Séances de travaux pratiques 1 et 2 :

Prise en main de Matlab et GNU Octave

Année universitaire 2013/2014

MR. BEY M-A

DAUPHINE
UNIVERSITÉ PARIS

MIDO
MATHÉMATIQUES ET INFORMATIQUE
DE LA DÉCISION ET DES ORGANISATIONS

Initiation aux projets informatiques

- On va surtout parler de Matlab et d'algorithmique.
- Premiers exemples de modélisation
- Peut servir dans d'autres cours

Qu'est-ce Matlab?

- Logiciel de Calcul Scientifique
- Langage interprété (pas besoin de compilation)
- Peut s'utiliser en ligne de commande ou avec une interface graphique

Avantages et inconvénients

Avantages:

- Facile à utiliser
- Prise en main rapide
- Intéressant en modélisation numérique
- Existence de toolboxes (stats, pde, images, ondelettes, ...)
- Possibilité de l'interfacer avec d'autres langages (C, C++, Fortran, ...)
- Permet de faire du calcul parallèle et du calcul GPU
- Possibilité de compiler le code et de l'exporter

Inconvénients:

- Payant donc pas disponible partout
- Limitations en mémoire

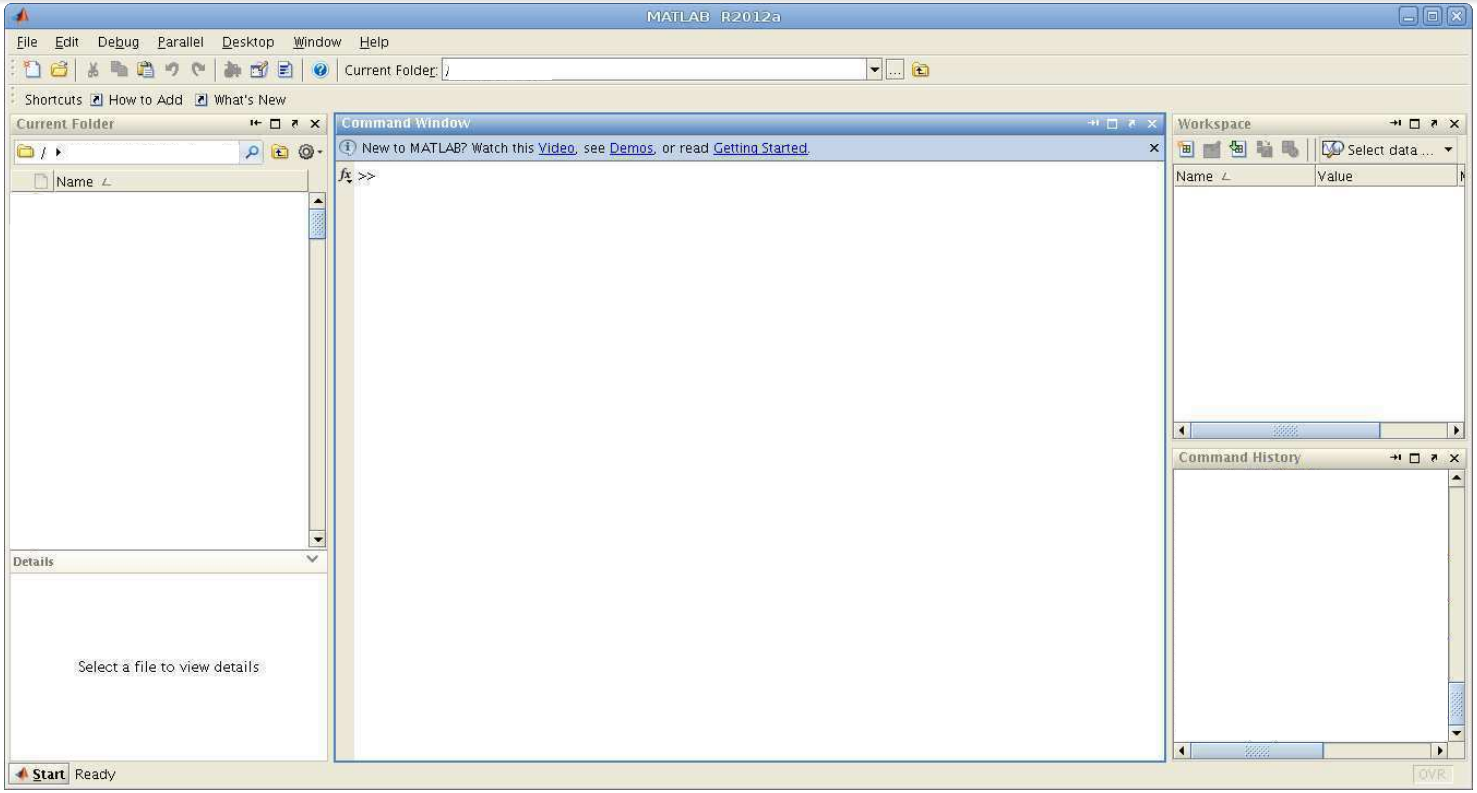
Contexte

- Matlab n'est pas destiné au calcul formel (Maple l'est)
- Alternatives libres
  - python
  - scilab
  - octave

Comment exécuter Matlab?

- Sous Linux, exécuter *matlab* dans un terminal
- *matlab&* à préférer
- Pour sortir, taper *exit* ou *quit*

Fenêtres



- “Current folder”: Arborescence et liste des fichiers
- “Command history” historique des commandes
- “Workspace”: contenu des variables
- “Command window”: fenêtre principale pour l’exécution des instructions

Utiliser l'aide

Pour n'importe quelle commande

- help
- Par exemple:

  ```
  help plot
  ```

Editeur

- Matlab contient son propre éditeur
- indenter est fortement recommandé pour faciliter la lecture!
- On peut choisir d'autres éditeurs

Rappels Unix

- *ls* $\rightarrow$ lister les fichiers
- *pwd* $\rightarrow$ connaître le répertoire courant
- *cd* $\rightarrow$ changer de répertoire
