Initiation à Matlab

FIP TI Santé 1re année – 2018–2019

Vincent Mazet (C219, vincent.mazet@unistra.fr)

Ce document est le support des deux séances d'initiation au logiciel Matlab (dans sa version R2016b). Il contient une présentation rapide du logiciel et des exercices (identifiés par le symbole ✎).

Sommaire

1 Présentation de Matlab

Matlab (*matrix laboratory*) est un logiciel de calcul numérique développé par la société MathWorks depuis 1984 et spécialisé dans le calcul matriciel. De nombreuses bibliothèques (*toolboxes*) complètent le noyau de base. Les domaines d'applications sont nombreux : mathématiques appliquées, statistiques, analyse numérique, automatique, traitement du signal et des images, etc. Des produits concurrents et libres existent également, comme Scilab ou Octave.

Matlab est un interpréteur, c'est-à-dire qu'il exécute les instructions directement, sans les compiler en langage machine (comme c'est le cas pour le C). Les instructions sont écrites dans la fenêtre de commande (*command window*) ou regroupées dans un fichier (qui portera l'extension .m, par exemple : monprogramme.m, et qu'on exécute en tapant simplement « `monprogramme` » dans la fenêtre de commande).

L'interface de Matlab est composée de plusieurs parties (que l'on peut afficher ou non à l'aide du menu *Desktop*) :

- la fenêtre de commande (*command window*) où l'on tape les instructions ;
- le répertoire courant (*current folder*) qui affiche le contenu du dossier courant ;
- l'espace de travail (*workspace*) qui contient la liste des variables existantes ;
- l'éditeur (*editor*) qui permet de créer des scripts ou des fonctions Matlab.

Matlab possède une aide très complète. Pour obtenir des détails sur les instructions et les fonctions, tapez « `help` *`nom_de_la_fonction`* » pour une aide rapide ou « `doc` *`nom_de_la_fonction`* » pour avoir une aide détaillée. Pour chercher une fonction particulière, utilisez la commande `lookfor`. La commande `demo` donne accès à un grand nombre de démonstrations du logiciel.

Avant de commencer le TP, créez un répertoire de travail (sur le disque dur ou une clé USB) dans lequel vous stockerez vos fichiers.

2 Variables et matrices

Avec Matlab, une variable est créée dès qu'elle est utilisée : il n'est donc pas nécessaire de la déclarer comme c'est le cas dans d'autres langages de programmation. Si une variable du même nom existe déjà, son contenu est remplacé par la nouvelle valeur, c'est pourquoi il convient de ne pas créer une variable qui porte le même nom qu'une fonction !
Le nom d'une variable est limité à 63 caractères, ceux-ci étant des lettres, chiffres ou tirets bas, mais le premier caractère doit toujours être une lettre. De plus, Matlab est sensible à la casse, c'est-à-dire qu'il différencie les majuscules et minuscules (`A` et `a` ne sont pas la même variable).
Sachez également que l'exécution d'un calcul peut être arrêtée à tout moment avec la combinaison de touche Ctrl+C.

2.1 Calculs sans variable

Matlab, à l'instar de n'importe quelle calculatrice, peut effectuer une opération en la tapant directement dans la fenêtre de commande. Le signe ≫ indique que Matlab est prêt à recevoir une instruction, qui sera exécutée en tapant sur *Entrée*. Par exemple :

```
≫ 1+1
```

renvoie 2 (ouf !). Dans la suite de ce document, le signe ≫ ne sera plus indiqué.

2.2 Créer des variables et des matrices

Les scalaire, vecteurs et matrice suivants

$$a = 2{,}23, \qquad b = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}, \qquad c = \begin{pmatrix} 10 \\ 20 \\ 30 \\ 40 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

sont rentrés ainsi :

```
a = 2.23;
b = [1 2 3 4];
c = [10 ; 20 ; 30 ; 40];
D = [1 2 3 ; 4 5 6];
```

L'espace et le point-virgule entre les crochets sont des séparateurs qui font respectivement passer à la colonne ou à la ligne suivante.

✎ **1** Créez la matrice

$$A = \begin{pmatrix} 12 & 56 & 32 \\ 45 & 65 & 78 \end{pmatrix}$$

Pour éviter l'affichage de résultats lors de l'exécution d'instructions, il suffit de terminer celles-ci par un point-virgule.

