Session A

Premières notions

BUTS :

- Utiliser la ligne de commande
- Manier expressions et instructions
- Connaître les commandes `help`, `lookfor`, `whos`, `who`, `format`
- Utiliser l'éditeur : scripts et fonctions

MATLAB est un environnement performant de programmation scientifique et de génération de graphiques. Il peut travailler en deux modes distincts : immédiat ou différé (interpréteur ou pseudo-compilateur¹).

⇒ **sp.1** voir page 35 sur les interpréteurs et compilateurs.

A.1 Ligne de commande : mode immédiat

L'installation et le démarrage de MATLAB ne font pas l'objet de cet ouvrage. On suppose donc que sur votre système d'exploitation favori vous avez lancé le logiciel dans une version au moins supérieure ou égale à 7.0, et que vous pouvez donc taper des entrées sur la ligne de commande.

MATLAB est avant tout un interpréteur, c'est-à-dire qu'il propose à l'utilisateur une invite (prompt) à laquelle on peut répondre par une commande. Les commandes peuvent être de deux sortes :

expression ou **instruction**

L'invite est toujours donnée par un double chevron qui ne doit pas être tapé.

Exercice A.1

Question : **Tapez à l'invite les expressions suivantes, suivies de la touche entrée :**

(1) » 2*3

(3) » pi

(5) » [1 4; 6 8]

(7) » 6:-2:-3

(2) » 2*3 ;

(4) » sin(pi/2)

(6) » 6:2:20

(8) » b

¹Un compilateur peut être, moyennant finance, ajouté à MATLAB pour produire des exécutables indépendants du logiciel de base.

Une **expression** est une «formule» permettant de calculer immédiatement un résultat. Pour qu'une formule soit correcte, **il faut qu'au moment de son évaluation par l'interpréteur, tous ses éléments soient connus** (Cf. Q.F.P.-1 page 31). C'est pour cela que la dernière formule de l'exercice A.1 produit une erreur. Si ces exemples ne vous semblent pas clairs, vous pouvez dans l'ordre : lire ce qui suit, vous aider de la commande `help` (par : `help punct` ou `help sin`), de l'aide html (commande `doc`), de l'aide-mémoire MATLAB (publié aux éditions Ellipses) ou enfin de la commande `lookfor` (par `lookfor sin`).

Les points clés à retenir sont les suivants :

1. Sauf pour le deuxième et le dernier exemple, toutes les lignes vous répondent `ans` = suivi du résultat du calcul sur la ligne suivante. La variable `ans`_(a.m. p18) est créée automatiquement par MATLAB et contient la valeur de l'expression courante (la dernière calculée).
2. La touche du clavier représentant une flèche vers le haut vous permet de circuler rapidement à travers toutes vos frappes précédentes, puis de les éditer (corriger) avec les flèches gauche et droite.
Essayez également la flèche vers le haut après avoir tapé les premières lettres de la commande que vous désirez modifier...
3. La commande `format`_(a.m. p127) vous permet de changer facilement l'apparence des résultats (tapez `format long` ↵ et utilisez la remarque précédente pour revoir rapidement vos résultats).
4. La troisième expression est une variable prédéfinie en MATLAB qui vaut la constante π (`pi`_(a.m. p18)) ; il en existe bien d'autres comme `eps`_(a.m. p18).
5. La quatrième expression utilise la fonction sinus qui est une des très nombreuses fonctions définies en MATLAB. Vous pouvez accéder à la liste de ces fonctions par l'intermédiaire de l'onglet d'aide du menu de MATLAB ou l'index de l'aide-mémoire MATLAB.
6. Les deux dernières expressions construisent des matrices et des vecteurs qui sont à la base des fonctionnalités de MATLAB _{(a.m. p75)(a.m. p88)}.

Les **instructions** sont constituées, soit par un ensemble structuré d'expressions (elles ne sont pas pratiques à utiliser dans les lignes de commandes car elles sont par nature multilignes), soit par une instruction d'**affectation**.

