MODULE : LOGICIELS DE SIMULATION

Université de Biskra
Faculté de la science et de la technologie
Département de Génie électrique
Filière Electrotechnique

Troisième année ELT

Cours : Logiciels de Simulation

Objectifs:

- Connaitre les logiciels de simulation
- Etre capable de reproduire un système électro-énergétique en vue de son étude et sa simulation

Enseignant B.KIYYOUR

L.B

Sommaire

I- Programmation à l'aide de Matlab (opérations simples)

II- Modélisation et implémentation d'un système composé électrique.

III- Utilisation de Matlab-Simulink et SimPowerSystems.

III-2-2 Construction d'un model par Simulink

III-2-2-1 Partie principale

III-2-2-2 Partie complémentaire

III-2-3 Les démarches nécessaires pour effectuer une simulation

III-2-4 Etapes de la Modélisation des équations mathématique sous simulink

III-2-5 Exemples :

Exemple 1 : Résolution d'une équation différentielle par simuulink

Exemple 2 : Système du seconde ordre (circuuit RLC)

Exemple 3 : simulation de la machine a couurant continu à aimant permanent

III-3 SimPowerSystems :

III-3-1 Définition

III-3-2 Librairie simpowersystems

III-3-3 Principales caractéristiques

Exemple : Redresseur simple alternance

III-4: ACQUISITION DE RESULTATS DE SIMULATION

1- sauvegarder des résultats de simulation sous Matlab

2- sauvegarder des résultats de simulation sous Simulink

VI- AUTRES LOGICIELS (PSPICE, PSIM, SCILAB, WORKBENCH, PROTEUS,...).

VI-1 Quelques simulateurs de circuit électrique

VI-2 PSpice (Orcad)

VI-3 Psim (Powersim)

VI-4 Scilab

VI-5 Proteus

III- Programmation à l'aide de Matlab (opérations simples)

I-1. Introduction :

MATLAB est une abréviation de *Matrix LABoratory*. Écrit à l'origine, en Fortran, par *C. Moler*. La version actuelle, écrite en C par the MathWorks Inc., c'est un logiciel de calcul numérique et symbolique. La majorité de ces instructions (fonctions) sont basées sur un calcul matriciel simplifié. Grâce à ses fonctions spécialisées, MATLAB est considéré comme un langage de programmation adapté pour les divers problèmes d'ingénierie.

MATLAB est un environnement puissant, complet et facile à utiliser destiné au calcul scientifique. Il apporte aux ingénieurs, chercheurs et à tout scientifique un système interactif intégrant calcul numérique et visualisation. C'est un environnement performant, ouvert et programmable qui permet de remarquables gains de productivité et de créativité.

Grâce aux fonctions graphiques de MATLAB, il devient très facile de modifier interactivement les différents paramètres des graphiques pour les adapter selon nos souhaits.
L'approche ouverte de MATLAB permet de construire un outil sur mesure. On peut inspecter le code source et les algorithmes des bibliothèques de fonctions (Toolboxes), modifier des fonctions existantes et ajouter d'autres.

Le Toolboxes (ou Boîtes à Outils) est un ensemble d'outils spécifiques à des domaines d'application spécifiques. Indispensables à la plupart des utilisateurs, les Boîtes à Outils sont des collections de fonctions qui étendent l'environnement MATLAB pour résoudre des catégories spécifiques de problèmes. Les domaines couverts sont très variés et comprennent notamment le traitement du signal, l'automatique, l'identification de systèmes, les réseaux de neurones, la logique floue, le calcul de structure, les statistiques, etc.

En complément du noyau de calcul MATLAB, l'environnement comprend des modules optionnels qui sont parfaitement intégrés à l'ensemble :

1. Une vaste gamme de bibliothèques de fonctions spécialisées (***Toolboxes***)
2. ***Simulink***, un environnement puissant de modélisation basée sur les ***schémas-blocs*** et de simulation de systèmes dynamiques linéaires et non linéaires
3. Des bibliothèques de blocs ***Simulink*** spécialisés (***Blocksets***)
4. D'autres modules dont un Compilateur, un générateur de code C, un accélérateur,...
5. Un ensemble d'outils intégrés dédiés au Traitement du Signal : le ***DSP Workshop***.

I-2 Quelles sont les particularités de MATLAB ?

MATLAB permet le travail interactif soit en mode commande, soit en mode programmation : tout en ayant toujours la possibilité de faire des visualisations graphiques. Considéré comme un des meilleurs langages de programmations (C ou Fortran), MATLAB possède les particularités suivantes par rapport à ces langages :

1. la programmation facile,
2. la continuité parmi les valeurs entières, réelles et complexes,
3. la gamme étendue des nombres et leurs précisions,
4. la bibliothèque mathématique très compréhensive,
5. l'outil graphique qui inclut les fonctions d'interface graphique et les utilitaires,
6. la possibilité de liaison avec les autres langages classiques de programmations (C ou Fortran).

Dans MATLAB, aucune déclaration n'est à effectuer sur les nombres. En effet, il n'existe pas de distinction entre les nombres entiers, les nombres réels, les nombres complexes et la simple ou double précision. Cette caractéristique rend le mode de programmation très facile et très rapide.