2.3 Accéder aux éléments d'une matrice ou d'un vecteur

Pour récupérer ou affecter un élément d'une matrice, il suffit d'écrire la variable suivie des coordonnées de l'élément entre parenthèses. Ainsi, `D(1,3)` vaut 3. Attention : il est important de noter que les éléments sont indexés à partir de 1, ce qui nécessitera quelques précautions, notamment lors du développement de programmes faisant intervenir le temps (qui, très souvent, commence à 0).
Pour accéder à une colonne ou une ligne d'une matrice, il suffit d'utiliser l'opérateur `:`. Par exemple, `D(:,3)` permet d'avoir tous les éléments de la troisième colonne.

✎ **2** Affichez le 2e élément de la diagonale de la matrice D précédente, puis la 1re ligne de D.

Il existe deux autres manières d'accéder à plusieurs éléments d'un vecteur ou d'une matrice :

- la première possibilité est d'utiliser un vecteur contenant les indices des éléments souhaités. Ainsi, `c([2 4])` renvoie `[20 40]` ;
- la seconde possibilité est d'utiliser un vecteur (ou une matrice) booléen ; on rappelle qu'une variable booléenne prend seulement deux valeurs : vrai (`true` en Matlab) ou faux (`false`). Dès lors, `true` permet de sélectionner l'élément correspondant. Le vecteur booléen doit donc être de même taille que la variable considérée. Ainsi, `c([false true false true])` renvoie `[20 40]` ;

✎ **3** Créez le vecteur

$$e = \begin{pmatrix} 11 & 22 & 33 & 44 & 55 & 66 & 77 & 88 & 99 \end{pmatrix}.$$

En utilisant les deux manières précédentes, créez un vecteur contenant les éléments 1, 3, 4, 5 et 8 de e.

✎ **4** Remplacer le cinquième élément du vecteur e par 101.

2.4 Opérateur d'énumération

Pour créer un vecteur dont les éléments constituent une suite arithmétique (suite de nombres de a à b par pas de p), on peut utiliser l'opérateur d'énumération :

valeur initiale`:`*pas*`:`*valeur finale*

le pas étant positif ou négatif. Ainsi, `f = 2:0.5:4` correspond au vecteur $f = \begin{pmatrix} 2 & 2{,}5 & 3 & 3{,}5 & 4 \end{pmatrix}$.

✎ **5** Créez les vecteurs suivants :

1. de -10 à 10 :
2. de 10 à 0 :
3. de 0 à 100 par pas de 5 :
4. de 5 à 1 par pas de $-0{,}1$:

2.5 Opérations

Les opérations classiques résumées ci-dessous peuvent être appliquées sur les variables scalaires, vectorielles ou matricielles.

Opérations matricielles	
`+`	addition
`-`	soustraction
`*`	multiplication
`/`	division
`^`	puissance

Opérations élément par élément	
`.*`	multiplication élément par élément
`./`	division élément par élément
`.^`	puissance élément par élément

✎ **6** On considère les variables suivantes :

$$x = 5, \quad y = \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}.$$

Calculez $2x$, $A+B$, Ay, A^2, $2y$, ainsi que les carrés des éléments de y et de A.

✎ **7** Testez les instructions suivantes :

```
[1 2 3 4]*5
[1 2 ; 3 4]^2
[1 2 ; 3 4].^2
sin(1:10)
exp(j*2*pi*(2:0.5:100))
1./[10:20]
```

2.6 Vecteurs logiques

L'une des forces de Matlab est de générer des vecteurs de booléens (ou vecteurs logiques) à partir des opérateurs relationnels et logiques résumés ci-dessous.

Opérateurs relationnels et logiques de Matlab

`==`	égalité
`~=`	inégalité
`<`	strictement plus petit que
`>`	strictement plus grand que
`<=`	plus petit ou égal
`>=`	plus grand ou égal
`|`	ou logique
`&`	et logique

Considérons par exemple le vecteur $a = \begin{pmatrix}6 & 7 & 8 & 9\end{pmatrix}$. L'instruction `a >= 8` produira alors un vecteur de même taille que a dont chaque élément $i \in \{1,\ldots,4\}$ est égal à 0 (si $a_i < 8$) ou 1 (si $a_i \geq 8$). Le vecteur ainsi produit peut être enregistré dans une variable `b` pour obtenir finalement l'instruction suivante :

```
b = (a >= 8)
```

(les parenthèses sont inutiles pour Matlab mais améliorent la lisibilité).