L'affectation est une opération qui donne des valeurs à des variables qu'elle crée ou modifie. Elle est notée par le signe =²

Exercice A.2

<i>Question :</i>	Tapez à l'invite les instructions suivantes, suivies de la touche entrée :	
	<pre>(1) > a = 2*3 (3) > zz = pi (5) > zorro = [1 4 ; 6 8] (7) > g = 6 :-2 :-3</pre>	<pre>(2) > b = 2*3 ; (4) > pipo = sin(pi/2) (6) > f = 6 :2 :20 (8) > b</pre>

²mais pas := comme en Pascal ou Maple, ni == qui est l'opérateur d'égalité logique.

La principale différence est ici que la réponse (sauf pour le deuxième exemple) est donnée par le nom de la variable (une lettre seule ou suivie de lettres, de chiffres ou de soulignés) suivi du signe d'affectation = et de la réponse. De plus, si l'on retape le nom de la variable à l'invite, elle contient encore la valeur précédemment entrée qui lui a été affectée.

1. Les commandes `who`_(a.m. p279) et `whos`_(a.m. p279) permettent de connaître la liste des variables que l'on a affectées.
2. La commande `clear`_(a.m. p53) permet de supprimer des variables. Essayez `who` ↵
`clear all` ↵ et enfin `who` ↵.
3. Une variable qui n'a pas été affectée ou qui a subi un `clear`_(a.m. p53) n'existe pas. Essayez `zorro` ↵ après avoir tapé les instructions précédentes, puis après avoir exécuté un `clear zorro`.
4. Le point-virgule (;_(a.m. p218)) est simplement un *inhibiteur d'affichage*. Si la seconde instruction n'affiche pas `b`, la dernière expression le fait.

Exercice A.3

Question :

1. **A quoi sert la fonction** `lookfor` ?
2. **Est-il possible d'appliquer la fonction** `sin` **au tableau** `zorro`. **Quel est le résultat ?**
3. **Est-il possible de calculer** `zorro-1`, `zorro+zorro`, `2*zorro`, `zorro+f`, `zorro*zorro` ? **Analyser les résultats.**

A.2 Utilisation de l'éditeur : scripts et fonctions

La ligne de commande est certes pratique pour utiliser MATLAB comme une calculatrice, mais si l'on désire réutiliser plusieurs fois la même séquence de commandes, il vaut mieux utiliser l'éditeur/dévermineur intégré (Cf. Q.F.P.-2 page 31).

A l'aide de l'éditeur, on peut écrire des **scripts** et des **fonctions** ou entrer des données (Cf. Q.F.P.-3 page 31). Le principal avantage des fonctions par rapport aux scripts est de permettre une gestion des échanges de données lorsqu'elles sont appelées par d'autres fonctions ou d'autres scripts (Cf. Q.F.P.-4 page 31). On accède à l'éditeur par le menu fichier, ou les icônes «nouveau fichier» ou «rappeler ancien fichier». On peut également l'invoquer sur la ligne de commande de MATLAB par `edit` suivi du (chemin) nom du fichier à éditer.

Les fichiers contenant un script ou une fonction **doivent avoir l'extension** «.m» pour pouvoir être utilisés par la suite sur la ligne de commande; leurs noms doivent être formés d'une lettre (non accentuée) suivie de lettres (non accentuées), de chiffres ou de soulignés. Tout autre caractère pourra être, par la suite, une source d'ennuis lors de l'utilisation du fichier.

Un autre problème fréquemment rencontré est la *collision de noms* :

Il faut éviter de donner à un script ou à une fonction le nom d'une fonction système; de même, il ne faut pas donner à une variable le nom d'une fonction (Cf. Q.F.P.-5 page 31).

Donnons deux exemples très simples.

Exemple A.2.1 *Bonjour*

Invocuez l'éditeur par un des moyens suggérés plus haut et tapez dans un nouveau fichier :

```
disp('bonjour');
```

Sauvez le fichier sous le nom *essai* (l'éditeur rajoutera automatiquement l'extension *.m*). Utilisez la combinaison de touches *Alt-Tab* (ou cliquez *MATLAB* sur la barre des tâches) pour passer de l'éditeur à la fenêtre de commande *MATLAB* et tapez *essai* suivi de ↵.

