ENISE
5ième Année

Aide mémoire Matlab

E.FAVIER

1. MATLAB®

MATLAB® signifie **Mat**rix **Lab**oratory.
La structure de données principale utilisée par Matlab est la matrice mono-dimensionnel ou multidimensionnelle.

MATLAB® constitue un excellent outil de prototypage et de test dans de très nombreuses activités scientifiques basées autour du traitement du signal ou des données.

MATLAB® intègre un langage de haut niveau dédié au calcul scientifique et technique. Ce langage permet le calcul, la visualisation, et la programmation dans un environnement facile à utiliser.
Les problèmes et les solutions sont exprimés en notation mathématique familière.

MATLAB® s'organise autour de 6 grands pôles :

1.1. L'environnement de développement :

C'est un ensemble d'outils et d' équipements qui permettent d'utiliser des fonctions et les fichiers de MATLAB. Plusieurs de ces outils possèdent une interface graphique. Cet environnement de développement comprend plusieurs fenêtre :

le bureau de travail de MATLAB : ***"Desktop"***
la fenêtre de commande : ***"Command Window"***
La fenêtre Historique : ***"Command History"*** permettant d'avoir accès à toutes les dernières actions réalisés
L'espace de travail : ***"Workspace"*** contenant toutes les données contenues en mémoire et pouvant être si leur taille n'est pas trop grande être consultées directement
L'aide de Matlab : très élaborée

1.2. La librairie des fonctions mathématiques :

C'est une vaste collection d'outils informatiques s'étendant des fonctions élémentaires comme la somme, la multiplication, les fonctions trigonométriques, l'arithmétique complexe, à des fonctions d'algèbre linéaire (matrice inverse, valeurs propres …), les fonctions de bessel, la transformation de Fourier ….

1.3. Le langage MATLAB®

C'est un langage de haut niveau utilisant la structure de données matrice comme base et possédant :

Les structures de contrôle de tous les langages de haut niveau
La possibilités d'écrire des fonctions
Les entrées sorties habituelles
La programmation Orientée Objet est également possible.

Ce langage permet aussi bien de développer des petites applications de façon très rapide que de complexes programmes d'application.

1.4. L'environnement graphique de développement :

C'est le système graphique de MATLAB.
Il inclut des commandes à niveau élevé pour la visualisation de données, le traitement d'image, l'animation, et les graphiques bidimensionnels et tridimensionnels de présentation.
Il inclut également des commandes de bas niveau qui permettent d'adapter entièrement l'aspect des graphiques et autorisent la création d' interfaces utilisateur graphiques complètes pour les applications de MATLAB.

1.5. L'interface de développement Application Program Interface(API)

C'est une bibliothèque qui permet d'écrire les programmes en Langage C et Fortran qui communiquent avec MATLAB. Le logiciel Matlab intègre un compilateur C (lcc) mais permet à l'utilisateur d'utiliser le compilateur C de son choix. Il n'y a pas de compilateur ForTran intégré.

1.6. Les boites à outils (Toolbox)

Il s'agit d'ensemble de fonctions organisées autour de différents thèmes :

- Logique floue
- Statistique
- Réseaux de neurones
- Calcul formel
- Traitement d'images
- Production de cartes
- Ondelettes
- Le calcul formel