La bibliothèque des fonctions mathématiques dans MATLAB donne des analyses mathématiques très simples. En effet, l'utilisateur peut exécuter dans le mode commande n'importe quelle fonction mathématique se trouvant dans la bibliothèque sans avoir à recourir à la programmation.
Pour l'interface graphique, des représentations scientifiques et même artistiques des objets peuvent être créées sur l'écran en utilisant les expressions mathématiques. Les graphiques sur
MATLAB sont simples et attirent l'attention des utilisateurs. vu les possibilités importantes offertes par ce logiciel.
Il existe deux modes de fonctionnement:

- ✓ **Mode interactif**

MATLAB est un interpréteur: les instructions sont interprétées et exécutées ligne par ligne. Noter que Matlab exécute les instructions au fur et à mesure qu'elles sont données par l'usager.

- ✓ **Mode exécutif**

Dans un fichier M (M file) l'exécution se fait ligne par ligne.

Lancement de Matlab
Matlab affiche plusieurs fenêtres sur l'écran

- La fenêtre de commandes (Command Window),
- La fenêtre de l'historique des commandes (Command History),
- La fenêtre du répertoire courant (celui de travail) (Current Directory).
- La fenêtre des variables définies (Workspace).

Une fois MATLAB lancé, nous sommes en présence de l'invite (***«prompt»***) de MATLAB:
>>
MATLAB est prêt à recevoir nos commandes:
>> a=1 % tout ce qui vient après le symbole % est un commentaire
a =
1

Par défaut, Matlab utilise une précision de 4 chiffres après la virgule pour l'affichage de résultats. L'utilisateur peut changer la précision des nombres en choisissant parmi les options suivantes :

Format	**Résultat**	**Exemple**
format short	4 chiffres après la virgule (par défaut)	1.1234
format long	14 chiffres après la virgule	1.12345678910111
format short e	4 chiffres après virgule + exposant	1234e+0011.
format short g	5 chiffres en tout avec ou sans exposant	12.126
format long e	15 chiffres après virgule + exposant	1.233333333333333e+043
format long g	15 chiffres après virgule au total, avec ou sans exposant	1.23333333333333e+043
format bank	"dollars et sous" format	9.75
format hex	affiche les bits en format hexadécimal	4028b0fcd32f707a
format +	seulement les signes sont affichés	+

Si vous voulez quitter Matlab mais que vous aimeriez reprendre la session de travail là où vous l'avez interrompue, il est possible de sauvegarder les variables en tout ou en partie à l'aide de la commande ***save***.

- ***save*** sauvegarde toutes les variables dans le fichier Matlab .mat
- ***save nom-de-fichier*** sauvegarde toutes les variables dans le fichier nom-de fichier .mat
- ***save nom-de-fichier var1 var2 var3*** sauvegarde les variables var1, var2 et var3 dans le fichier ***nom-de-fichier.mat.***

La commande ***save*** crée un fichier binaire par défaut. . Ceci est très utile si les résultats doivent être importés dans un autre logiciel . Consultez l'aide pour plus de détails.

Pour récupérer le contenu de nom-de-fichier, il s'agit tout de taper ***load nom-de fichier***.

Si vous désirez effacer toutes les variables, tapez la commande ***clear***.

I-3 Les variables

Avant de manipuler une variable (matrice, vecteur ou scalaire), il faut évidemment la créer. Il existe plusieurs façons de le faire : en tapant son contenu au clavier, en modifiant une variable existante, en effectuant des calculs sur une variable existante, etc.

a-Forme générale

Notons que la structure générale pour la saisie de données est :

nom-de-la-variable = contenu-de-la-variable

- Si l'usager omet de spécifier le nom d'une variable, MATLAB assigne automatiquement le contenu à la variable ***ans***.
- ***contenu-de-la-variable*** peut être un nombre, une constante prédéfinie, -une expression mathématique sur un nombre, une variable définie auparavant ou encore une chaîne de caractères.

b-Chaînes de caractères

Une chaîne de caractères est délimitée par des apostrophes. Par exemple :

s='votre nom' assigne la chaîne de caractères « votre nom » à la variable s. Puisque les variables sont de dimension variable, il est possible de rallonger une chaîne. Ainsi :

s=['Quel est', s, '? '] génère la chaîne « Quel est votre nom? »

On peut afficher un message, une valeur à l'écran avec ***disp***:

disp('Ceci est un test')	Afficher "Ceci est un test" sur l'écran
x = input('Valeur de x = ')	Afficher sur l'écran "Valeur de x = " et attendre qu'un nombre soit tapé sur le clavier

I-4- Les Vecteurs

Nous verrons quatre méthodes pour créer un vecteur :

- En tapant chaque valeur : Dans ce cas, le vecteur est délimité par des crochets. Chaque élément du vecteur est séparé par une virgule ou au moins un espace. Tous les éléments doivent être inscrits sur une seule ligne, sinon MATLAB essaiera de générer une matrice (voir section I.3). Voici deux exemples :

```
B=[1 2 3 4 5]
B=[1,2,3,4,5]
```

- En utilisant l'opérateur « : »: La forme générale d'utilisation du symbole « : » est :

variable = valeur initiale : incrément : valeur finale

- ***incrément*** est optionnel et sa valeur par défaut est égale à 1.
- les bornes et l'incrément peuvent être des nombres fractionnaires et même négatifs.

