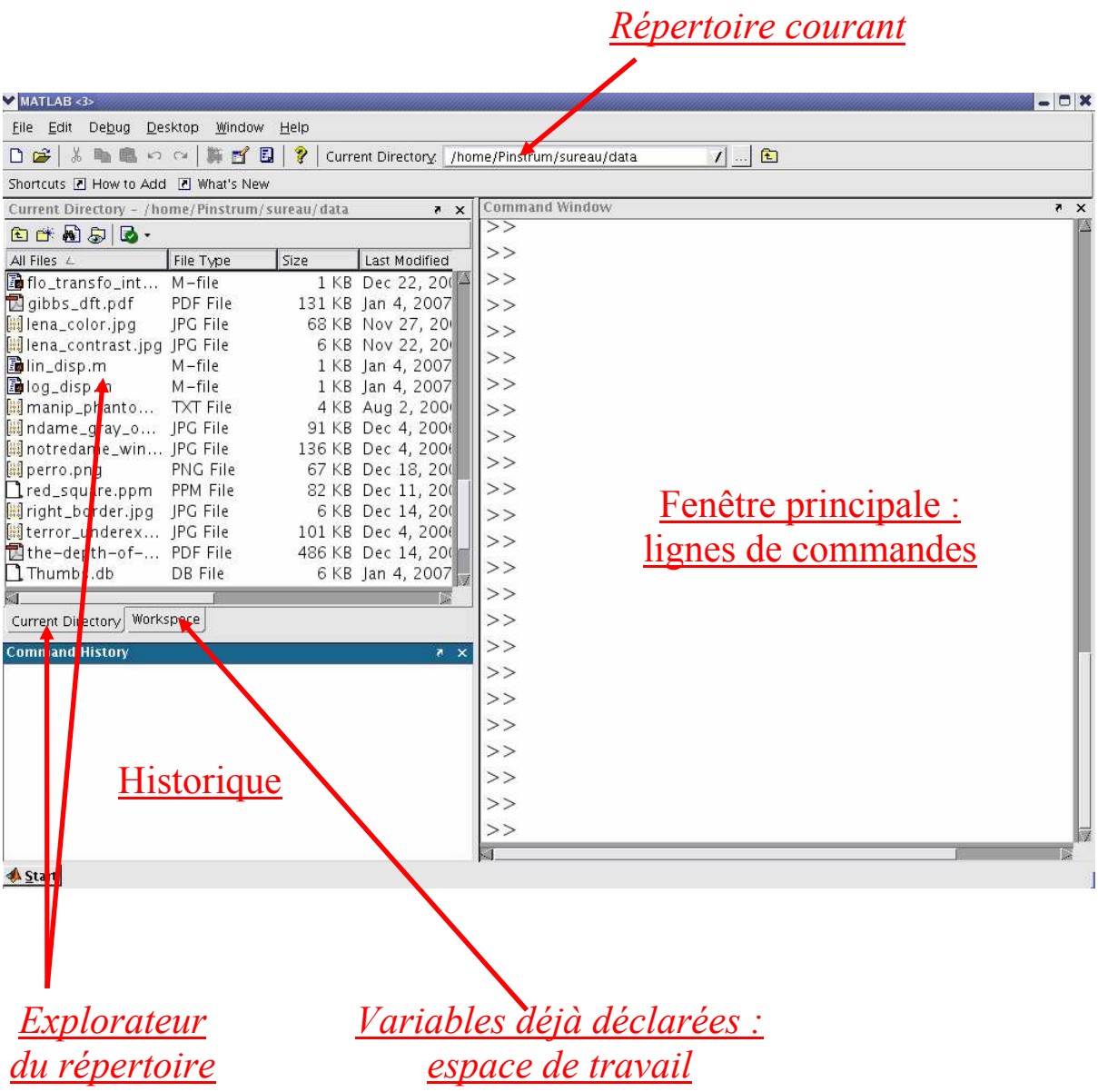
Interface Matlab

Interface par défaut

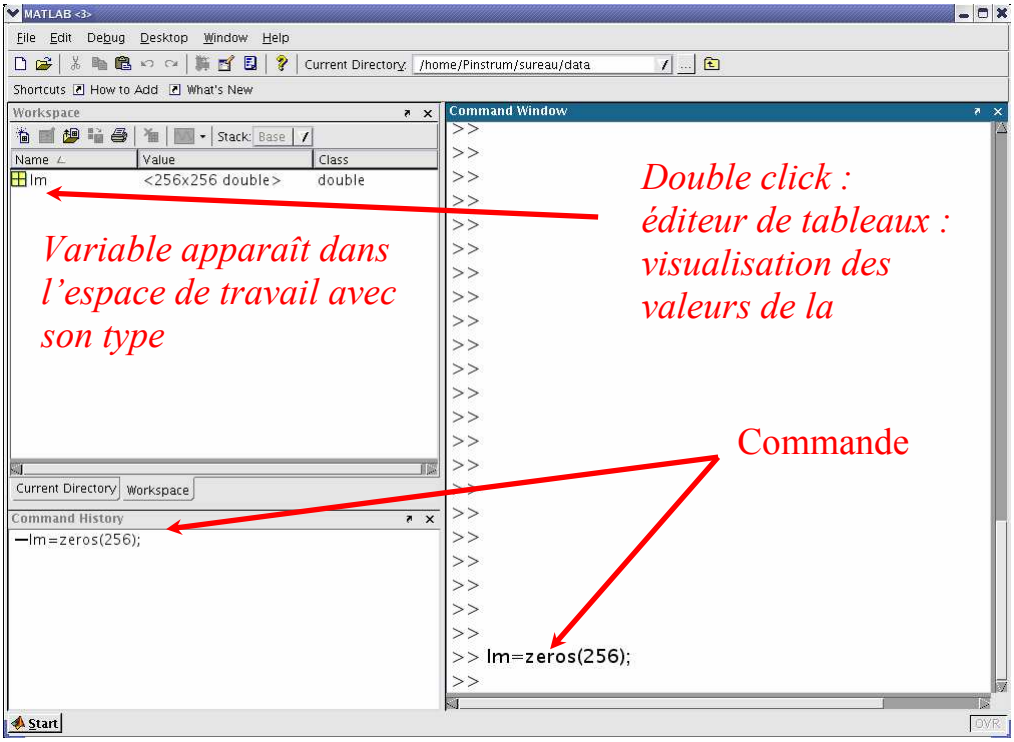
On peut la sélectionner via le menu Desktop->Desktop Layout->Default

4 fenêtres importantes, 3 montrées en simultané :