2. Le bureau de MATLAB®

Les deux figures ci-dessous permettent de visualiser l'environnement de Matlab

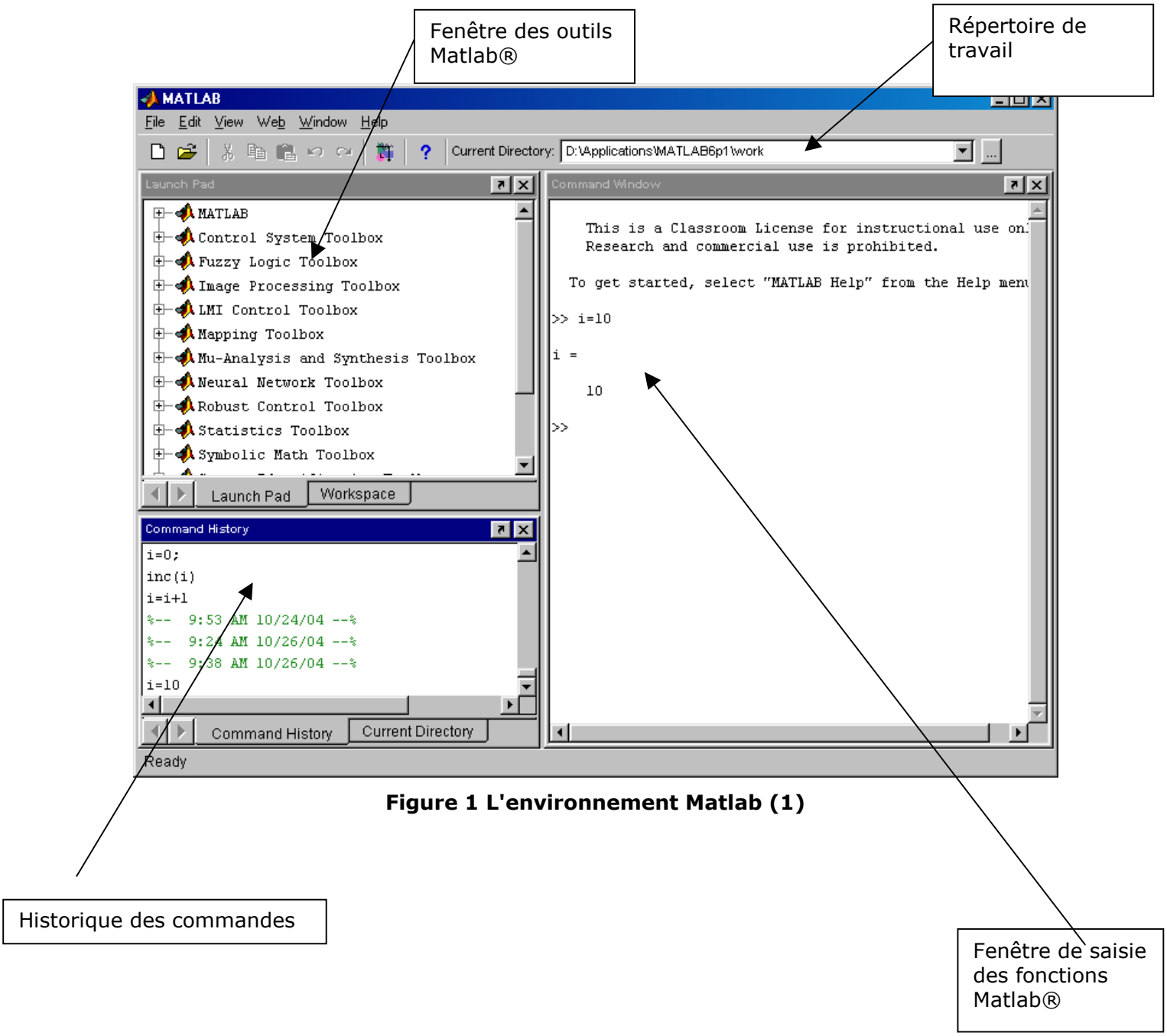


Figure 1 L'environnement Matlab (1)

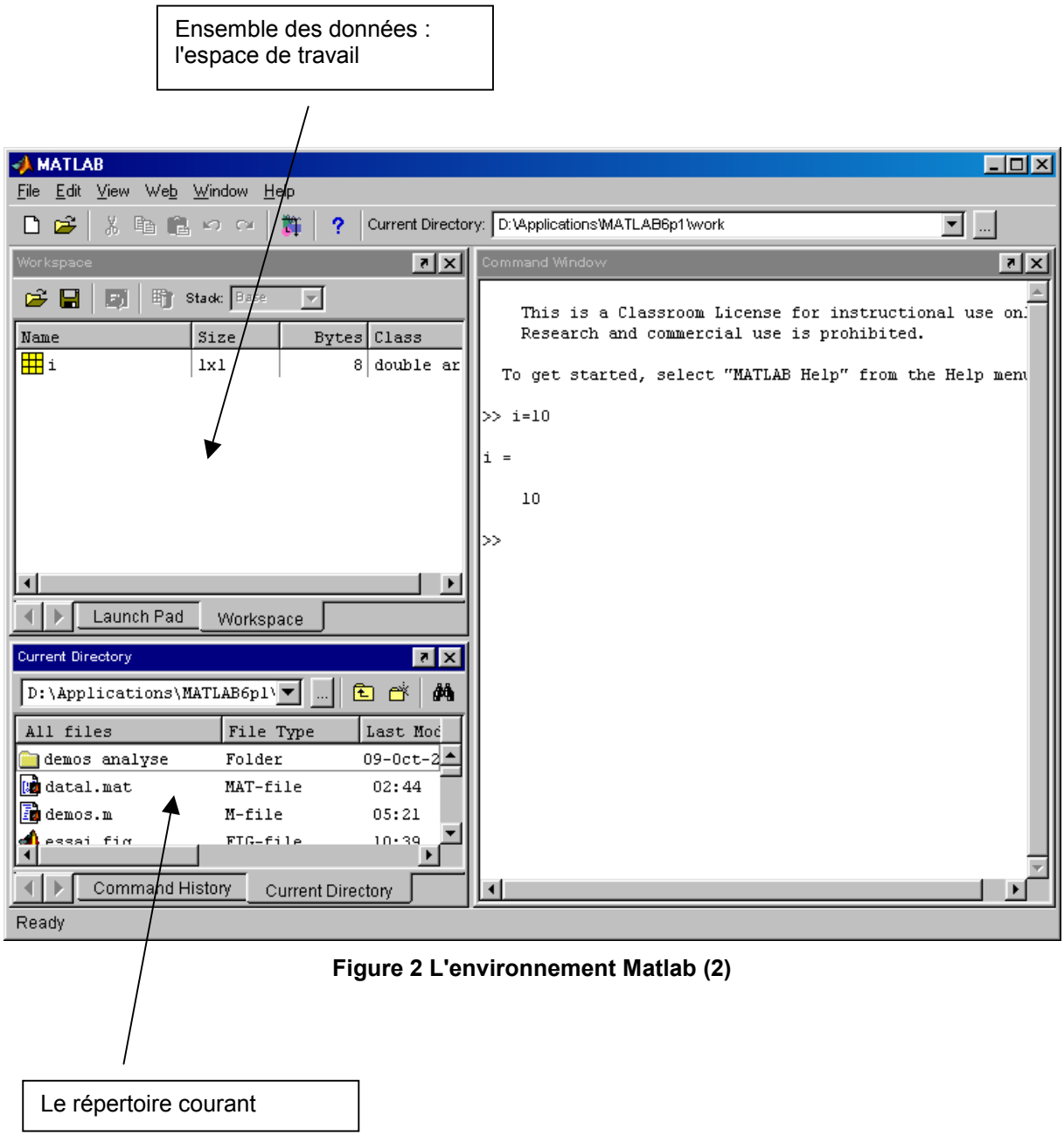


Figure 2 L'environnement Matlab (2)

Cet environnement est paramétrable en fonction de la volonté du l'utilisateur.