Voici quelques exemples :

```
>> x=3:7
x =
   3   4   5   6   7
>> y=0:pi/4:pi
y =
        0   0.7854   1.5708   2.3562   3.1416
>> z=6:-1:1
z =
   6   5   4   3   2   1
```

- En utilisant la commande linspace : La forme générale pour la commande linspace est la suivante :

linspace(valeur initiale, valeur finale, nombre des elements)

Exemple:

```
>>k=linspace(-pi, pi,4)
k =
  -3.1416  -1.0472   1.0472   3.1416
```

- à partir d'une matrice : L'application de l'opérateur « : » à une matrice a pour effet de transformer la matrice en vecteur en plaçant tous les éléments un à la suite des autres, par ordre de numéro d'élément

```
>> A=[1 2; 3 4; 5 6]
b=A(:)
A =
   1   2
   3   4
   5   6
b =
   1
   3
   5
   2
   4
   6
```

I-5 Les Matrices

Il existe plusieurs options pour générer une matrice. Il faut cependant savoir qu'elle est remplie ligne par ligne et qu'elle est délimitée par des crochets [ ].

- sur une seule ligne : Tous les éléments sont écrits sur une seule ligne de commande.

Chaque ligne est délimitée par des points virgules et les éléments d'une même ligne doivent être séparés par une virgule ou au moins un espace.

```
>> A=[1 2 3 ; 4 5 6 ; 7 8 9 ]
A =
   1   2   3
   4   5   6
   7   8   9
```

- sur plusieurs lignes : Chaque ligne de la matrice est écrite sur une ligne de commande séparée

```
>> A=[1 2 3
4 5 6
7 8 9]
A =
   1   2   3
   4   5   6
   7   8   9
```

L'élément de la 2eme ligne et la 3eme colonne s'écrit :

A(2,3)

On peut également extraire une sous matrice, ligne ou colonne d'une autre matrice

Exemples :

```
>> B=A(2:3,1:2)        % Extraire d'une matrice
B =
   4   5
   7   8
B=A(2,:)               % Extraire d'une ligne

B =
   4   5   6
>> B=A(:,3)            % Extraire d'une colonne
```

B =

3
6
9

On peut faire référence à des rangées ou colonnes sans que celles-ci soient placées côte à côte. Pour ce faire on place les rangées visées entre crochets.

Exemple :

A([2 4], [1 3 5]) fait référence aux éléments contenus dans les colonnes 1, 3 et 5 des rangées 2 et 4.

>> A=[1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16; 17 18 19 20]

A =

1	*2*	*3*	*4*
5	*6*	*7*	*8*
9	*10*	*11*	*12*
13	*14*	*15*	*16*
17	*18*	*19*	*20*

>> B=A([1 5],[1 2 4])

B =

1	*2*	*4*
17	*18*	*20*

I-5-a Opérations sur les matrices

- transposition : La transposition est symbolisée par l'apostrophe « ' ».

A =

1	*2*	*3*
4	*5*	*6*
7	*8*	*9*

>> A'

ans =

1	*4*	*7*
2	*5*	*8*
3	*6*	*9*

- **Multiplication** : Le symbole * fait référence à la multiplication matricielle. Avant de multiplier deux matrices, il faut auparavant s'assurer que leurs dimensions sont compatibles : si A*B, le nombre de colonnes de A doit être égal au nombre de rangées de B.

Exemples :

· A=[2 1 -1; 3 1 0] et B=[4 6; 0 2; 2 3]

A*B donne [6 11; 12 20]

· X=4*A donne [8 4 -4; 12 4 0]

Voici quelques remarques importantes concernant la multiplication de matrices :

· la multiplication de deux matrices n'est pas commutative. Ainsi A*B ¹ B*A.

· un vecteur peut être considéré comme une matrice à une rangée.

. la multiplication d'une matrice par un scalaire est équivalente à la multiplication de chacun des éléments par ce scalaire. Cette opération est commutative et équivalente à la multiplication sur les éléments « .* ». Ainsi 4*A=A*4=4.*A=A.*4

- **Division** : Il existe deux symboles de division de matrice dans MATLAB : la division à droite « / » et la division à gauche « \ ». Ils correspondent aux deux situations suivantes :

· X=A\B est la solution de A*X=B par élimination de Gauss. L'opération A\B peut aussi être considéré équivalent à inv(A)*B.

· X=B/A est la solution de X*A=B. Dans ce cas-ci, l'opération est équivalente à B*inv(A).

