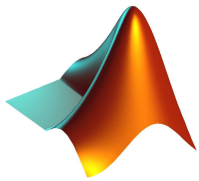
Séance 1 : Introduction à Matlab



Assistants : Zhenzhu Meng, Daniel Vito Papa, Ivan Pascal, Oscar Sepúlveda Steiner, Tomás Trewhela

1 Présentation

Matlab Le terme Matlab vient de la contraction de l'anglais « Matrix - Laboratory ». Comme son nom l'indique, le logiciel Matlab a été initialement conçu pour manipuler des tableaux (*arrays*) et matrices (*matrix*). C'est aujourd'hui un logiciel commercial (de la société MathWorks), mais qui a été initialement développé il y a une trentaine d'années par des universitaires américains.

Pour quoi faire ? Matlab est un logiciel de calcul numérique (à la différence de Maple ou Mathematica qui sont en premier lieu conçus comme des calculateurs symboliques). Matlab est capable de manipuler et de faire des opérations complexes sur des matrices. On peut aussi l'utiliser pour tracer des graphiques, courbes 2D ou 3D, visualiser des données, simuler des événements aléatoires, résoudre des équations différentielles, créer des interfaces ou interagir avec d'autres langages de programmation (C, C++, Fortran 77, etc.) et à bien d'autres fins. Pour cela, outre les fonctions classiques, il existe une multitude de « boites à outils » (*Toolbox*) qui offrent des fonctions spéciales adaptées à un domaine particulier. Par exemple, la boite à outils « Statistics Toolbox » met à disposition plusieurs fonctions utiles en statistique comme le calcul de fonctions de densité de probabilité ou de fonctions de distribution à partir d'échantillons de données.

Des alternatives Chaque logiciel de calcul scientifique à ses particularités, ses défauts et ses avantages. Il est donc important de savoir ce dont on a besoin avant de choisir tel ou tel logiciel. Il existe aussi des alternatives libres (et gratuites !) à Matlab, qui permettent de faire globalement les mêmes choses. Par exemple, Scilab – créé en France – est une puissante plate-forme de calcul libre et gratuite (mais attention, la syntaxe est légèrement différente de celle de Matlab). GNU Octave est un logiciel libre compatible avec Matlab, qui offre globalement les mêmes fonctions que ce dernier.

En route !

2 Le logiciel

Lancer Matlab Après l'ouverture d'une session utilisateur sous Linux, cliquer sur `Application` → `Accessoires` → `Terminal`.

- Créer un répertoire de travail pour le TD1 avec la commande : *mkdir myfiles/TD1*
- Se déplacer dans ce répertoire avec la commande : *cd myfiles/TD1*
- Ouvrir Matlab avec la commande : *matlab*

(Ne pas fermer la fenêtre du terminal, sinon Matlab se ferme !)

Quelques notions de base à savoir

- Il n'y a pas d'indice 0.
- On utilise la commande *clear all* pour supprimer toutes les variables. Utiliser de préférence la commande *clearvars* ou *clear variables* car *clear all* supprime toutes les variables ainsi que les fonctions et scripts qui sont utilisés.
- On utilise la commande *close all* pour fermer toutes les fenêtres.
- Dans le menu `help` sous l'onglet `function browser` on trouve des informations sur l'utilisation de la fonction désirée. Quand on ne connaît pas la ou les fonctions à utiliser, on choisira l'onglet `product help`. Une recherche sur internet peut aussi permettre de trouver des informations rapidement. On peut aussi utiliser les fonctions *help* et *lookfor*, suivies du nom de la fonction dont on souhaite obtenir des informations.
- Comme nous allons le voir un peu plus loin, Matlab permet de créer des scripts et des fonctions grâce aux fichiers .m.
- Les noms de fonction commencent toujours par une lettre : par exemple *2en1* n'est pas un nom de fonction valide.
- Attention, Matlab fait la distinction entre les lettres minuscules et majuscules.

Les fenêtres (disponibles dans l'onglet `Desktop`)

- `Current Folder` : cela permet d'ouvrir rapidement des fichiers dans Matlab. Notez que le répertoire de travail (*Work directory*) doit être choisi de façon à ce que l'on y trouve tous les fichiers utiles au TD (données, m-files, etc.). C'est ici le répertoire TD1.

- `Workspace` : il permet de connaître les variables déclarées et les valeurs de celles-ci.