3. Le langage de MATLAB®

3.1. Comment programmer

Il y a deux façons pour écrire des fonctions Matlab :

- soit directement dans la fenêtre de commandes,
- soit en utilisant l'éditeur de développement de Matlab (cf. **Figure 3**) et en sauvant les programmes dans des fichiers texte avec l'extension ".m"

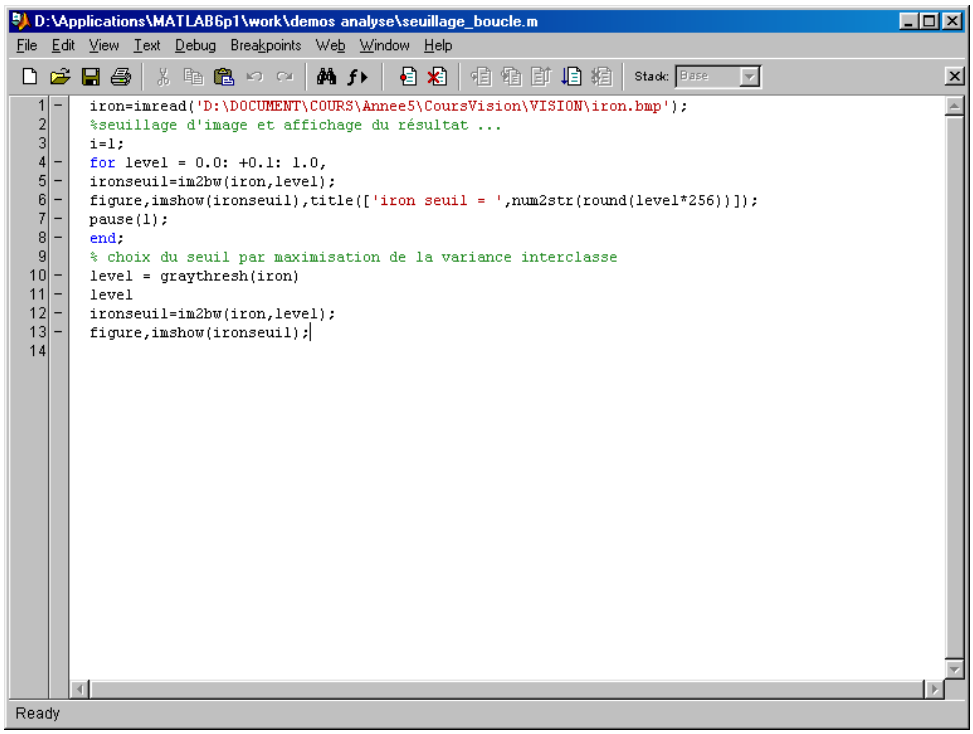


Figure 3 L'éditeur de programmation Matlab

Les programmes sauvés dans les fichiers matlab (*.m) sont alors directement utilisables comme des fonctions Matlab à partir de la fenêtre de commande. Pour cela le répertoire dans lequel se trouve le fichier doit se trouver dans les chemins reconnus par celui-ci (→ File → Set Path…). Il est toutefois conseillé de toujours sauver ces programmes dans un même répertoire de travail.

3.2. Le langage Matlab®

Le langage de MATLAB® est sensible à la casse (distinction entre les majuscules et les minuscules).

Pour écrire une ligne de commentaire il suffit de la faire précéder du caractère : %

3.2.1. La déclaration des variables

La déclaration et le typage des variables n'existent pas dans le langage Matlab. Mais bien entendu, avant d'assigner une variable à une autre il faut s'assurer de la compatibilité des types. Il est toutefois possible de créer des variables globales (global x, par exemple).

N'importe quelle opération qui assigne une valeur à une variable permet la création de la variable si nécessaire, ou recouvre sa valeur courante si elle existe déjà.
Les noms de variables de MATLAB se composent d'une lettre suivie de tout nombre de lettres, chiffres, et soulignages.

MATLAB reconnaît seulement les 31 premiers caractères des noms de variables.

Par exemple :

```
>> i=10
```

Permet de déclarer une variable de type entier et de l' initialiser à la valeur 10 (elle apparaît alors dans le Workspace cf. ci-dessous)

```
>>x=2.3
```

Permet de déclarer une variable de type flottant et de l' initialiser à la valeur 2.3 (elle apparaît alors dans le Workspace cf. ci-dessous)

```
>>m=[2 3 1 ; 4 5 2 ; 3 2 1]
```

Permet de déclarer une matrice 3*3 d'entiers et de l' initialiser (elle apparaît alors dans le Workspace cf. ci-dessous)

```
>>u=1+2.j
```

Permet de déclarer une valeur complexe (elle apparaît alors dans le Workspace cf. ci-dessous)

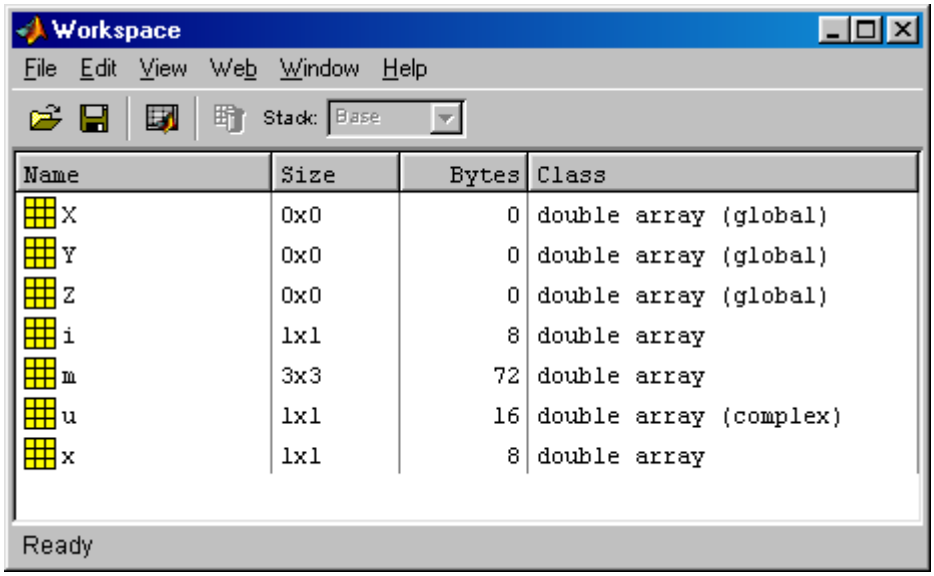


Figure 4 Workspace

Il est possible d'avoir accès aux données du workspace de façon immédiate par l'intermédiaire de celui-ci (cf. **Figure 5**) et de les modifier.