Si on veut multiplier ou diviser les matrices élément par élément on ajoute un « . » avant l'operateur

A.*B
A./B
A.\B
Exemple :

```
>> A=[2 1 -1; 3 1 0],B=[2 4 6; 0 2 3]
A =
   2   1  -1
   3   1   0
B =
   2   4   6
   0   2   3
>> C=A.*B
C =
   4   4  -6
   0   2   0
>>D=A./B
D =
   1.0000   0.2500  -0.1667
      Inf   0.5000        0
```

I-5-b Matrices particulières

Les fonctions ones, zeros permettent de construire des matrices remplies de 1 et de 0, respectivement. On a également la possibilité de construire une matrice identité (eye), diagonal (diag), une matrice dont les composantes sont aléatoires (rand),

I-5-c Quelques exemples

Exemple1

Evaluer l'expression suivante pour x=-1.2 **:**

$$\frac{\sqrt{\sqrt{|x|}+1}(\sin(\exp(x^3)+1))}{\arctan(x^2)+(\ln\left(\sqrt{|x|}+1\right))^{\frac{3}{2}}}$$

Rep :

```
>> x=-1.2 ;
>> y= (sqrt(sqrt(abs(x))+1)*(sin(exp(x^3)+1)))/(atan(x^2)+(log(sqrt(abs(x))+1))^(3/2))

y =
   0.8357
```

Exemple 2 :

Déclarer le vecteur suivant par deux méthodes

v =(100.5 90.5 … 10.5 0.5)

Rep :

```
>> v=100.5:-10:0.5
>>k=linspace(100.5,0.5,11)
```

Exemple3 :

Construisez une matrice A = (a_{ij}), de genre 6 x 6, définie par : $A = I + \frac{u^t u}{4}$

où

I : matrice identité

u = (1 2 3 4 5 6)

u^t = vecteur transposé de u

b) Ajoutez 2 à l'élément a_{23} et multipliez la deuxième colonne de A par ln(2) ; on appellera B la nouvelle matrice ainsi obtenue.

c) Calculez la matrice inverse de la matrice B et vérifiez que le produit B^{-1} *B donne l'identité.

Rep

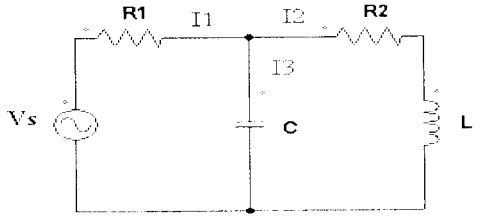
```
u=1:6;
A=eye(6)+u'u/4
B=A;
B(2,3)=B(2,3)+2;
B(:,2)=B(:,2)*log(2);
B
>> C=inv(B)
C*B
```

Exercice:

Soi le circuit de la figure suivante, on souhaite calculer les valeurs efficaces et phases des courants on donne :

Vs= 220 V, f = 50 Hz , R1 = 10 Ω

R2 = 3 Ω, C = 10 µF, L = 100 mH



Solution :

Pour résoudre ce circuit, il faut bien entendu commencer par écrire les équations qui le décrivent. Considérant les courants I_1, I_2, I_3 circulant dans les 3 branches du circuit, celui-ci est complètement décrit par les équations suivantes :

$$V_s = R_1 I_1 + \frac{1}{jC\omega} I_2$$

$$0 = -\frac{1}{jC\omega} I_2 + (R_2 + L\omega) I_3$$

$$0 = I_1 - I_2 - I_3$$

Equations que l'on peut récrire sous forme matricielle :

$$\begin{bmatrix} R_1 & \frac{1}{jC\omega} & 0 \\ 0 & -\frac{1}{jC\omega} & (R_2 + L\omega) \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} V_s \\ 0 \\ 0 \end{bmatrix}$$

La solution s'obtient en multipliant à gauche les 2 membres de l''equation par l'inverse de la matrice décrivant le circuit

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} R_1 & \frac{1}{jC\omega} & 0 \\ 0 & -\frac{1}{jC\omega} & (R_2 + L\omega) \\ 1 & -1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} V_s \\ 0 \\ 0 \end{bmatrix}$$

Le calcul des courants avec Matlab se fait comme suit :

```
% Calcul d'un circuit électrique
clear all ; close all ; format compact ;
% donnees
f = 50 ; w = 2*pi*f ; Vs = 220 ; % Vs = valeur efficace
R1 = 10 ; R2 = 3 ; L = 100e-3 ; C = 10e-6 ;
% rappel des équations du circuit
% R1 I1 + 1/jwC I2 + 0 I3 = Vs
% - 1/jwC I2 + (R2 + jwL) I3 = 0
% I1 - I2 - I3 = 0
% description matricielle du circuit
Z = [ R1 +1/(j*w*C) 0
0 -1/(j*w*C) R2+j*w*L
1 -1 -1] ;
U = [Vs ; 0 ; 0] ; % vecteur colonne
% resolution du circuit
I = inv(Z) * U ;
% affichage des valeurs
I
Ieff = abs(I)
PhaseI = angle(I) * 180 / pi
% affichage sous forme matricielle
Courants = [I Ieff PhaseI]
```

I-6 Manipulation de fonctions polynomiales dans MATLA

Soit le polynôme suivant :

$P(x)=a_n x^n+a_{n-1} x_{n-1}+\ldots+a_2 x^2+a_1 x+a_0$

où n est degré du polynôme et a_i (i=0,1,2,6....n) sont les coefficients du polynôme.