- `Command history` : il rappelle tout l'historique des commandes qui ont été exécutées dans Matlab.

- `Command Window` : il permet d'executer directement une opération. On n'utilise en général pas cette fenêtre sauf pour effectuer des calculs très simples.

- `Editor` : c'est là ou vous allez passer le plus clair de votre temps ! Cette fenêtre permet de créer, d'éditer et de lancer les fichiers de commandes Matlab avec l'extension .m. Pour exécuter de tels fichiers, il suffit de cliquer sur `Run` ou bien de sélectionner la partie des commandes à executer et presser la touche `F9`. On peut aussi lancer directement le fichier « monfichier.m » en tapant monfichier dans la fenêtre `Command Window`.

3 Exercices depuis la « Command Window »

1. Simplifier l'écriture du nombre complexe $z_1 = (\sqrt{3} + 3i)^5 = ... - .....i$
2. Simplifier l'écriture du nombre complexe $z_2 = (2\sqrt{3}e^{\frac{i\pi}{3}})^5 = ... - .....i$
3. Faire une figure de la fonction *sinus* dans l'intervalle $[0; 2\pi]$.
   fonctions utiles : générer un vecteur avec $x = [0 : 0.1 : 2\pi]$ *et utiliser la fonction plot pour visualiser la figure.*
4. Représenter le graphique de $sin(\pi x)$ entre 0 et 3 en trait continu rouge. Rajouter sur la même figure le graphique de $cos(\pi x)$ entre 0 et 1 en trait continu vert, puis tracer cette dernière fonction dans une autre fenêtre.
   fonctions utiles : fplot, hold on, figure
5. Générer un vecteur colonne de 10 nombres aléatoires. Afficher sa longueur et calculer sa somme, sa moyenne, sa variance, son minimum et son maximum.
   fonctions utiles : rand, mean, var, max, min
6. Générer une matrice de nombres aléatoires de dimension 20×30.
   - Afficher ses dimensions et calculer la moyenne, la variance et le maximum de ses lignes.
   - Lire la valeur se trouvant à la ligne 2 et colonne 3. Affecter la valeur π à la place de la valeur se trouvant à la ligne 1 et colonne 2.
   - Calculer la transposée de cette matrice.
     fonctions utiles : rand, mean, var, max, min

4 Scripts et fonctions Matlab

La fenêtre de commande ne permet pas d'effectuer plusieurs opérations successivement sans l'intervention de l'utilisateur. On utilise pour cela des fichiers texte qui regroupent plusieurs commandes. Ces fichiers ont l'extension .m et sont couramment appelés scripts ou m-files. De la même façon, il peut exister des fonctions qui par défaut n'existent pas dans Matlab et que l'utilisateur doit définir par lui-même. Ces fonctions sont définies dans un fichier portant aussi l'extension .m.

Observer cet exemple de script et décrire son fonctionnement :

```
%============================
% SCRIPT n°1
%============================

clear all ;
close all ;

m=input('Nombre:?');
f=1;

for k=1:1:m
    f=f*k;
```

```
end

fprintf('%i ! = %i',m,f);
```

Matlab a déjà un grand nombre de fonctions pré-programmées. Par exemple, la fonction factorielle s'écrit *factorial(m)*. Un rapide coup d'œil dans l'aide de Matlab vous permet de savoir si la fonction désirée existe, ou si elle est à programmer (`Help` → `Function Browser`). Ne pas oublier de regarder sur le web quand une certaine fonction est nécessaire, certains l'ont sans doute déjà programmée !

5 Exercices sur les Scripts

1. Simulation d'un dé :
   - Tirer un nombre aléatoire d entre 0 et 1.
   - Multiplier d par 6 et arrondir au nombre entier supérieur D. D représente le nombre tiré au dé.
   - Jeter ce dé virtuel 100 fois et tracer en un histogramme. Ce dé vous semble-t-il juste (non pipé) ?
   - Quelle sont sa moyenne et sa variance ?
   - Écrire un script dans un fichier .m séparé qui affiche seulement l'histogramme et la moyenne.

   fonctions utiles : ceil, floor, rand, hist
2. Traitement de données hydrologiques : les précipitations à Davos sur une période de dix ans. Les données sont disponibles sur le site web du LHE.
   - Charger les données dans matlab. N'utiliser que la 7$^{\text{e}}$ colonne.
   - Tracer un histogramme à 15 barres.
   - Déterminer la fonction de distribution.
   - Calculer la moyenne et la variance.