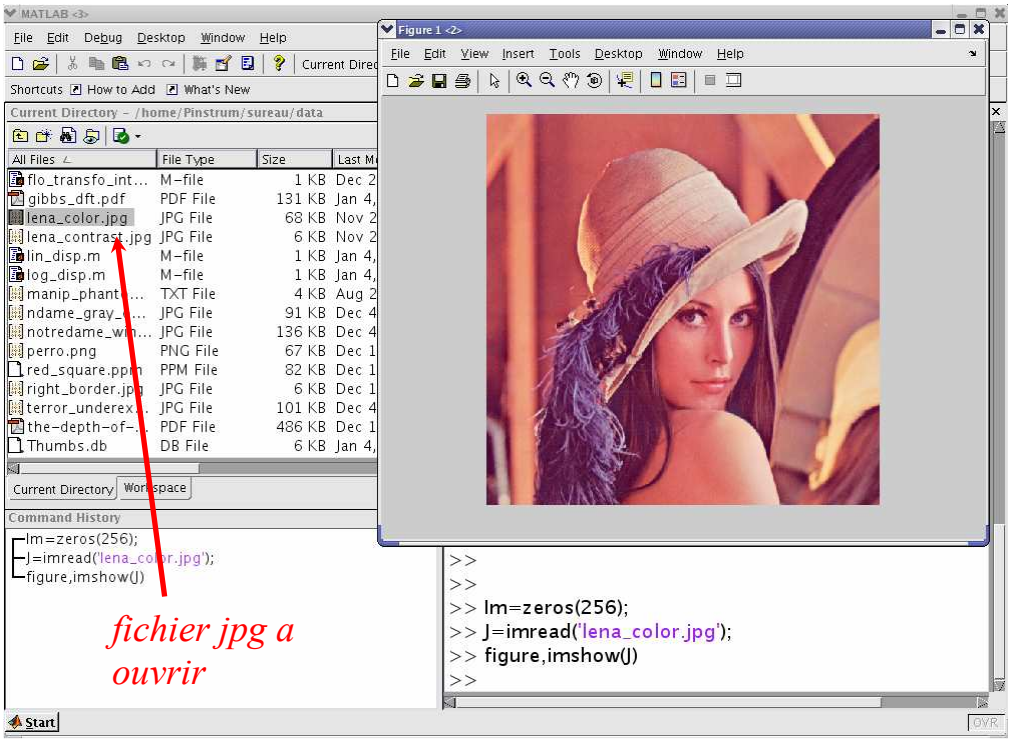
- Fenêtre principale où on va venir entrer les lignes de commandes
- L'historique des lignes de commandes déjà entrées (on peut copier coller via des clicks droit)
- Un explorateur de répertoire et l'espace de travail qui contient les données déjà déclarées avec leur taille, leur type. On passe de l'un à l'autre par les onglets
- Le répertoire courant à partir duquel on va venir charger/sauver des images, des nouvelles fonctions etc



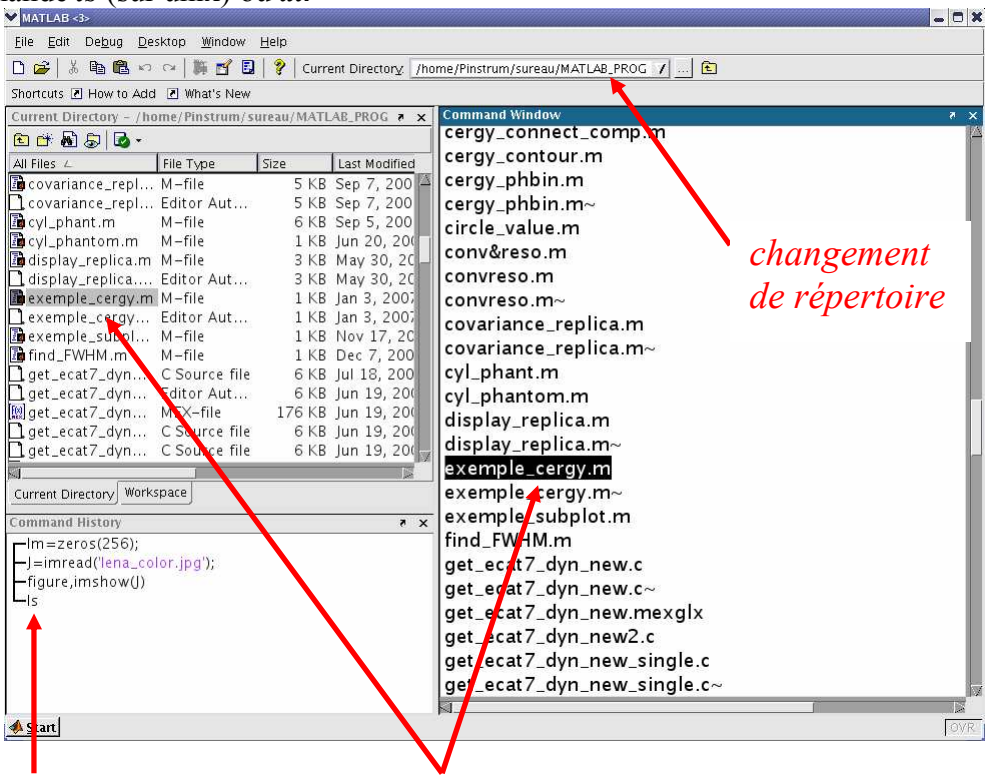
Dans l'exemple suivant, on a crée une matrice im de taille 256x256 remplie de zéros (similaire à calloc).