Dans MATLAB, un polynôme est représenté par un vecteur contenant les coefficients dans un *ordre décroissant*.

- Pour trouver ces racines, on doit exécuter la fonction '*roots*'.
- Si on connaît les racines on peu trouver le polynôme par la fonction poly
- Pour évaluer le polynôme p(x) en un point donné, on doit utiliser la fonction '*polyval*'.
- La fonction ***'polyfit'*** retourne le polynôme P de degré n permettant d'approcher la courbe au sens des moindres carrés.

Exemple 1

Soi le polynôme $p(x) = 2x^4+5x^2+x$

```
>>P=[2 0 5 1 0]      % déclaration de polynôme
>>y=polyval(p,2)     % évaluation de polynôme au point x=2
```

Exemple 2

```
x=[1.1 2.3 3.9 5.1];
>>y=[3.887 4.276 4.651 2.117];
>>a=polyfit(x,y,length(x)-1)
a =
-0.2015 1.4385 -2.7477 5.4370
```

Ainsi, le polynôme d'interpolation de y (d'ordre $length(x)-1=3$) est :

$P(x)= -0.2015x^3 + 1.4385x^2 -2.7477x + 5.4370$

I-7 Instructions de contrôle

Les instructions de contrôle sous Matlab sont très proches de celles existant dans d'autres langages de programmation.

I-7-a Boucle FOR :

Une première possibilité pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle pour les valeurs d'un indice, incrémenté à chaque itération, variant entre deux bornes données. Ce processus est mis en œuvre par la « boucle for ».

Syntaxe :

```
for indice = borne_inf : borne_sup

séquence d'instructions

end
```

où

- ✓ ***indice*** est une variable appelée l'indice de la boucle ;
- ✓ ***borne_inf*** et ***borne_sup*** sont deux constantes réelles (appelées paramètres de la boucle) ;
- ✓ ***séquence d'instructions*** est le traitement à effectuer pour les valeurs d'indices variant entre borne_inf et borne_sup avec un incrément de 1. On parle du corps de la boucle.

Interprétation :

Si ***borne_inf*** est plus petit ou égal à ***borne_sup***, le traitement séquence d'instructions est exécuté ***(borne_sup - borne_inf + 1)*** fois, pour les valeurs de la variable ***indice*** égales à ***borne_inf, borne_inf + 1, ..., borne_sup-1, borne_sup***.

Si ***borne_inf*** est strictement plus grand que ***borne_sup***, on passe à l'instruction qui suit immédiatement l'instruction de fin de boucle (***end***).

Remarque :

L'indice de boucle ne prend pas nécessairement des valeurs entières. D'autre part il n'est pas nécessaire que l'indice de la boucle apparaisse dans le corps de la boucle : par contre il est interdit de modifier sa valeur s'il apparaît. Il est possible d'imbriquer des boucles mais elles ne doivent pas se recouvrir. On peut utiliser un incrément (pas) autre que 1 (valeur par défaut).

La syntaxe est alors

borne_inf : pas : borne_sup.

Le pas peut être négatif. Attention à bien gérer la borne supérieure ! Voici un exemple venant illustrer les possibilités de variations de l'indice de la boucle

Exemples :

Voici un exemple d'une boucle for pour générer un vecteur de dimension n :

```
x = [ ];
for i = 1:n
x = [x,i];
end
% génèrent le même vecteur mais dans l'ordre inverse.
x = [ ];
for i = n:-1:1
x = [x,i];
end
% À noter que x = [ ] définit une matrice VIDE.
% génèrent une matrice de Hilbert de m _ n éléments.
for i = 1:m
        for j = 1:n
                H(i, j) = 1/(i+j-1);
        end
end
H
% Voici un exemple d'utilisation d'une boucle pour calculer n!
>> n = 4;
>> nfac = 1;
>> for k = 1:n
nfac = nfac*k;
end
>> nfac
nfac =24
```

I-7-b Boucle WHILE : tant que . . . faire

Une seconde possibilité pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle tant qu'une condition reste vérifiée. On arrête de boucler dès que cette condition n'est plus satisfaite. Ce processus est mis en œuvre par la boucle ***while***

Syntaxe :

```
while expression logique
séquence d'instructions
end
```

où

- ✓ expression logique est une expression dont le résultat peut être vrai ou faux ;
- ✓ séquence d'instructions est le traitement à effectuer tant que expression logique est vraie.

Interprétation :

Tant que expression logique est vraie le traitement séquence d'instructions est exécuté sous forme d'une boucle. Lorsque expression logique devient faux, on passe à l'instruction qui suit immédiatement l'instruction de fin de boucle (end).

Remarque :

Expression logique est en général le résultat d'un test (par exemple i < Imax) ou le résultat d'une fonction logique (par exemple all(x)). Il est impératif que le traitement de la séquence d'instructions agisse sur le résultat de expression logique sans quoi on boucle indéfiniment.