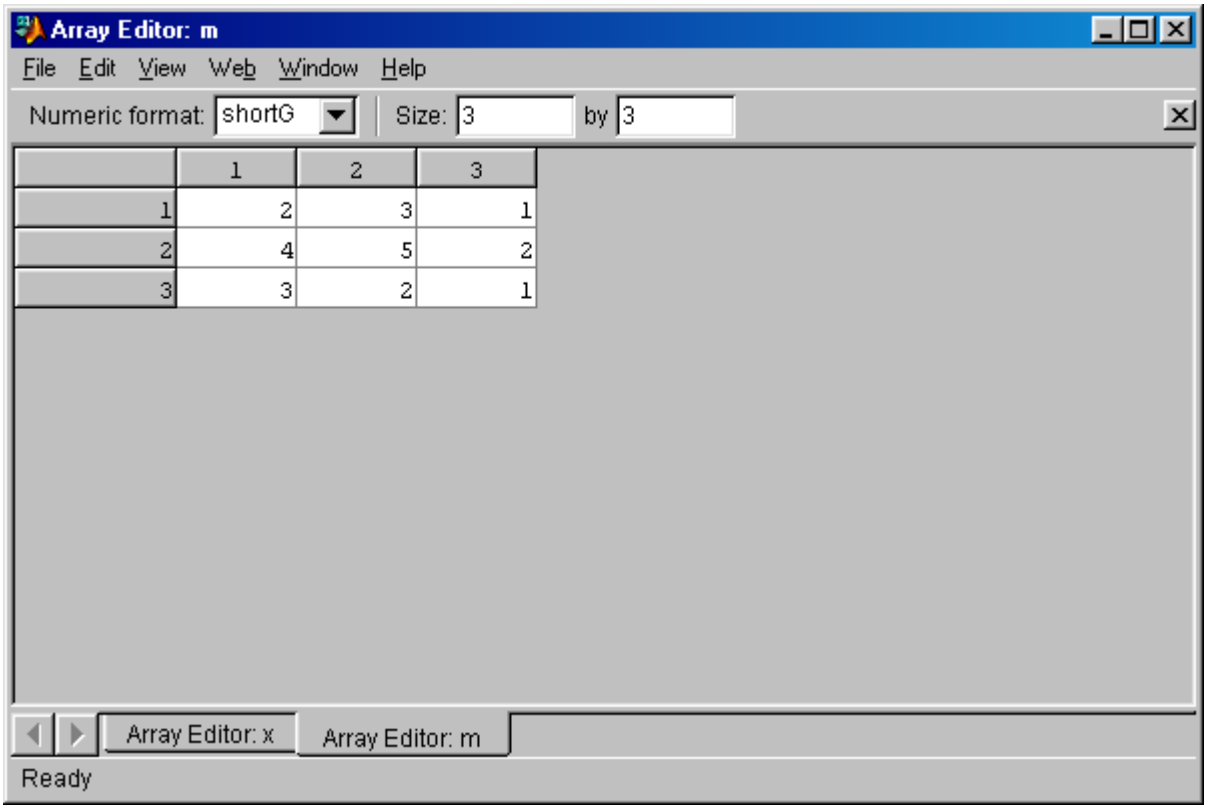


Figure 5 Accès aux données du Workspace

Les types de données de bases sont les suivants :

Six types entiers

int8 : entiers signés codés sur un octet (-128 à +127)
int16 : entiers signés codés sur deux octets (-32768 à +32767)
int32 : entiers signés codés sur quatre octets (-2147483648 à 2147483647)
uint8 : entiers non signés codés sur un octet (0 à +255)
uint16 : entiers non signés codés sur deux octets (0 à +65535)
uint32 : entiers non signés codés sur quatre octets (0 à 4294967295)

Deux types flottants

single : flottants sur 4 octets (non utilisable dans des opérations donc seulement utilisé pour le stockage)
double : flottants sur 8 octets
Matlab permet également l'utilisation de ***complexe*** codée en fait par un double

Caractères

char : codé sur un octet

3.2.2. Les matrices dans MATLAB®

Dans MATLAB®, une matrice est un tableau de nombres de taille nxm, avec n et m des entiers naturels. Les matrices de taille 1x1 sont les grandeurs scalaires, et les matrices 1xn ou nx1 sont des vecteurs.

Saisie des matrices :

Pour saisir une matrice, plusieurs possibilités s'offrent aux programmeurs :

- Écrire une liste explicite d'éléments :
  >> [1 2 3;4 5 6;7 8 9]
- Charger les matrices à partir des fichiers externes de données.
  *>>C = fix(10*rand(3,2))*
- Produire des matrices en utilisant des fonctions intégrées.
- Créer les matrices avec ses propres fonctions dans les fichiers Matlab.

Manipulation des matrices :

Si la matrice A est créer de la manière suivante :

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

Alors, l'accès aux éléments des matrices peut se faire de façon individuelle : l'élément dans la rangée i et la colonne j de A est noté par A(i, j). Dans l'exemple précédent A(1,4) est égale à 13.

L'opérateur colonne :

L'opérateur colonne ***":"*** permet la création de vecteur de toute pièce ou d'extraire un vecteur d'une matrice existante

>> 1:10	permet d'obtenir un vecteur contenant les entiers de 1 à 10
>> 200:-4:180	permet d'obtenir : 200 196 192 188 184 180
>>A(1:4,4)	permet d'obtenir la 4^{ième} colonne de la matrice A

L'algèbre linéaire :
MATLAB® implémente de très nombreux outils d'algèbre linéaire (cf. 4.2)

3.2.3. Les structures dans MATLAB®

Il est possible de définir des structures (enregistrement contenant plusieurs champs -fields-). L'accès aux champs se fait par l'intermédiaire de l'opérateur.