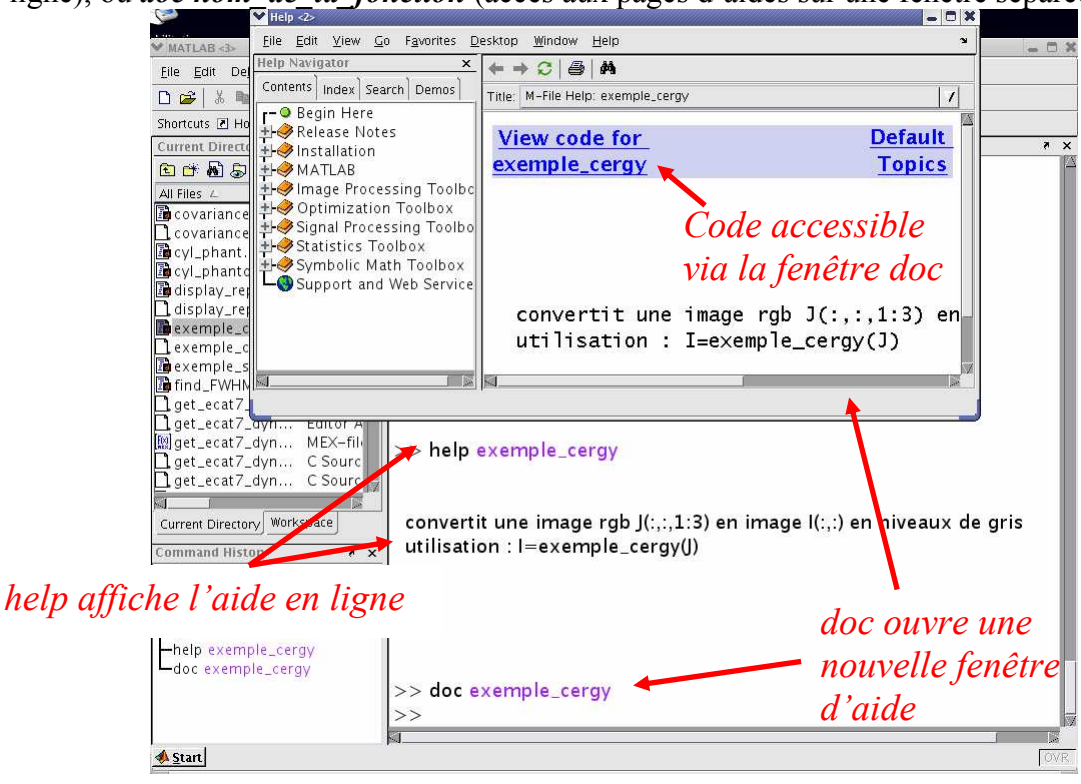
Deux autres types de fenêtre vont être utilisées : les figures, et la fenêtre d'édition.
Pour la fenêtre figure : on vient lire une image et la représenter dans une nouvelle fenêtre, en utilisant les commandes imread puis figure et imshow



