)$
- *tracer son graphe.*
- *Donner sa forme canonique*

Solution 10.2

```
>>clear all

>>p=[1 0 -1] ;

>>z=polyval (p,0)

z=-1

>>t=polyval (p,[-2, -1, 0, 1, 2])

t=3  0  -1  0  3

>>fplot('polyval([1 0 -1], x )' , [-3, 3]), grid);

>>sol=roots(p)  % Les racines du polynôme Pn(x)

sol= -1.000   1.000

>>poly(sol)     % donne forme canonique du polynôme Pn(x)

ans   1.000   0.000 -1.000

>>quit
```

LES STRUCTURES DE CONTROLE

*La **programmation**, appelée aussi **codage**, est l'ensemble des activités qui permettent l'écriture et la rédaction des codes sources appelés programme informatique. Pour écrire un programme, on utilise un langage de programmation (logiciel) tel que matlab.*

*En programmation informatique, les programmes doivent être capables de prendre des décisions. Pour y parvenir, matlab comme tous les langages de programmation utilise ce qu'on appelle des **structures de contrôle.** L'utilisateur est appelé à savoir bien manier ces structures de contrôle. Elles lui serviront durant toutes sa vie car lorsqu'on écrit un programme, on se retrouve généralement face à l'étude de plusieurs situations qui ne peuvent pas être traitées seulement par les séquences d'actions simples de solution. Puisqu'on a plusieurs situations, et qu'avant l'exécution, on ne sait pas à quel cas de figure on aura à exécuter, dans les programmes on doit prévoir tous les cas possibles. Ce sont seulement les structures de contrôles qui le permettent, en se basant sur ce qu'on appelle conditions.*

Définition

*Une **structure de contrôle** est une instruction particulière d'un langage de programmation impératif pouvant dévier le flot de contrôle du programme, la contenant lorsqu'elle est exécutée.*

Principe

Le logiciel matrix_laboratory offre des constructions plus élaborées comme les structures ou primitives conditionnelles simple et alternatives (if, if–else, switch, ...), les boucles répétitives (while, do–while, for, ...) qui permettent de contrôler l'exécution des programmes, vérifier leurs validités et leurs terminaison (code fini) et de prendre des décisions. Pour effectuer ceci, le logiciel matlab utilise des opérateurs de comparaisons et de relations à savoir :

== Opérateur d'égalité

< Opérateur d'inégalité inférieure

> Opérateur d'inégalité supérieur

>= Opérateur double inégalité supérieure ou égale

<= Opérateur double 'inégalité inférieure ou égale

! Opérateur de non égale (est différent de)

And opérateur logique de conjonction

Or opérateur logique de disjonction

Not opérateur de négation

A/ Primitives conditionnelles simples et alternatives

L'instruction simple '' if '' évalue une expression mathématique logique et exécute un ensemble d'actions élémentaires écrite entre début et fin de cette instruction lorsque le résultat de l'expression logique est vrai. Dans le cas contraire aucune des instructions se trouvant à l'intérieur du bloc if ne sera exécutée.

Les instructions alternatives elseif et else permettent d'imbriquer des boucles conditionnelles supplémentaires, et elles décrivent des blocs d'instructions différents dont l'exécution est alternative, c'est-à-dire quelque soit le résultat de la condition se trouvant devant l'instruction ''if'' une seule séquence d'instruction doit être exécutée. L'instruction end, est la fin du corps de la condition if.

Les Syntaxes des deux conditions sont

```
if < conditions >
```

- *Ensemble d'actions (instructions à exécuter si les conditions sont vraies)*

```
end
```

```
if < conditions >
  instructions à exécuter (si les conditions sont vraies)
else   [elseif  conditions]
  instructions à exécuter (si les conditions ne sont pas vraies)
```

```
end
```

Exercice 11.1

Soit n un scalaire donné. Ecrire une fonction matlab qui permet de vérifier si n donné est un entier naturel pair ou impair.

Solution 11.1

```
>> clear all
>>function parité (n) % ici on suppose que le scalaire n est connu, donc la lecture est faite
>>if rem (n,2) == 0  % un test de contrôle sur le reste de la division entière de n sur 2
  >>disp (' n donné est un nombre pair')  % imprimer le message : le scalaire n est pair
>>else
>>disp ('n donné est un nombre impair')  %imprimer le message : le scalaire n est impair
>>end
>>quit
```

L'exécution donne :

```
Par exemple on donne n=2 et n=5
>> parité (2)
n donné est un nombre pair


>> parité (5)
n donné est un nombre impair
```

- *rem : retourne le reste de la division entière de deux nombres,*
- *disp : affiche le message spécifié sous forme d'une chaine de caractères écrit entre*

les caractères apostrophes

Exercice 11.2

Soit A(3,3) une matrice carrée donnée. Ecrire une fonction matlab qui permet

- *d'afficher si la matrice A est inversible*
- *calculer son inverse*
- *d'afficher sa matrice inverse*

Solution 11.2

```
>> clear all

>>function inverse(A) % ici on suppose que la matrice carrée A est connue et la lecture par la commande matlab 'input' est faite

>>if det (A) == 0  % un test de contrôle sur le déterminant de la matrice A

  >>disp (' A est non inversible car son déterminant est nul')

>>else

>>disp (' A est inversible et égale à :')

>>B= inv(A)  % la commande inv calcul l'inverse de la matrice A et sera affectée a la variable indiciaire B

B=  ……..  % ça dépend de la matrice A donnée, son inverse est affecté à B et s'affiche ici

>>end

>>close all

>>quit
```

B/ Les Commandes ou Instructions « switch » et « case »

L'instruction ''switch'' est utilisée lorsqu'on se retrouve devant plusieurs situations alternatives et qu'on ne peut exécuter qu'une seule. C'est-à-dire exécute une seule séquence

d'instruction (ensemble d'actions élémentaires) relatives à la valeur prise par une variable. Les instructions ''case et otherwise'' délimitent les groupes des instructions.