Voici comment calculer n! avec une boucle while :

```
>> n = 4;
>> k = 1; nfac = 1;
```

```
>> while k <= n
nfac = nfac*k;
k = k+1;
end
>> nfac
nfac =
24
```

I-7-c L'instruction conditionnée IF

On a parfois besoin d'exécuter une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée au préalable. Différentes formes d'instruction conditionnée existent sous matlab.

L'instruction conditionnée la plus simple a la forme suivante :

Syntaxe :

```
if expression logique
séquence d'instructions
end
```

où

- ✓ expression logique est une expression dont le résultat peut être vrai ou faux ;
- ✓ séquence d'instructions est le traitement à effectuer si expression logique est vraie.

Interprétation :

La séquence d'instructions n'est exécutée que si le résultat de l'évaluation de l'expression logique est vraie (c'est-à-dire vaut 1). Dans le cas contraire on exécute l'instruction qui suit le mot clé end. Dans le cas où l'expression logique est vraie, après exécution de la séquence d'instructions on reprend le programme à l'instruction qui suit le mot clé end.

Remarque :

Contrairement à certains langages de programmation, il n'y a pas de mot clé « then » dans cette instruction conditionnée. Notez également que la marque de fin de bloc conditionné est le mot clé end et non pas « endif ».

Il existe une séquence conditionnée sous forme d'alternatives :

Syntaxe :

```
if expression logique
séquence d'instructions 1
else
séquence d'instructions 2
end
```

où

- ✓ expression logique est une expression dont le résultat peut être vrai ou faux ;
- ✓ séquence d'instructions 1 est la séquence d'instructions à exécuter dans le cas où expression logique est vraie et séquence d'instructions 2 est la séquence d'instructions

à exécuter dans le cas où expression logique est faux.

Interprétation :

Si expression logique est vraie la séquence d'instructions 1 est exécutée, sinon c'est la séquence d'instructions 2 qui est exécutée. Le déroulement du programme reprend ensuite à la première instruction suivant le mot clé end.

Il est bien entendu possible d'imbriquer des séquences d'instructions conditionnées (au sens où la séquence d'instruction conditionnée contient des séquences d'instructions conditionnée).

Pour une meilleure lisibilité, il est recommandé d'utiliser des indentations afin de mettre en évidence l'imbrication des séquences d'instructions conditionnées.

Il est possible d'effectuer un choix en cascade :

Syntaxe :

if expression logique 1

séquence d'instructions 1

elseif expression logique 2

séquence d'instructions 2

...

elseif expression logique N

séquence d'instructions N

else

séquence d'instructions par défaut

end

Interprétation :

Si **expression logique 1** est vraie **la séquence d'instructions 1** est exécutée et le programme reprend ensuite à la première instruction suivant le mot clé **end**, **sinon si expression logique** 2 est vraie la séquence d'instructions 2 est exécutée et le programme reprend ensuite à la première instruction suivant le mot clé end, etc. Si aucune des expressions logiques 1 à N n'est vraie alors séquence d'instructions par défaut est exécutée.

Remarque :

Attention à ne pas laisser d'espace entre else et if ; le mot clé est ***elseif***. On utilise fréquemment un choix en cascade lors d'initialisation de données.

Exemple :

```
if moy >= 16
mention = 'très bien'
elseif moy >= 13
mention = 'bien'
elseif moy >= 12
mention = 'Assez bien'
elseif moy >=10
mention = 'Passable'
else
mention = 'Ajourné(e)'
end
```

I-7-d Interruption d'une boucle de contrôle

Il est possible de provoquer une sortie prématurée d'une boucle de contrôle.

- L'instruction ***break*** permet de sortir d'une boucle ***for*** ou d'une boucle ***while***. L'exécution se poursuit alors séquentiellement à partir de l'instruction suivant le mot clé end fermant la boucle.

En cas de boucles imbriquées, on interrompt seulement l'exécution de la boucle intérieure contenant l'instruction ***break***.

- L'instruction ***return*** provoque un retour au programme appelant (ou au clavier). Les instructions suivant le ***return*** ne sont donc pas exécutées. L'instruction ***return*** est souvent utilisée conjointement avec une instruction conditionnée par exemple pour tester dans le corps d'une fonction si les paramètres d'entrée ont les valeurs attendues.

- L'instruction ***error*** permet d'arrêter un programme et d'afficher un message d'erreur. La syntaxe est ***error*** (' message d erreur ').
- L'instruction ***warning*** permet d'afficher un message de mise en garde sans suspendre l'exécution du programme.

La syntaxe est ***warning***(' message de mise en garde '). Il est possible d'indiquer à matlab de ne pas afficher les messages de mise en garde d'un programme en tapant ***warning off*** dans la fenêtre de commandes. On rétablit l'affichage en tapant **warning on.**

La commande ***pause*** permet d'interrompre l'exécution du programme. L'exécution normale reprend dès que l'utilisateur enfonce une touche du clavier. L'instruction ***pause(n)*** suspend l'exécution du programme pendant n secondes.