   fonctions utiles : ecdf, load
3. Dimensionnement d'une digue pour protéger une petite commune vaudoise. Par expérience, on sait que les pluies journalières sont distribuées exponentiellement avec une moyenne de 10 mm par jour. La digue que vous avez calculée peut résister à des précipitations maximales de 70 mm par jour (ou bien 50 mm par jour pendant 2 jours). Quelle est la probabilité que la digue soit submergée ?
   - Simuler une telle pluie et compter le nombre d'événements critiques sur une période de 10 ans.
   - Pensez-vous que la digue offre une protection correcte ?

   fonctions utiles : exprnd, find, sum

6 Créer et utiliser ses propres fonctions

La syntaxe pour créer une nouvelle fonction est la suivante :

```
function [out1, out2, ...] = mafonction(in1, in2, ...)
```

On sauvegarde alors *mafonction* dans le fichier *mafonction.m* (attention au nom de ficher !) et on pourra ensuite faire appel à elle dans un script par la commande *mafonction(in1, in2,...)*.

1. Créer une fonction binom.m qui calcule le binôme $\binom{n}{p}$ et la tester. On rappelle que :
$$\binom{n}{p} = \frac{n!}{p!(n-p)!}$$
2. Créer un script binomNewton.m qui calcule $z = (\sqrt{3} + 3i)^5$ grâce au binôme de Newton. On rappelle la formule suivante :
$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$
3. Comparer le résultat obtenu avec celui des questions 1 et 2.

7 Traitement d'une série de données

Pluies à Davos

Utiliser les données de pluie à Davos.

1. Regardez le fichier de données.
2. Chargez les données.

On peut utiliser différentes fonctions pour lire un fichier, la plus universelle pour un fichier de données quelconque est [`fopen`] *suivi de* [`textscan`] *qui permet, entre autres, de sélectionner les données voulues et de sauter l'en-tête (ce qui est le cas ici), on peut aussi rapidement supprimer la première ligne et utiliser.* [`load`] *mais c'est un cas particulier*

3. La 7^{e} colonne vous donne les précipitations $pluie_i$ journalières.
4. Visualiser ces données en utilisant des points.
5. Faire un histogramme et déterminer la fonction de répartition. [`hist,ecdf`]
6. Calculer la densité de probabilité.
   - Fractionner le domaine des valeurs en 30 classes de même taille dx. Calculer le milieu de ces classes. [`max`] [`length`]
   - Compter le nombre N_i d'événements par classe et en calculer la probabilité P. [`hist`]
   - La densité de probabilité se calcule maintenant par
   $$\rho = \frac{\mathrm{d}P}{\mathrm{d}x} = \frac{N_i}{\mathrm{d}x \times \sum N_i}$$
7. Déterminer la fréquence et le temps de retour pour une précipitation de plus de 30 mm, 50 mm et de 80 mm.
8. (Si vous avez du temps) Calculer les moyennes par année. [`mean,find,==`] On fera en sorte de ne prendre en compte que les années complètes.

En hydrologie on devrait parler d'année hydrologique c'est à dire du 1er Octobre au 30 Septembre, mais par souci de temps de code nous utiliserons ici l'année calendaire (les plus à l'aise peuvent trouver un code pour sélectionner les jours de l'année hydrologique)

Chutes de neige à Chamonix

En utilisant les données de chutes de neige à Chamonix, déterminer :

1. les chutes de neige annuelles ; [`sum`]
2. la fréquence et le temps de retour pour une chute de plus de 30 cm. Attention à la définition donnée au temps de retour, il n'y a pas 365 jours de données par an.
3. faire un diagramme avec le temps de retour T fonction de la chute de neige ; [`ecdf`]

Stationnarité des données

Pour les chutes journalières de pluie et de neige ainsi que pour leurs cumuls annuels, déterminer le caractère stationnaire des séries de données.

1. Fixer un seuil S et compter le nombre $N(i)$ d'événements dépassant ce seuil parmi les i premières valeurs de votre série temporelle. [`find,cumsum`]
2. Tracer cette fonction $N(i)$ et comparer la avec une ligne droite.
3. Qu'en concluez-vous sur le caractère stationnaire de ces données ?