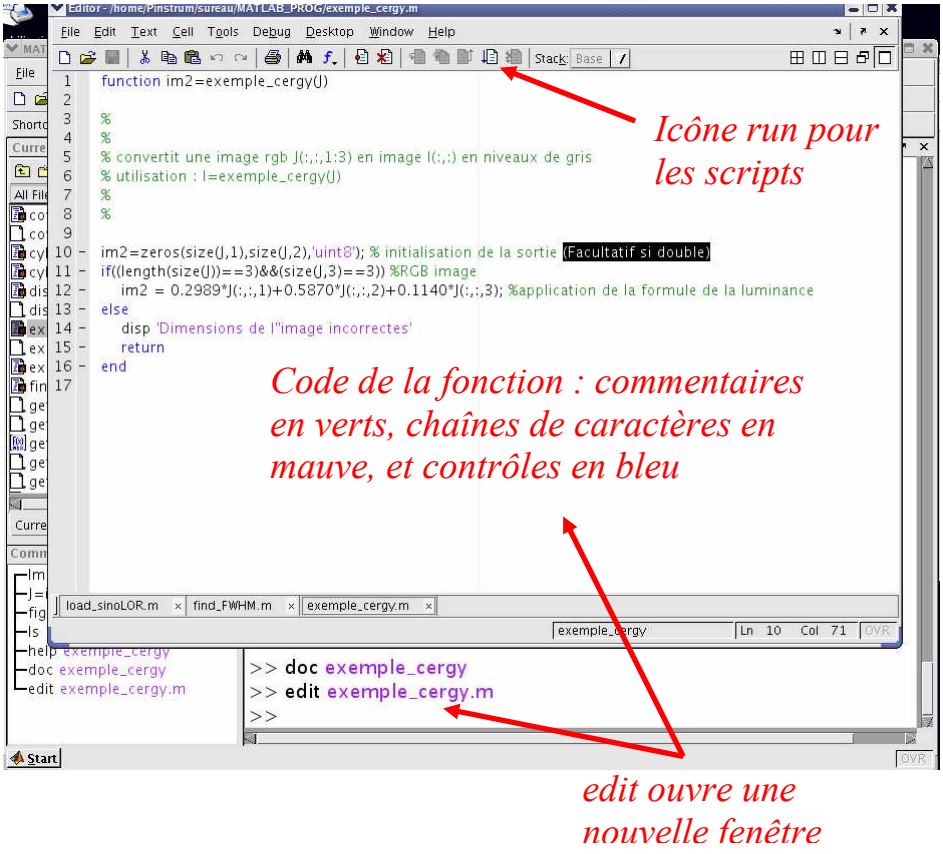
Pour la fenêtre d'édition, il faut tout d'abord se placer dans le bon répertoire (celui où sont sauvées les fonctions). Les fonctions ont pour extension « .m » (nom_de_la_fonction.m)
On peut voir le contenu du répertoire dans la fenêtre « explorateur », ou bien en tapant la ligne de commande ***ls*** (sur unix) ou ***dir***



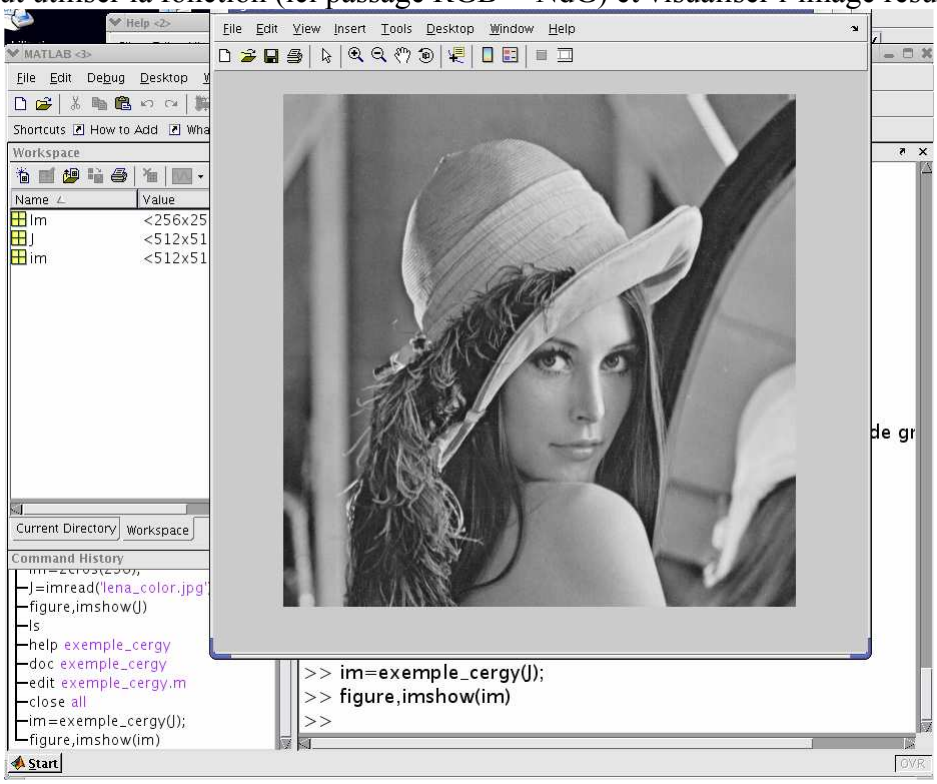
On peut obtenir des informations sur la fonction en tapant ***help nom_de_la_fonction*** (aide en ligne), ou ***doc nom_de_la_fonction*** (accès aux pages d'aides sur une fenêtre séparée)