>> personne.age =12	déclaration du premier champ
>> personne.sexe = 'M'	déclaration du second champ
>>personne	renvoie les deux champs

Les structures peuvent être bien entendu combinées avec des tableaux ….
De nombreuses fonction permettent de manipuler les structures (struct, setfield, getfield, rmfield …)

3.2.4. Les opérateurs arithmétiques

Les opérateurs de Matlab sont les suivants (dans l'ordre de priorité) :
+, **-**, **x** , **/** , **** , **^**, **'** et **()** pour inverser les ordres de priorité des opérations

**** : *"X = A\B"* est la solution de AX=B calculée par l'algorithme du pivot de Gauss dans le cas ou A est une matrice carrée.

^: pour l'élévation à la puissance

' : conjugué d'un complexe

3.2.5. L'affichage et la saisie de données

La fonction ***"disp"*** permet d'afficher du text ou une matrice, par exemple :

```
>> disp(' Hello World')
```

La fonction input permet d'afficher une chaîne de caractères et de récupérer une valeur saisie au clavier par l'opérateur, par exemple :

```
>> val = input('Donner une valeur')
```

3.2.6. Les opérateurs logiques

Les opérateurs logiques de bases sont implémentés de la façon suivante :

NON	~ ou not(.)
ET	& ou and(.,.)
OU inclusif	| ou or(.,.)
OU exclusif	xor(.,.)

3.2.7. Les opérateurs de relations

égalité	==
différence	~=
Inégalités strictes ,larges	< >,<=,>=

3.2.8. Les structures de contrôle

L'alternative

A une branche, la syntaxe est la suivante :

```
>>  if ( CB  )  inst
    end
```

A deux branches, la syntaxe est la suivante :

```
>>  if ( CB  )  instsv
    else instsf
    end
```

Voici un exemple concret d'utilisation :

```
>>      if (det(A)==0) disp('A est inversible')
        else disp('A est non inversible');
        end;
```

Le selon que

L'instruction **switch** permet l'éxécution de certaines instructions en fonction de la valeur d'une variable ou d'une expression. Sa forme de base est

```
>>switch expression (scalaire ou chaîne de caractères)
   case value1
      instruction1
   case value2
      instruction2
   .
   .
   otherwise
      instruction SN % Executes if expression does not
                     % does not match any case
  end
```

Le choix logique

Contrairement aux langages de programmation C et Pascal, Matlab implémente le choix logique (cf . cours d'algorithmie de première année). Cette mise en ouevre se fait de la façon suivante :

```
if (CB1)
   instruction1
elseif (CB2)
   instruction2
elseif  (CB3)
  instruction3
.
.
else
   instruction
end
```

La boucle TantQue

La boucle TantQue est implémentée de façon quasiment analogue au langage C :

```
while ( CB)
   instruction
end
```

La boucle Pour

Avec Matlab, la boucle Pour est implémentée de la façon suivante :

```
for cpt = debut:increment:fin
   instruction
end
```

Le pas de l'incrément est un paramètre du réglage de la boucle (comme dans le langage C).
Voilà deux exemples d'utilisation :

1 Un exemple tout simple, la table de multiplication par 2 :

```
for i = 0:10
disp(2*i);
end;
```

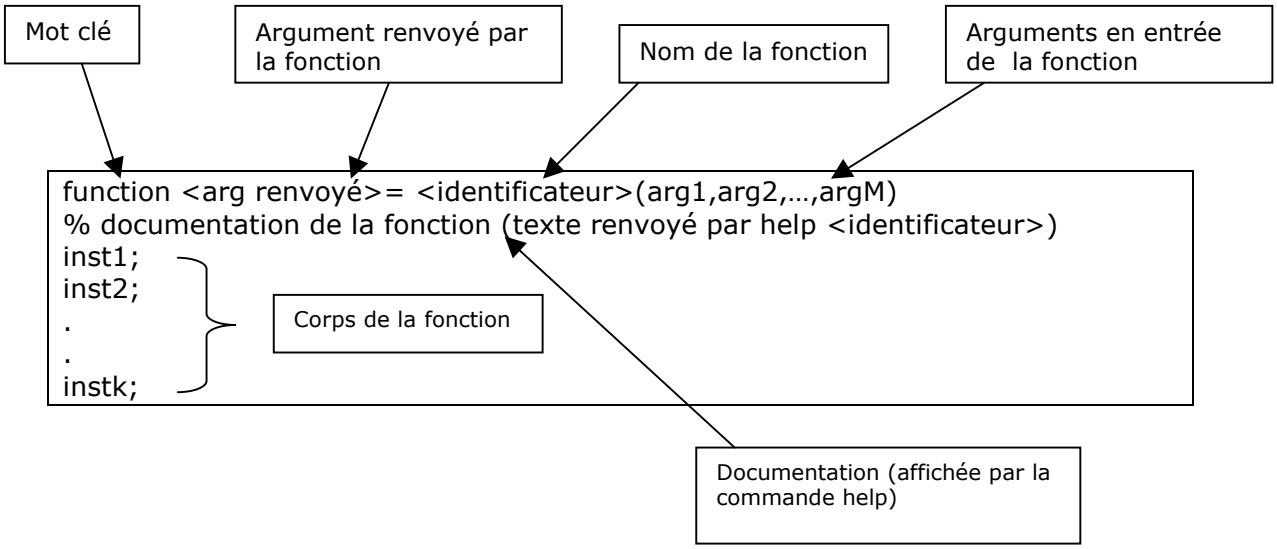
2 Une illustration en traitement d'image, le seuillage d'une image à plusieurs valeur de seuil :

```
for level = 0.0: +0.1: 1.0,
ironseuil=im2bw(iron,level);
figure,imshow(ironseuil),title(['iron seuil = ',num2str(round(level*256))]);
pause(1);
end;
```