- . représente le répertoire courant
- .. représente le répertoire parent (au-dessus)
- Dans Matlab, on peut exécuter n'importe quelle commande Unix en commençant la ligne par *!*

Vecteurs et matrices

- Plusieurs méthodes:
-

```
v=[1 2 3 4] %dimension (1,4)
M=[1 2 3 4;5 6 7 8] %dimension (2,4)
```

```
v=1:10 %séquence
```

- Pour initialiser un vecteur: *zeros*, *ones*, *eye*
- Pour accéder aux éléments d'un tableau

```
v(1)
M(2,2)
```

Vecteurs et matrices

- Pour connaître la taille d'un tableau

```
size(M)
length(v) %pour les vecteurs
```

- Opérations matricielles +, -, *
  Exemples:

```
M=ones(4,4);
M+M
M*M
M=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16];
v=ones(4,1);
M*v
```

Affichage graphique

- Affichages 2D et 3D
- Commandes *plot*, *surf*, *mesh*
- Exemple:

```
%une droite
a=1:10;
plot(a)
```

Chaînes de caractères

```
s1='hello ';
s2='world';
[s1,s2] %concatenation
```

Fonctions pour le calcul matriciel

- Produit matriciel

```
A=rand(5,5); B=rand(5,5);
A*B
```

- Inversion: on cherche x tel que: $Ax = b$

```
x=inv(A)*b; %Attention!! x=b/A;
```

- Déterminant d'une matrice

```
det(A)
```

- Valeurs propres, vecteurs propres

```
[vp,lambda]=eig(A)
```

- Autres: conditionnement, norme, ...

Exemple de fonction

- Exemple de la fonction f (fichier f.m).
  Conseil: le nom du fichier doit être identique à celui de la fonction.

```
function [s1,s2,s3]=f(e1,e2,e3,e4)
% e1, e2, .., e4: arguments d'entree
% s1, s2, s3: arguments de sortie
....
f=...
return; %facultatif mais conseille
```

- Exemple *cross_prod.m*

```
%calcul du produit vectoriel en 2D
%
function val=cross_prod(v1,v2)
val=v1(1)*v2(2)-v1(2)*v2(1);
return;
```

- Manipulation de fonction: @f

Commandes pratiques en vrac

- *clear* supprime une variable en mémoire
- *close* ferme une figure
- *save* permet de sauver les variables dans un fichier au format Matlab
- *ans* donne la valeur de la variable la plus récente