Si on veut maintenant éditer la fonction, on peux utiliser la fenêtre doc ou la commande ***edit nom_de_la_fonction.*** Une nouvelle fenêtre apparaît avec les commandes qu'exécute la fonction.



On peut utiliser la fonction (ici passage RGB-> NdG) et visualiser l'image résultante.



Routines de base de Matlab

Toutes les routines natives disposent d'une page d'aide, accessible via ***doc nom_de_la_routine*** (pages d'aides) ou ***help nom_de_la_routine*** (aide en ligne), ou via le menu help

Matlab ressemble au C. Quelques variations :

- le point virgule après chaque instruction n'est pas nécessaire. Il signifie que l'opération est exécutée mais que le résultat n'est pas affiché. Si on veut afficher le résultat, mettre des virgules entre commandes sur une même ligne ou si aucun symbole si la commande est en fin de ligne
    - Exemple : ***x=2, y=3 ; z =2*** affichera x=2 puis z=2
- Il n'est pas nécessaire de déclarer les variables, elles le sont par défaut selon leur première affectation. Les nombres sont en « double » par défaut.
    - ATTENTION lorsqu'on ne veut pas qu'elles soient en double (souvent le cas !!!)
    - ATTENTION lorsqu'on remplit une matrice : la taille va changer au cours des affectations… Parfois très long… Essayer de déclarer la taille de la matrice en utilisant la routine ***zeros()…***
- Les variables s'éliminent avec ***clear*** (***clear all*** pratique pour éliminer toutes les variables)
- Les vecteurs/matrices sont indexées à partir de 1 et non 0
- Les strings sont rentrés entre ***' '***
- Les commentaires sont après ***%***
- L'opérateur « not » est ~ et non ***!***

Déclarations de variables pour les nombres

Utiliser les routines classiques au moment de la première affectation

Routine	Interprétation
logical()	binaire (1-bit)
uint8(), uint16(), uint32(), uint64()	entiers non signés 8,16,32,64 bits
int8(), int16(), int32(), int64()	entiers signés 8,16,32,64 bits
single()	simple précision
double()	double précision

Autres conversion : ***str2num(), str2double(), bitget()***
tests : ***isempty(), islogical(), isinteger()....***

Tests:

| , **&** : ou binaire, et binaire

||, &&, = =, ~ =, <, <=, >, >= : ou logique, et logique, test d'égalité (sans espace entre les =), test de différence (sans espace entre ~ et =), relations d'ordre...

Vecteur

Affectation directe (pénible)

Vecteur ligne ***A=[2 5 8]***, vecteur colonne ***B=[2 ;5 ;8]***

Affectation plus rapide à taper (pratique)

A=2 : 3 : 8 (à lire : de 2 à 8 par pas de 3),

B=A' (transposée de A) ou ***B=(2 : 3 : 8)'***

Routines utiles :

length() longueur du vecteur

sum(), mean(), std(), min(), max(), mod(), floor(),ceil() : somme de tous les éléments du vecteur, moyenne, ecart-type, min, max, vecteur modulo (ex ***mod(v,2)***) partie entière, partie entière+1

abs(), angle() : valeur absolue/module, phase (nombre complexes)