Le message *bonjour* apparaîtra à l'écran.

Attention : le fichier *essai.m* doit être accessible par *MATLAB*, c'est-à-dire qu'il faut le sauver sous le répertoire proposé par *MATLAB*, ou bien émettre avant toute tentative d'exécution la commande *cd <rep>* (où *<rep>* représente le répertoire de sauvegarde) sur la ligne de commande de *MATLAB*.

Retournez sous l'éditeur ; ouvrez un nouveau fichier vide ; tapez et/ou collez à partir du fichier précédent de façon à avoir :

```
function essai(nom)
disp(['bonjour ' nom ' !']);
```

Sauvez (la fonction de sauvegarde «*Save*» du menu «*File*» de l'éditeur proposera *essai1* pour *nom*) et exécutez de la ligne de commande *essai1('jean-thierry')* ou *essai1 jean-thierry* pour voir apparaître à l'écran :

```
bonjour jean-thierry !
```

Si le script est strictement équivalent à la suite des instructions qu'on aurait pu écrire successivement sur la ligne de commande, a contrario la déclaration de fonction débute obligatoirement par une en-tête ou «*prototype d'appel*». Celui-ci comporte le nom de la fonction, mais aussi ses paramètres «*d'entrée*» et «*de sortie*» (Cf. Q.F.P.-6 page 32) :

```
function [s1, s2, ... , sn] = id(p1, p2, ... , pm)
```

id est l'identificateur de la fonction. Il est impératif que :

- le nom du fichier dans lequel la fonction se trouve soit identique au nom choisi pour remplacer *id*, suivi de l'extension *.m*;
- ce nom commence par une lettre et ne contienne que des lettres ordinaires, des chiffres et des soulignés. **En particulier cela INTERDIT les caractères accentués, les blancs et les signes de ponctuation.**

Les paramètres *s1, s2, ... , sn* et *p1, p2, ... , pm* permettent à un processus d'être répété plusieurs fois avec des objets différents. On distingue deux sortes de paramètres :

- les paramètres «*d'entrée*» *p1, p2, ... , pm* (situés après l'identificateur) ;
- les paramètres «*de sortie*» *s1, s2, ... , sn* (situés à gauche du signe =).

Les paramètres d'entrée de la fonction sont ceux dont les valeurs sont connues avant son utilisation et nécessaires à son exécution ; les paramètres de sortie sont ceux que l'on voudra faire calculer par la fonction.

⇒ **sp.2** voir page 35 sur les paramètres des fonctions.

A vrai dire, MATLAB n'impose pas la coïncidence du nom de fichier et de l'identificateur de la fonction. Cependant, comme **c'est le nom du fichier qui est utilisé au moment de l'appel**, il serait bien maladroit et donc fortement déconseillé de donner des noms différents à ceux-ci.

Après l'en-tête, la fonction se poursuit par quelques lignes de commentaires qui sont particulièrement importantes pour sa documentation : la première de ces lignes est prise en compte par la commande `lookfor`. De plus, toutes ces lignes jusqu'à la première ligne blanche (ou la première ligne de code) sont affichées par la commande `help` de MATLAB. Enfin vient le *corps* de la fonction qui contient la «recette» exécutable, c'est-à-dire les commandes qui vont effectuer les actions et éventuellement calculer les paramètres de sortie.

Un autre avantage des fonctions est qu'il est possible de mettre plusieurs fonctions dans un même m-fichier (.m) (Cf. Q.F.P.-7 page 32). L'utilisation d'un résultat fourni par une fonction dans une autre fonction ou un script est décrite dans la Q.F.P.-8 (page 32). Une erreur classique de débutant est liée à la mauvaise gestion des paramètres de sortie d'une fonction lorsqu'il y a plus d'un paramètre de sortie (Cf. Q.F.P.-9 page 33).

Il existe deux types particuliers de fonctions qui peuvent parfois se substituer aux fonctions classiques décrites ci-dessus.