I-8 Les Fonctions :

Matlab utilise les fonctions. Une fonction dans Matlab peut être telle qu'on l'entend au sens mathématique. Par exemple, nous pouvons créer une fonction qui calcule x^n pour un x et un n donnés. La syntaxe est la suivante :

```
function [y]=puissancen(x,n)
y=x.^n;
```

La commande **function** annonce la création d'une fonction, entre crochets [] sont les variables de sortie séparées par des virgules, après le signe = est le nom que nous souhaitons donner à la fonction (puissancen), et entre parenthèses () sont les variables d'entrée de la fonction séparées par des virgules. Il est conseillé d'enregistrer ce fichier sous le même nom que celui donné à la fonction, soit ici sous le nom de fichier puissancen.m. Par défaut, les variables utilisées dans la fonction sont locales, i.e., elles ne sont pas connues du programme principal ou de la fenêtre de commande.

Par **exemple**, dans un nouveau fichier, on peut définir la fonction

```
function f = myfunc(x)
% f = myfunc(x)
% Calcule x - exp(x)
f = x - exp(x);
end
```

On l'enregistre sous le nom myfunc.m dans le répertoire de travail. Notons que cette fonction peut retourner un scalaire, un vecteur ou une matrice, selon la nature de x. On peut alors effectuer une représentation graphique de la fonction

```
>> x = linspace(-5, 5, 200);
>> y = myfunc(x);
>> plot(x,y)
```

Exemple2

Il peut y avoir une seule variable retournée comme dans la fonction myfunc de la section précédente (dans ce cas les crochets ne sont pas nécessaires).

Une fonction peut également retourner plusieurs variables

```
function [mi,ma,avg] = vectstat(x)   % mi,ma et avg variable de sortie, x variable d'entré.
% Calcule le min, max et la moyenne d'un vecteur
l = length(x);
mi = min(x);
ma = max(x);
avg = sum(x) / l;
end
```

I-9 Liste de quelques fonctions Matlab

Cette liste, qui se veut non exhaustive, regroupe les commandes et fonctions les plus usuelles. Elle illustre également la richesse des possibilités offertes. Pour plus de détails, il faut évidemment consulter l'aide en ligne offerte par Matlab et profiter des nombreux exemples qui y sont proposés.

Commandes et fonctions

Help	Aide
what	Liste des fichiers *.m présents dans le répertoire courant
typeé	Imprime un fichier
lookf	Recherche d'une entrée dans l'aide
which	Localise les fonctions et fichiers
demo	Lance la démonstration
path	Définition des chemins d'accès aux fichiers et fonctions
version	Affiche le numéro de version de Matlab
whatsnew	Affiche les fichiers README de la toolbox
info	Informations sur Matlab et Mathworks
why	Fournit une réponse aléatoire

Informations sur l'espace de travail

who	Affiche les variables courantes
whos	Affiche les variables courantes avec leurs dimensions
save	Sauve l'espace de travail sur disque
load	Restaure l'espace de travail à partir du disque
clear	Efface les variables et fonctions de la mémoire
close	Ferme la fenêtre courante
pack	Réorganise la mémoire
size	Renvoie la taille d'une matrice
length	Renvoie la longueur d'un vecteur
disp	Affiche une matrice de texte

Commandes système

cd	Change le répertoire courant
pwd	Affiche le répertoire courant
dir, ls	Liste les fichiers
delete	Suppression de fichiers
getenv	Renvoie la variable d'environnement
!	Appelle et exécute une commande système
diary	Sauvegarde le texte d'une session Matlab

Fenêtre de commandes

clc	Efface le contenu de la fenêtre de commandes
home	Place le curseur en haut de l'écran
format	Définit le format d'affichage
echo	Affiche les instructions exécutées par un script
quit, exit	Ferme Matlab
Matlabrc	Fichier de démarrage

Caractères spéciaux

[ ]	Définition de matrices ou vecteurs : enserre les arguments de sortie des fonctions
( )	Gère la priorité des opérations : enserre les arguments d'entrée des fonctions
.	Point décimal
..	Répertoire parent
...	Indique une ligne suite
,	Séparateur d'arguments ou d'instructions
;	Fin de lignes (matrices) ou suppression de l'affichage
%	Commentaires

:	Manipulation de sous matrices ou génération de vecteurs

Opérateurs logiques

<	Inférieur à
&	Et
>	Supérieur à
|	Ou
<=	Inférieur ou égal à
~	Non
>=	Supérieur ou égal à
xor	Ou exclusif
==	Egal à
~=	Différent de
=	Assignation

Variables prédéfinies ; durée et date

ans	Réponse à une expression sans assignation
eps	Précision de la virgule flottante
realmax	Plus grand nombre flottant
realmin	Plus petit nombre flottant positif
pi	π
i,j	$\sqrt{-1}$
inf	∞
NaN	Not a Number
Flops	Nombre d'opérations flottantes par seconde
nargin	Nombre d'arguments d'entrée d'une fonction
nargout	Nombre d'arguments de sortie d'une fonction
computer	Type du calculateur
date	Date courante
clock	Horloge
etime	Durée d'exécution
tic,toc	Affiche le début et la fin d'exécution
cputime	Temps CPU écoulé