3.2.9. Les fonctions en Matlab

Par l'intermédiaire des fichiers Matlab (*.m) nous avons déjà vu que l'on pouvait écrire des scripts directement réutilisables dans la fenêtre de commande.
Il est également possible d'écrire des fonctions réutilisable de la même façon.
Les fonctions MatLab peuvent posséder leurs propres variables locales et acceptent des arguments d'entrée.
A la manière de tous les langages de haut niveau, ces fonctions commencent par une entête

La syntaxe est la suivante :



Remarque :
L'argument renvoyé peut être bien entendu un vecteur

Exemples :

1 Une fonction très simple

```
function prix = prixTTC(prixHT,tva)
% calcul du prix avec TVA
prix = prixHT * (1+tva);
```

2 Matlab accepte les fonctions récursives (ce n'est pas forcement un bon exemple dans ce cas là :

```
function f=fibo(n)
% fibonnaci
if ((n==0) | (n==1) ) f=1;
else f = fibo(n-1) + fibo(n-2);
end
```

3 Sur l'exemple suivant, on définit une fonction nommée **stat** renvoyant deux valeurs (la moyenne -mean- et l'écart type -std- du vecteur x passé en argument).

```
function [mean,stdev] = stat(x)
n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2/n));
```

n est une variable local de cette fonction

pour appeler une telle fonction (renvoyant deux valeurs sous la forme d'un tableau), il suffit,par exemple, de saisir l'instruction suivante :

```
>>x=[ 1 5 8 7 4 9]
>>[m,s]=stat(x)
```

Pour visualiser le code d'une fonction, il suffit de saisir la commande suivante :

```
>> type <id de la fonction>
```

4. Les opérations de bases de Matlab®

Nous allons présenter ici seulement quelques unes des très nombreuses fonctions de base de Matlab. Une présentation complète et exhaustive n'est pas le sujet de cette introduction au logiciel Matlab. Le lecteur pourra approfondir ces connaissances grâce à l'aide de Matlab. De plus, le lecteur est invité à consulter l'aide des commandes présentées afin de pouvoir comprendre l'ensemble de leur utilisation.
Les fonctions présentées sont regroupées en catégories par rapport au traitement effectué. Les catégories sont les suivantes :

- Environnement de développement
- Fonctions mathématiques
- Manipulation des données
- Entrées / Sorties
- Tracée graphique et visualisation 3D

4.1. Environnement de développement

Fonctions liées à la fenêtre de commande :

clc : effacement de la fenêtre de commande des anciennes commandes effectuées

exit : permet de quitter Matlab

Fonctions d'aide :

help : génère l'aide d'une commande

info : fournit des informations sur la version de Matlab

lookfor <mot_clé> : Recherche du mot-clé indiqué dans toutes les entrées d'aide

Fonctions liées à l'espace de travail :

clear : supprime toutes les variables du workspace, si une variable est passé en argument seul cette variable est supprimée

who, whos : fait la liste des variables de l'espace de travail

Fonctions liées à la gestion des fichiers et au système d'exploitation :

dir ,ls (sous unix): fait la liste des fichiers du répertoire de travail

pwd : indique le répertoire courant

cd : permet de changer de répertoire

mkdir : création d'un répertoire

path : visualiser ou changer le chemin de recherche de répertoire de MATLAB

4.2. Fonctions mathématiques

Il s'agit ici de ne présenter que les fonctions les plus usuelles. De nombreuses autres fonctions sont implémentées dans des domaines très diverses (transformées de Fourier, méthodes numériques linéaires et non linéaires, interpolation, géométrie algorithmique - en particulier les problèmes de maillage- …)

Fonctions mathématiques de base

sin, cos, tan : pour le sinus, le cosinus ou la tangente d'un angle
sinh, cosh, tanh : pour le sinus hyperbolique, le cosinus hyperbolique ou la tangente hyperbolique d'un angle.
asin, acos, atan : pour l' arcsinus, l' arccosinus ou l' arctangente d'un angle.
asinh, acosh, atanh : pour le arcsinus hyperbolique, le arccosinus hyperbolique ou la arctangente hyperbolique d'un angle.
exp : fonction exponentiel
log, log10, log2 : fonctions logarithmes
sqrt : racine carrée
real, imag : parties réelles et imaginaires d'un nombre complexe
conj : conjugué d'un complexe
abs : module d'un complexe (ou valeur absolu)
angle : argument d'un complexe
rem ,mod : quotient et reste de la division euclidienne
sign : signe
round, fix, floor, ceil : les différents arrondis

Fonctions liées aux tableaux et matrices:

size : renvoie la taille d'une matrice
max, min : renvoie le maximum, le minimum d'un tableau
sort : trie en ordre croissant les éléments d'un tableau
sum : effectue la somme des éléments d'un tableau
prod : effectue le produit des éléments d'un tableau
zeros : création d'un tableau de zeros (zero(4) renvoie une matrice 4x4 de zeros , zeros(4,1) renvoie le vecteur nul de dimension 4)
eye : renvoie la matrice identité
rand : génération de nombre aléatoire selon un loi gaussienne (rand(4) renvoie une matrice 4x4 générée de façon aléatoire)
randn : même principe mais la génération suit une loi normale

Fonctions liées à l'algèbre linéaire

det : calcul du déterminant d'une matrice carrée
rank : rang de la matrice passée en argument
trace : trace d'une matrice (somme de la diagonal)

inv : calcul de la matrice inverse d'une matrice carrée
lu : factorisation LU d'une matrice
eig : valeurs et vecteurs propres (*[V,D] = eig(A)* permet de récupérer les vecteurs propre V et les valeurs propres D)
polyeig : permet de récupérer le polynôme caractéristique l'une matrice

Fonctions liées à l'analyse des données

mean : moyenne des valeurs d'un tableau de données
median : valeur médiane d'un tableau de données
std : écart type d'un tableau de données
corrcoef : coefficient de corrélation

Fonctions liées aux polynomes

roots : racine d'un polynome
polyval : évaluation d'un polynome
polyfit : polynome d'interpolation