⇒ **sp.3** voir page 37 sur les fonctions «anonymes», «en ligne» et imbriquées.

La connaissance des fonctions et des scripts acquise dans cette session permettra d'utiliser les m-fichiers fournis comme solutions des exercices de cet ouvrage (Cf. Q.F.P.-10 page 33).

Exercice A.4

Question :	Écrivez une fonction signe calculant le signe d'un nombre (-1, 0 ou 1 suivant que le nombre est négatif, nul ou positif).
Aide :	<p>Pour effectuer un choix consécutif à un test, la structure de contrôle <code>if...elseif...else...end</code> peut être utile (Cf. Q.F.P.-11 page 33).</p> <p>Attention : «afficher le résultat» n'est pas identique à «calculer le résultat». Il faut que l'écriture <code>a=3*signe(sin(3*pi/2))</code> donne à <code>a</code> la valeur <code>-3</code> et non un message d'erreur (Cf. Q.F.P.-8 page 32).</p>

Exercice A.5

Question :	<p>Résolvez les équations du premier et second degré avec deux fonctions d'en-têtes respectives : fonction <code>x = equ1(a, b)</code> et fonction <code>x = equ2(a, b, c)</code> en ne calculant que les racines réelles et en réfléchissant aux problèmes suivants :</p> <ol style="list-style-type: none"> 1. Quel est le domaine de validité de vos fonctions ? 2. Un paramètre de sortie unique <code>x</code> est-il suffisant ? 3. Peut-on faire en sorte que l'appel <code>equ2(a, b)</code> soit valide et équivalent à <code>equ1(a, b)</code> ?
Aide :	<ol style="list-style-type: none"> 1. Voyez la session L, page 233 ; 2. <code>x</code> peut être un vecteur. <code>x(1)</code> (et s'il le faut) <code>x(2)</code> contiendront les racines ; 3. testez la valeur de <code>nargin</code>_(a.m. p165).

Exercice A.6

Question :	A l'aide de l'opérateur «:»_(a.m. p88) (help colon), écrivez une fonction des paramètres a, b, n, qui calcule un vecteur à n composantes à intervalles réguliers entre a et b.
Aide :	Il s'agit du problème des poteaux et des intervalles : n poteaux $\Leftrightarrow$ n-1 intervalles de longueur $(b-a)/(n-1)$... Voir <code>linspace</code> _(a.m. p178) pour une version MATLAB. (N.B. n'appellez pas votre fonction <code>linspace</code>)

A.3 QCM

Chaque affirmation est vraie ou fausse. L'important est de comprendre pourquoi.

Exercice A.7

Q.C.M.	
Question :	On écrit une fonction dont l'en-tête est : <code>function total = somme(x, y)</code>
Choix :	<ol style="list-style-type: none"> 1. <code>total</code> est un paramètre d'entrée ; 2. <code>total</code> est un paramètre de sortie ; 3. <code>x</code> est un paramètre de sortie ; 4. <code>somme</code> est le nom de la fonction ; 5. <code>somme.m</code> est le nom du fichier contenant la fonction.

Exercice A.8

Q.C.M.	
Question :	On considère l'instruction : <code>v = [1:5 ; 3:0.5:5]</code>
Choix :	<ol style="list-style-type: none"> 1. <code>v</code> est un vecteur à 5 éléments ; 2. <code>v</code> est une matrice à 2 lignes et 5 colonnes ; 3. la syntaxe de l'expression donnant <code>v</code> est incorrecte ; 4. <code>v(2,3)</code> est égal à 4 ; 5. <code>v(3)</code> est égal à 2.