Opérations sur les vecteurs/ou matrices :

sur l'ensemble des éléments : **+, -, *** (produit matriciel), /, ^, \

éléments à éléments : **.***, **./**, **.^**

Indexations rapides :

a(5 : end) = 0 ; mise à 0 des valeurs de 5 à ***length(a)***

Matrices

Déclaration détaillée:

A=zeros(256,512,'uint8') ; : matrice de 256 lignes et 512 colonnes d'uint8

Déclaration rapide :

B=logical(zeros(size(A))); Note : zeros(size(A),'logical') n'existe pas : seul moyen de déclarer une matrice binaire

Accéder aux valeurs d'une matrice

Im matrice de taille 128x128

Alors ***Im(129)*** pareil que ***Im(2,1)***

Plus généralement, ***Im(:)*** transforme Im en le vecteur des composantes de Im.

Faiblesse de Matlab : boucles. Il faut chercher à vectoriser au maximum, si on recherche des routines rapides.

Routines utiles :

size(), std(), det()

sum(), mean(), min(), max() opèrent sur une dimension… Les répéter autant de fois que de dimensions, ou utiliser l'opérateur de vectorisation :

sum(sum(Im)) ou ***sum(Im(:))***

« : » indique tous les nombres dans une dimension ; **Im(1, :)=5** veut dire que la première ligne est mise à 5

Une routine extrêmement pratique : ***find()***

Renvoie l'adresse vectorisée des éléments qui rendent l'expression entre parenthèse vraie

Exemple

***Im(find(Im<=10))=0*;** est une façon extrêmement rapide et compacte de mettre toutes les valeurs de Im inférieures ou égales à 10 à 0

Dessin de figures

figure ouvre une nouvelle figure (sinon on écrit dans la figure précédente)

en 2D : ***plot(), hist(), contour() ...***

en 3D : ***imshow()*** (pratique car s'adapte immédiatement au format et à l'échelle), ***image(), imagesc(), surface()***

Ex : forcer l'échelle : ***imshow(Im, [0 255])***

Ex : dessin en rouge d'une fonction sinus sur une période, chaque échantillon étant representé par un point et des traits plein reliant chaque échantillon consécutif :

```
x=0 :0.1 :2*pi ;
y=sin(x) ;
figure,plot(x,y,'r.-') ;
```

A noter la commande classique ***imagesc(abs(Im)), axis image, colormap gray, colorbar*** : on représente le module de l'image, avec des axes qui respectent la taille de l'image (ici pixels carrés par défaut), on lui applique une palette indexée en niveau de gris, et on affiche la palette de couleur à côté de l'image

I/O : très proche du C

fopen, fread, fwrite, fclose
fscanf, fgetl, fprintf
disp
imread, imwrite, imfinfo

Scripts/fonctions

edit ouvre le menu d'édition

Un script est une série de commandes qui sont lancées à partir de l'icône run

Une fonction est accessible via le menu de commandes en appelant le nom sous lequel la fonction a été sauvé (attention, ce n'est pas nécessairement le même nom que le nom de la fonction principale), avec le chemin correct (PENSER A CHANGER/AJOUTER LE CHEMIN DANS LES REPERTOIRES DE FONCTION/DE TRAVAIL)

```
function [a, b]=nom_interne_de fonction(Im1, Im2) ;
...
[J K]=appelle_sous_fontion() ;
...
end

function [J,K]=appelle_sous_fonction();
...
end
```

Boucles

```
for k=50:-1:1
...
end

While (...)
...
end

if(Im(k1,k2)~=3)...
elseif...
else...
end
```

switch expr
case expr<10...
case {11 12 13}...
otherwise...
end

ATTENTION, MATLAB GERE MAL LES BOUCLES (très lent...)
POUR DES UTILISATIONS AVANCEES, VECTORISER AUTANT QUE POSSIBLE LE CODE.

Les routines plus spécifiques au traitement d'images seront présentées au début de chaque TP...