Fonctions à la conversion de coordonnées

cart2sph : conversion de coordonnées cartésiennes vers les coordonnées sphériques
cart2pol : conversion de coordonnées cartésiennes vers les coordonnées polaires
pol2cart : conversion de coordonnées polaires vers les coordonnées cartésiennes
sph2cart : conversion de coordonnées sphériques vers les coordonnées cartésiennes

4.3. Manipulation des données

Fonctions d'entrée / sortie sur les fichiers

fopen : Ouverture d'un fichier
fclose : fermeture d'un fichier
fread : lecture de données binaires dans un fichier ouvert
fscanf : lecture de données formatées dans un fichier ouvert
fprintf : écriture de données formatées dans un fichier ouvert
fwrite : écriture de données binaires dans un fichier ouvert
fseek : positionnement dans un fichier ouvert

Fonctions à la lecture des images

imfinfo : retourne de l'information sur une image passée en paramètres
imread : lecture d'une image
imwrite : écriture d'une image

4.4. Tracée graphique et visualisation 3D

Nous allons voir dans ce paragraphe quelques outils de tracés sous Matlab, ces outils permettent de tracer aussi bien en 2D (courbes) qu'en 3D (surfaces et volumes). Des exemples en fin de document devraient vous permettre de mieux comprendre …

En 2D

box : affichage d'un repère en deux dimensions
plot : affichage d'une droite 2D passant par les points du vecteurs de complexes passés en paramètre, affichage d'une courbe (x,y=f(x)) (cf. exemple 6.1)
polar : tracée en coordonnées polaire (cf exemples 6.2)

En 3D

plot3 : tracé en 3D (cf. exemples 6.3)
cylinder, sphere : génération d'un cylindre, de la sphère unité (cf. exemple 6.4)
mesh, surf, waterfall : 3 visualisations de surface en 3D (cf.exemples 6.5)

5. La toolbox traitement d'images de Matlab®

Ce paragraphe regroupe une grande partie des opérations étudiées dans le cours de traitement et d'analyse d'images du chemin de spécialité ***"Vision"*** de 5ième année de l'ENISE. Le lecteur est donc invité à consulter le polycopié de cours ***"Traitement et Analyse des images"*** pour découvrir la théorie de ces différents traitements.
Comme dans le paragraphe précédent, il n'est pas fait une description exhaustive des différents outils implémentés dans "***Image Processing Toolbox"*** de Matlab. Le choix effectué correspond au cours de 5ième année, le lecteur pourra découvrir d'autres outils en auto apprentissage.

5.1. Ouverture, sauvegarde et conversions d'images

imfinfo : retourne de l'information sur une image passée en paramètres
imread : lecture d'une image
imwrite : écriture d'une image

Les exemples 6.6 et 6.7 illustrent l'utilisation de ces fonctions

5.2. Opérations arithmétiques sur les images

Les exemples de cette section se trouvent en 6.8

imadd : additionne deux images
imsubstract : différence de deux images
imabsdiff : effectue la différence absolue de deux images (|f(x,y)-g(x,y)|)
immultply : multiplie une image par une constante ou multiplie deux images
imdivide : divise une image par une constante ou divise deux images
imcomplement : négatif d'une image

5.3. Valeurs des pixels et Histogramme

imhist : calcule l'histogramme d'une image (cf. exemple 6.9)
histeq : effectue l'égalisation de l'histogramme
impixel : renvoie la valeur du pixel d'une image
improfile : renvoie le profile d'une ligne d'une image
accès aux pixels : On peut modifier la valeur des pixels d'une image directement dans la matrice image (cf. exemple 6.10)
mean2 ,std2 : moyenne et écart type d'une image

5.4. Seuillage des d'images

im2bw : seuillage d'une image
graythresh : détermination du seuillage par maximisation de la variance interclasse

Pour ces deux fonctions consultés l'exemple 6.11

5.5. Convolution numérique des images

edges : détection de contour d'une image, plusieurs filtres sont implémentés (sobel, prewitt, roberts, laplace, canny)
imfilter : filtrage par convolution. Attentien le masque de convolution doit intégrer le division par la somme des éléments (exemple m=[0.33 0.33 0.33 ; 0.33 0.33 0.33 ;0.33 0.33 0.33] et cf. 6.12)
medfilt2 : implémentation du filtre médian. Rappel :il ne s'agit pas d'un filtrage par convolution numérique cf. cours. Dans l'exemple 6.13 , il est également montré comment ajouté du bruit à une image (**imnoise**).

5.6. Outil issu de la morphologie mathématique.

imerode : érosion
imdilate : dilatation
imclose : fermeture
imopen : ouverture

Ces quatre opérations demandent deux paramètres : l'image à traiter (binaire ou en niveau de gris) et l'élément structurant utilisé. Ce dernier peut soit être définit directement par une matrice soit définit en utilisant la fonction **strel** de matlab qui renvoie en fonction de différents paramètres les éléments structurants les plus classiques. Les exemples 6.14 de la section illustrent ces différents points.

bwmorph : opérations sur les images binaire (bouchage de trous -1 pixel isolé-, suppression des pixels isolés, suppression des pixels interieurs**-** contour**-,** squelétisation …) cf. exemple3 de la section 6.14
imclearborder : suppression des objets qui touchent le bord, cette fonction marche aussi sur les images en niveau de gris
imbothat : chapeau haut de forme

5.7. Espace de couleurs

hsv2rgb, rgb2hsv : conversion de hsv en rgb et vice-versa
ycbcr2rgb, rgb2ycbcr : conversion de yCbCr en rgb et vice-versa
rgb2ntsc, ntsc2rgb : conversion de ntsc en rgb et vice-versa

6. Exemples

6.1. Tracé de fonctions avec la fonction plot

```
x = -2*pi:.1:2*pi;
y = cos(x)+sin(x);
plot(x,y)
```

6.2. Tracé de fonctions en coordonnées polaires

Sans indiqué une forme de trait :

```
t = 0:.01:2*pi;
polar(t,sin(2*t).*cos(2*t))
```