Fonctions logiques

exist	Teste l'existence d'une variable ou d'une fonction
any	Vrai si un élément est vrai
all	Vrai si tous les éléments sont vrais
find	Cherche l'indice des éléments non nuls
isnan	Vrai si l'élément n'est pas un nombre
isinf	Vrai pour tout élément infini
finite	Vrai pour tout élément fini
isieee	Vrai si la représentation est au format IEEE
isempty	Vrai pour une matrice vide
issparse	Vrai pour une matrice creuse
isstr	Vrai pour une chaîne de caractères
strcmp	Comparaison de deux chaînes

Instructions de contrôle

if, else, elseif	Test conditionnel
for	Instruction de répétition avec compteur
while	Instruction de répétition avec test
end	Terminaison de if, for et while
break	Interrompt une boucle
return	Retour
error	Affiche un message et interrompt l'exécution

Instructions spécifiques

input	Indicateur d'attente d'entrée
keyboard	Considère le clavier comme un fichier script
menu	Génère un menu de choix pour l'utilisateur
pause	Attente
function	Définition de fonction
eval	Exécute une chaîne de caractère
feval	Exécute une fonction définie dans une chaîne
global	Définit les variables comme globales
nargchk	Valide le nombre d'arguments d'entrée

Fonctions mathématiques

Fonctions élémentaires

abs	Valeur absolue ou module d'une grandeur complexe
angle	Argument d'une grandeur complexe
sqrt	Racine carrée
real	Partie réelle
imag	Partie imaginaire
conj	Complexe conjugué
round	Arrondi à l'entier le plus proche
floor	Arrondi vers le bas
ceil	Arrondi vers le haut
sign	Signe de
rem	Reste de la division
exp	Exponentielle
log	Log népérien
log10	Log décimal
log2	Log base 2
pow2	Calcule 2 à la puissance y

Fonctions trigonométriques

sin, asin, sinh, asinh, cos, acos, cosh, acosh, tan, atan, tanh, atanh, cot, acot, coth, acoth, sec, asec, sech, asech, csc, acsc, csch, acsch

Matrices et algèbre linéaire

1 Opérateurs sur les matrices

+, -, *, ^	Addition, Soustraction, Multiplication, Puissance
/, \	Division à droite, Division à gauche
'	Transposition conjuguée

2 Opérateurs sur les composantes matricielles

+, -, .*, .^	Addition, Soustraction, Multiplication, Puissance
./, .\	Division à droite, Division à gauche
.'	Transposition

3 Manipulation des matrices

diag	Création ou extraction de la diagonale
rot90	Rotation de 90 degrés
fliplr	Retournement gauche-droit
flipud	Retournement haut-bas
reshape	Redimensionnement
tril	Partie triangulaire inférieure
triu	Partie triangulaire supérieure
.'	Transposition
:	Conversion matrice -> vecteur

4 Matrices prédéfinies

zeros	Matrice de 0
ones	Matrice de 1
eye	Matrice identité

diag	Matrice diagonale
magic	Carré magique
compan	Matrice compagnon
linspace	Vecteurs à composantes linéairement espacées
logspace	Vecteurs à composantes logarithmiquement espacées
meshgrid	Grille pour les graphiques 3D
rand	Nombres aléatoires à répartition uniforme
randn	Nombres aléatoires à répartition normale

5 Opérations sur les matrices

poly	Polynôme caractéristique
det	Déterminant
trace	Trace
inv	Inversion
svd	Décomposition en valeurs singulières

Fonctions graphiques

1 Graphiques 2D

plot	Dessine le graphe d'une fonction
loglog	Graphe en échelle log-log
semilogx	Graphe en échelle semi-log (abscisse)
semilogy	Graphe en échelle semi-log (ordonnée)
polar	Graphe en coordonnées polaires
bar	Graphe en barres
stairs	Graphe en marches d'escalier
stem	Graphe de raies
errorbar	Graphe avec barres d'erreur
hist	Histogramme
rose	Histogramme en coordonnées polaires
feather	Représentation de vecteurs sur un axe linéaire

Annotation de graphiques

title	Titre du graphique
xlabel	Légende pour l'abscisse
ylabel	Légende pour l'ordonnée
zlabel	Légende pour la cote
grid	Dessin d'une grille
text	Texte
gtext	Placement de texte avec la souris
ginput	Entrée graphique par la souris

Contrôle des fenêtres graphiques

figure	Ouvre une fenêtre graphique
clf	Efface la figure courante
close	Ferme la figure courante
hold	Gère la surimpression
ishold	Etat de la surimpression
subplot	Sous fenêtres graphiques
axes	Axes en position arbitraire
gca	Retourne le numéro des axes courants
axis	Contrôle l'apparence et l'échelle des axes
whitebg	Dessine sur fond blanc

2- Objets 3D

sphere	Génération de sphères
cylinder	Génération de cylindres
peaks	Démonstration

Graphiques tridimensionnels

mesh	Surface maillée
meshc	Combinaison mesh + dessin des équi-niveaux