✎ **8** Créez un vecteur x de taille 8 contenant des entiers aléatoires entre 0 et 10 (fonction `randi`) puis affichez les vecteurs logiques correspondant :

— aux éléments strictement supérieurs à 5 ;
— aux éléments nuls ;
— aux éléments différents de 8 ;
— aux éléments égaux à 1 ou 7.

On peut alors accéder aux éléments vérifiant la condition en utilisant le résultat de la même manière que dans la section 2.3 : l'instruction `a(a>=8)` renvoie donc le vecteur $\begin{pmatrix}8 & 9\end{pmatrix}$.

✎ **9** Reprendre l'exercice précédent et afficher seulement les éléments vérifiant les conditions.

✎ **10** Générez un vecteur t allant de -10 à 10 avec un pas de 1. À l'aide de t, créez le vecteur x nul partout sauf lorsque t est égal à 0, et égal à 1 pour $t = 0$.

✎ **11** Générez un vecteur t allant de -10 à 10 avec un pas de 1. Faites en sorte que les éléments négatifs de t soient mis à 11.

3 Programmation

Comme dans tout langage de programmation, il est très important d'*indenter* son code ! Le programme est ainsi plus lisible, la syntaxe est plus facile à vérifier et cela oblige à bien construire le programme en blocs (conditions, boucles, ...).
Par ailleurs, prenez l'habitude de commenter vos programmes (à l'aide du caractère `%`) afin d'expliquer en français ce qu'il est censé faire ainsi que les différentes étapes du code.

3.1 Script

Un script Matlab est un fichier portant l'extension « .m » qui contient une suite d'instructions telles qu'on aurait pu en taper dans la fenêtre de commande. Pour créer un script avec l'éditeur de Matlab, cliquez sur le menu *File*, *New* puis *Script*. Pour exécuter un fichier, il suffit simplement de taper son nom (sans l'extension) dans l'invite de commande ou de cliquer sur l'icône  . Par conséquent, votre fichier ne doit pas avoir le même nom qu'une autre fonction (comme « exp.m » ou « filter.m ») !

3.2 Condition `if`

La condition `if` teste une expression logique et exécute une suite d'instructions si cette expression est vraie. Le mot-clé optionnel `else` permet quant à lui d'exécuter une autre suite d'instructions si l'expression est fausse. La syntaxe est la suivante :

```
if expression à tester
  ...suite d'instructions à exécuter si l'expression est vraie...
else
  ...suite d'instructions à exécuter si l'expression est fausse...
end
```

Les opérateurs relationnels et logiques ont été donnés section 2.6. La seule différence concerne les opérateurs | et `&` qui dans ce cas peuvent être remplacés par les équivalents scalaires || et `&&`.

✎ **12** Définissez deux nombres quelconques a et b. Écrivez un programme indiquant quel est le plus grand de ces deux nombres (la fonction `disp` permet d'afficher un texte). Tenez compte, dans un deuxième temps, du cas de l'égalité entre les deux nombres (utilisez `elseif`).

3.3 Boucle `for`

La boucle `for` répète des instructions un nombre déterminé de fois. La syntaxe est la suivante :

```
for indice = valeur initiale:pas:valeur finale
  ...suite d'instructions...
end
```

✎ **13** Avec une boucle `for`, affichez les entiers de 0 à 10.

✎ **14** Avec une boucle `for`, affichez les nombres pairs de 100 à 90.

✎ **15** Créez un vecteur x de taille 10 contenant des entiers aléatoires entre 0 et 10 (fonction `randi`) puis calculez la somme de ses éléments deux à deux.

✎ **16** En Matlab, dans la mesure du possible, n'utilisez par la boucle `for`. Il existe souvent une manière plus rapide en travaillant en vectoriel. Par exemple, pour calculer les carrés d'un vecteur x, le programme

```
y = x.^2;
```

est bien plus rapide que le suivant :

```
y = zeros(length(x),1);
for i = 1:length(x)
  y(i) = x(i).^2;
end;
```

Testez les deux programmes ci-dessus et vérifiez que le premier est plus rapide. Utilisez les fonctions `tic` et `toc` pour mesurer le temps de calcul de chacun. Si les temps sont trop faibles pour être mesurés, il est nécessaire d'exécuter plusieurs fois le même programme.