En pointillé rouge … :

```
t = 0:.01:2*pi;
polar(t,sin(2*t).*cos(2*t),'--r')
```

6.3. Courbe 3D

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
```

La même chose avec grille permettant une meilleurs visualisation :

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
grid on
axis square
```

6.4. Cylindres

Cylindre de profil cosinus, (cylinder (n) génère un cylindre de rayon n)

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(2+cos(t));
surf(X,Y,Z)
axis square
```

6.5. Visualisation de surfaces

```
[X,Y,Z] = peaks(30);
waterfall(X,Y,Z)
```

6.6. Lecture et affichage d'images

```
% nettoyage de l'environnement Matlab
clear,close all;
% lecture d'un fichier image
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
% affichage d'une image
i=2;
imshow(iron),title(['iron',num2str(i)]);
% affichage avec les niveaux de gris
figure, imshow(iron,[]), colorbar;
imfinfo('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp')
```

6.7. Conversions d'images

```
% nettoyage de l'environnement Matlab
clear,close all;
% lecture d'un fichier image
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
% conversion
imwrite(iron,'D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.png','png');
imwrite(iron,'D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.jpg','jpg');
```

6.8. Opérations arithmétiques sur les images

Addition de deux images (ici il s'agit de la même image pour augmenter la luminosité)

```
% addition des images
I=imread('pout.tif');
I2=imread('pout.tif');
K=imadd(I,I2);
Imshow(I);figure;imshow(K);
```

Multiplication d'une image par une constante

```
I = imread('moon.tif');
J = immultiply(I,1.2);
imshow(I);
figure, imshow(J)
```

Négatif d'une image

```
im= imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
imn = imcomplement(im);
imshow(im),title('image iron');
figure, imshow(imn),title('négatif de l''image iron');
```

6.9. Histogramme des images

```
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
% sauvegarde de l'histogramme dans un tableau
hist = imhist(iron);
% affichage de l'image et de son histogramme
imshow(iron),title('image iron');
figure,imhist(iron),title('Histogramme de l''image iron');
```

6.10. Accès au pixels, valeurs des pixels

Exemple de segmentation en trois classes (on peut implémenter le seuillage binaire de façon analogue) :

```
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
for x=1:256
for y=1:256
if (iron(x,y) < 50) iron(x,y)=0;
elseif   (iron(x,y) < 100) iron(x,y)=100;
else iron(x,y)=255;
end;
end;
end;
imshow(iron);
```

6.11. Seuillage des images

```
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
%seuillage d'image et affichage du résultat ...
level=0.3;
ironseuil=im2bw(iron,level);
figure,imshow(ironseuil);
% choix du seuil par maximisation de la variance interclasse
level = graythresh(iron)
level
ironseuil=im2bw(iron,level);
figure,imshow(ironseuil);
```

6.12. Filtrage par convolution numérique des images

```
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
h=[1/16 2/16 1/16
2/16 4/16 2/16
1/16 2/16 1/16]
irongaus=imfilter(iron,h);
imwrite(irongaus,'D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\irongaus.bmp');
imshow(iron), title('Original'), figure, imshow(irongaus),title('filtrée');
```

6.13. Filtre médian

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
imshow(I)
figure, imshow(J)
L = medfilt2(J,[3 3]);
figure, imshow(L)
```

6.14. Morphologie mathématique

Morphologie mathématique binaire et en niveau de gris avec "détermination manuelle" de l'élément structurant :

```
% lecture de l'image à traiter
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
% Seuillage automatique par maximisation de la variance inter-classe
level = graythresh(iron)
level
bw=im2bw(iron,level);
% définition de l'élément structurant SE
SE=[1 1 1;1 1 1;1 1 1]
% dilatation de l'image binaire
dilbw=imdilate(bw,se);
figure, imshow(dilbw), title('image binaire dilaté');
% érosion de l'image binaire
erodbw=imerode(bw,se);
figure, imshow(erodbw), title('image binaire érodé');
% ouverture de l'image binaire
openbw=imopen(bw,se);
figure, imshow(openbw), title('ouverture morphologique de l''image binaire');
% ouverture de l'image binaire
closebw=imclose(bw,se);
figure, imshow(closebw), title('fermeture morphologique de l''image binaire');
% morpho en niveau de gris
closeiron=imclose(iron,se);
figure, imshow(closeiron), title('fermeture morphologique ndg de iron');
```

Morphologie mathématique binaire en utilisant la fonction strel pour calculer l'élément structurant :

```
% lecture de l'image à traiter
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
% Seuillage automatique par maximisation de la variance inter-classe
level = graythresh(iron)
level
bw=im2bw(iron,level);
% définition de l'élément structurant SE sous la forme d'un disque de rayon lissé
se1 = strel('disk',4,0)
% définition de l'élément structurant SE sous la forme d'un disque de rayon non lissé
se2 =  strel('disk',4)
% dilatations de l'image binaire
dilbw1=imdilate(bw,se1);
figure, imshow(dilbw1), title('image binaire dilaté avec se1');
dilbw2=imdilate(bw,se2);
figure, imshow(dil2bw), title('image binaire dilaté avec se2');
```

Différentes utilisations de la fonction bwmorph :

```
% lecture de l'image à traiter
iron=imread('D:\DOCUMENT\COURS\Annee5\CoursVision\VISION\iron.bmp');
% Seuillage automatique par maximisation de la variance inter-classe
level = graythresh(iron)
level
bw=im2bw(iron,level);
% élimination des pixels isolés
bw1 = bwmorph(bw,'clean');
figure, imshow(bw1), title('élimination des pixels isolés');
% suppression des pixels intérieurs
bw2 = bwmorph(bw,'remove');
figure, imshow(bw2), title('suppression des pixels intérieurs');
% squelétisation
bw3 = bwmorph(bw,'skel',Inf);
figure, imshow(bw3), title('squelétisation');
```