Le ''end'' est obligatoire à la fin de la structure.

Syntaxe

```
switch < variable_de_choix >
 case constante 1
   séquence d'instructions 1
 case constante 2
   séquence d'instructions 2
 ...
 case constante N
   séquence d'instructions N
 otherwise
   séquence d'instructions par défaut
end
```

Ci-dessous, un exemple de programme qui affiche un texte en fonction de la valeur de nombre donné à chaque fois.

```
>>function exemple_ (nombre)
>>switch nombre
   >>case -1
      >> disp('valeur négative 1');
   >>case 0
      >> disp('valeur zéro');
```

```
   >>case 1
      >>disp('valeur positive 1');
   >>otherwise
     >> disp('Autre valeur');
end
```

L'exécution donne :

```
>> exemple_switch(1)
valeur positive un

>> exemple_switch(0)
valeur zéro

>> exemple_switch(3)
Autre valeur
```

B/ Primitives conditionnelles répétitives

Dans les problèmes quotidiens, on ne traite pas uniquement des séquences d'actions, sous ou sans conditions, mais il peut être fréquent d'être obligé d'exécuter un traitement (séquence d'actions), plusieurs fois. En effet, pour saisir les N notes d'un étudiant et calculer sa moyenne, on est amené à saisir N variables, puis faire la somme et ensuite diviser la somme par N. Cette solution nécessite la réservation de l'espace par la déclaration des variables, et une série de séquences d'écriture/lecture. Ce problème est résolu à l'aide des structures répétitives. Celles ci permettent de donner un ordre de répétition d'une action ou d'une séquence d'actions une ou plusieurs fois.

L'instruction « for »

L'instruction de la boucle ''for'' permet de répéter un nombre de fois connu auparavant une séquence d'instructions. Le mot réservé ''end'' est obligatoire à la fin de la structure.

```
for variable = valeur Début : pas  : valeur Fin
  Instructions
end
```

L'exemple suivant permet de générer, à l'aide de la boucle for, les normes de type 1 pour les colonnes paires et les normes de type inf pour les colonnes impaires d'un matrice donnée A (i, j) de dimension m lignes et n colonnes.

Fichier normes.matlab

```
>>clear all
function x = normes un_inf
>>n = 10;
>>z=1 ;
>>zz=z ;
>>x1 = [ ]; % initialement le vecteur correspondant aux normes_1 des colonnes paires de la matrice A (n, n) est vide
>>xinf == [ ]; % initialement le vecteur des normes_inf des colonnes impaires de A est vide
>>for j=1:n
    >>if rem(j,2)==0
            >>for t=1:n
                >>y1[t]=A[t, j] ;
            >>end  % fin de for t
```

```
        >>x1[z] =norm (y1, 1);
        >>z =z+1;
    >>else
        >>for t=1:n
            >>yinf[t]=A[t, j] ;
        >>end   %fin de for t
        >>xinf[zz ] =norm (yinf, inf);
        >>zz=zz+1;
    >>end  % fin de if-else
>>end  %   fin de for    j
>>disp('les normes de type 1 des colonnes paires de A sont respectivement', x1);
>>disp('les normes de type inf des colonnes impaires de A sont respectivement', xinf);
>>close all
><quit
```

L'exécution donne :

```
>> normes un_inf
 x1 =  ……..  %Des valeurs normes de type 1 associées aux colonnes paires de A
yinf = …… % Des valeurs normes  de type inf associées aux colonnes impaire de A
```

L'instruction « while »

L'instruction de la boucle ''while'' permet de répéter une séquence d'instructions tant qu'une condition est vérifiée. Le mot réservé '''end'' est obligatoire à la fin de la structure.

while conditions

- *ensemble d'actions (instructions)*

% parmi les instructions à exécuter tant que la ou les conditions écrites devant while sont vrais, il existe une qui modifie le résultat

end

Les entrées/sorties

- *Entrée au clavier et clics souris*

La commande input permet de demander à l'utilisateur d'un programme de fournir des données. La syntaxe est var = input(' données '). La phrase 'données' est affichée et ***MATLAB*** *attend que l'utilisateur saisisse une donnée au clavier. Cette donnée peut être une valeur numérique ou une instruction* ***MATLAB***

Le programmeur peut donc saisir des informations au clavier grâce à la commande x = input(...). La commande [x y] = ginput(n) retourne les vecteurs x et y des coordonnées de n clics souris sur la fenêtre courante.

>> x = ***input*** *(' Saisir une valeur de x : ') ;*
% Saisir une valeur pour la variable x par exemple: 20
>> x
x = 20
>> x = ***input*** *(' Saisir un message ' , ' s ') ; %pour saisir un message par exemple BONJOUR*
>> x
x = BONJOUR

- ***Sortie à l'écran***

Pour afficher quelque chose à l'écran, l'utilisateur peut soit utiliser les commandes disp, qui affiche le contenu d'une variable (chaîne de caractères, vecteur, matrice...), ou fprintf, qui fonctionne comme la fonction printf en C.

```
>>clear all
>> X = [12   34   2  95] ;
>> disp (X)
      ans  12    34     2    95
>>quit
```

La commande sprintf fonctionne de la même manière que fprintf, à la seule exception qu'elle retourne le résultat de l'affichage sous forme d'une chaîne de caractères.