3.4 Fonction

Une fonction Matlab est un script qui peut recevoir des arguments d'entrée, et peut retourner des arguments de sortie. Pour cela, il suffit d'ajouter l'instruction suivante au début du script :

```
function [s1,s2,...] = mafonction(e1,e2,...)
% Description de la fonction
```

Notez que la fonction doit avoir le même nom que le fichier ! `s1`, `s2`, ... sont les variables de sortie et `e1`, `e2`, ... sont les variables d'entrée ; elles peuvent être des scalaires, vecteurs, matrices, etc.

✎ **17** Écrivez une fonction qui calcule la norme euclidienne $\sqrt{x_1^2 + x_2^2}$ d'un vecteur $x = (x_1,x_2)$. Donnez une description de la fonction en commentaires (tapez `help fonction` pour vérifier que vous pouvez lire votre description).

3.5 Amélioration et débogage du code

Matlab possède un analyseur de code qui peut vous aider à écrire un code correct et efficace. Ainsi, les portions de code pouvant être améliorées sont soulignées en orange, et celles contenant des erreurs de syntaxe sont soulignées en rouge. En passant la souris sur les portions soulignées, des info-bulles permettent d'identifier le problème.

✎ **18** Ouvrir le fichier sommecumulee.m et indiquez sur le code ci-dessous les parties soulignées et le message associé.

```
% Calcul de la somme cumulée des éléments d'un vecteur
% vincent.mazet@unistra.fr, 08/2012

% Note : comme tout bon programme, pensez à :
%   - indenter votre code
%   - mettre des commentaires pour expliquer ce que vous faites

clear all;                      % Supprime toutes les variables

% Demande à l'utilisateur la taille du vecteur x
N = input('Quel est le nombre d''éléments (entier positif) ? ');

% Vérifie que N est bien un entier strictement positif
if (N < 1) || (N ~= floor(N)),
    % Si ce n'est pas le cas, génère une erreur
    error('N doit être un entier strictement positif.');
end;

% Crée le vecteur [1 2 3 ... N]
x = 1:N

% Calcule la somme cumulée
y = zeros(1,N);                 % Vecteur des sommes cumulées
y(1) = x(1);
for n = 2:N,
    y(n) = y(n) + x(n);
    end;

% Affichage du résultat
disp('Vecteur x :');
disp(x);
disp('Somme cumulée des éléments de x :');
disp(y);
```

✎ **19** Corrigez le script sommecumulee.m.

Il se trouve que même si le code est exempt d'erreurs, il ne fasse pas ce que l'on souhaite. Par exemple, le fichier sommecumulee.m est censé calculer la somme cumulée d'un vecteur : vous pouvez vérifier que ce n'est pas le cas en l'exécutant.
Dans ce cas, on peut ajouter des points d'arrêt (*breakpoints*) dans le code avec l'instruction `keyboard` : le programme s'arrête alors temporairement à chaque point d'arrêt (une flèche verte indique la ligne). La commande `dbstep` (icône ) permet d'exécuter la ligne courante et passer à la suivante ; la commande `dbquit` (icône ) arrête complètement le programme.

✎ **20** Débugguez le code sommecumulee.m en utilisant les points d'arrêt.

4 Graphiques

Matlab permet de représenter des données sous diverses formes, et en particulier de tracer des courbes avec la fonction `plot`.

4.1 Représenter une courbe

`plot(x,y)` trace `y` en fonction de `x`. En fait, cette commande relie les points dont les abscisses sont contenues dans le vecteur `x` et les ordonnées dans le vecteur `y`.
D'autres types de points, de ligne et de couleur peuvent être spécifiés à l'aide d'un troisième argument : par exemple `plot(x,y,'r+:')` trace une ligne pointillée (`:`) rouge (`r`) où les données sont représentées par une croix (`+`) ; voir l'aide pour plus de détails.

✎ **21** Définir les vecteurs $x = \begin{pmatrix}1 & 2 & 3 & 4 & 5\end{pmatrix}$ et $y = \begin{pmatrix}23 & 54 & 12 & 85 & 45\end{pmatrix}$. Tracer y en fonction de x puis x en fonction de y : comparez. Vous pouvez éventuellement modifier les types de points et de lignes.