Questions fréquemment posées sur la session A

QFP-1	
Question :	Dois-je comme en C ou en Pascal déclarer mes variables ?
Réponse :	<i>En général non. La seule restriction est qu'une variable doit être initialisée, c'est-à-dire affectée lors de sa première utilisation (ou être un paramètre d'entrée d'une fonction, Cf. page 35). A l'intérieur d'une fonction, les nouvelles variables définies sont locales (Cf. page 35) : elles n'ont aucun rapport avec des variables de même nom qui pourraient avoir été définies dans d'autres fonctions ou scripts et perdent leur existence à la fin de chaque exécution de la fonction.</i>

QFP-2	
Question :	Pourquoi est-il préférable d'utiliser l'éditeur à la ligne de commande ?
Réponse :	<i>Pour pouvoir réutiliser sans peine le travail effectué.</i>

QFP-3	
Question :	Qu'est-ce qu'une fonction ? Qu'est-ce qu'un script ?
Réponse :	<i>Il arrive souvent qu'une certaine suite d'énoncés doive être répétée à plusieurs endroits dans un programme. Pour éviter au programmeur de recopier ces énoncés, la plupart des langages de programmation offrent la possibilité de créer des sous-programmes. Ce moyen autorise à utiliser un nom (identificateur) comme abréviation chaque fois que cette suite d'énoncés apparaît. MATLAB propose deux méthodes à cette fin, la première permet de définir des noms pour des programmes entiers : les scripts ; la seconde des noms pour des morceaux de programmes réutilisables : les fonctions.</i>

QFP-4	
Question :	J'ai écrit une fonction. Quand puis-je l'utiliser ?
Réponse :	<i>Partout et en particulier à l'intérieur d'une autre fonction.</i>

QFP-5	
Question :	Comment éviter les collisions de noms ?
Réponse :	<i>MATLAB définit plusieurs milliers de fonctions dans des m-fichiers ou dans son noyau ; en conséquence, tout nom appartenant à la langue anglaise et désignant une action courante a de bonnes chances d'être utilisé. Ce n'est généralement pas grave si vous n'utilisez pas la fonctionnalité MATLAB, votre fonction ayant généralement la préséance. Cependant, en cas de doute, utilisez la commande <code>which</code> (a.m. p279) qui vous dira quelle est la fonction que vous utilisez réellement.</i>

QFP-6	
Question :	Je désire écrire une fonction ; quel doit-en être l'en-tête ?
Réponse :	<p><i>Lors de l'écriture d'une fonction, il faut toujours se poser les deux questions suivantes :</i></p> <ul style="list-style-type: none"> – <i>Les paramètres d'entrée sont-ils suffisants (et nécessaires) pour effectuer le calcul ?</i> – <i>Les paramètres de sortie permettent-ils à l'utilisateur de la fonction de prendre toute décision relative à la poursuite du programme ?</i>

QFP-7	
Question :	Je trouve un peu lourd d'avoir de nombreux fichiers pour mes fonctions. N'y a-t-il pas moyen d'en mettre plusieurs dans un même fichier ?
Réponse :	<p><i>Oui, un m-fichier contenant une fonction peut en contenir plusieurs, mais seule la première (dont le nom est celui du fichier) est accessible de l'extérieur. Les autres sont locales et peuvent seulement s'appeler entre elles.</i></p> <p><i>Il existe aussi deux autres types de fonctions en MATLAB : les fonctions <code>inline</code>_(a.m. p162) et des fonctions «anonymes»_(a.m. p19) (à partir de la version 7). Elles sont utiles pour définir des fonctions simples et courtes dont la définition tient en une ligne : elles peuvent être ainsi incluses n'importe où dans un script ou une fonction. Les fonctions anonymes ont la capacité très précieuse de capturer les valeurs des variables locales (voir page 37).</i></p>

QFP-8	
Question :	Comment permettre à une fonction de retourner des résultats utilisables dans la suite d'un programme ?
Réponse :	<p><i>Dans le corps de la fonction <code>[s1, ..., sn]=fun(p1, ..., pm)</code>, il suffit de trouver chaque paramètre de sortie <code>si</code> ($1 \leq i \leq n$) à gauche d'un signe = d'affectation (et d'être certain que le flot des calculs passera toujours par ces lignes).</i></p> <p><i>Il ne faut pas confondre «afficher un résultat» et «retourner un résultat».</i></p>

QFP-9	
Question :	Ma fonction <code>moncalculprefere</code> devrait produire deux résultats et je n'en obtiens qu'un ou je n'en vois aucun. Que se passe-t-il ?
	Suite page suivante.