✎ **22** Tracez la fonction $f(x) = x^2$ sur $[-10,10]$ (créez avant tout le vecteur des abscisses puis le vecteur des ordonnées).

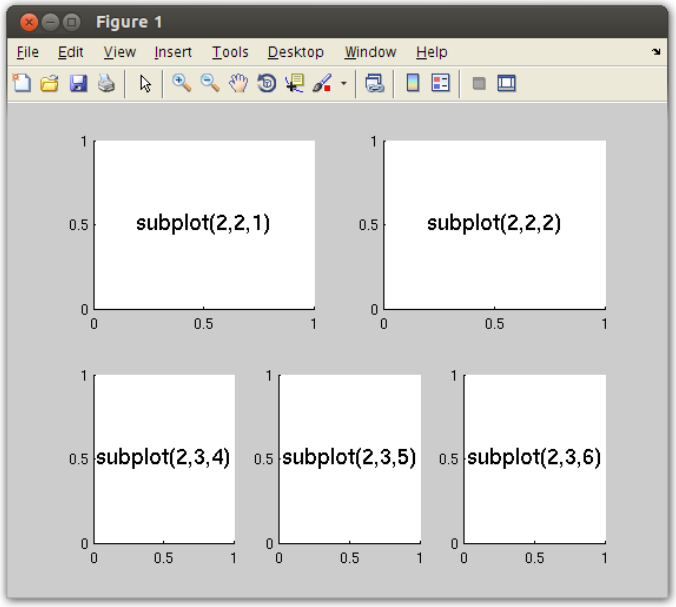
4.2 Curseur de données

Un curseur de données (*data cursor*) permet d'afficher les coordonnées des points des courbes ou les coordonnées et la couleur des pixels d'une image. Ces informations peuvent être exportées dans l'espace de travail, sauvegardées ou imprimées en même temps que la figure, etc.
Pour utiliser le curseur de données, cliquez sur l'icône dans la barre d'outils ou sélectionnez *Data Cursor* du menu *Tools*. Cliquez ensuite sur une courbe (ou tout autre objet graphique) pour afficher les coordonnées du point sélectionné. Vous pouvez déplacer le marqueur à l'aide des touches fléchées ou de la souris.

✎ **23** Tester le curseur de données sur la courbe tracée précédemment.

4.3 Représenter plusieurs courbes

Pour pouvoir afficher plusieurs courbes superposées (sur le même graphe), on peut utiliser la commande `hold('on')`.
Pour pouvoir afficher plusieurs courbes dans une même fenêtre mais sur plusieurs graphes, il faut utiliser la commande `subplot(L,C,i)` qui découpe la fenêtre en un tableau de `L` lignes et `C` colonnes et sélectionne la `i`$^{\text{e}}$ case pour afficher une courbe (cf. figure suivante).
Enfin, l'instruction `figure` permet d'ouvrir une nouvelle fenêtre graphique.



✎ **24** On considère les fonctions $f(t) = \sin(t)$ et $g(t) = \cos(t)$.

1. Tracez f et g sur le même graphe

2. Tracez f et g sur deux graphes distincts d'une même fenêtre

3. Tracez f et g dans deux fenêtres

4.4 Titre et légendes des axes

Les fonctions `title`, `xlabel` et `ylabel` permettent de donner un titre au graphique et des légendes aux abscisses et aux ordonnées.

✎ **25** Testez ces fonctions sur l'un des graphes précédents.

4.5 Catalogue de graphiques

Un grand nombre de types de graphiques sont disponibles sous Matlab (courbes, histogrammes, camemberts, etc.). On peut également représenter très rapidement une variable selon l'une de ces représentations. Pour cela, cliquez sur une variable dans l'espace de travail de Matlab (*Workspace*) puis choisissez l'une des représentations disponibles dans la liste *Catalog plot* (onglet *Plots* de la barre d'outils).

✎ **26** Amusez-vous avec cette fonctionnalité sur un exemple.

5 Fichiers de données

Matlab permet d'enregistrer et de charger des fichiers de données. Les fichiers MAT sont enregistrés sous un format propre à Matlab qui permet de compresser les données. En revanche, les fichiers ASCII sont plus gros mais très répandus car utilisables par tous les logiciels de traitement de données. Les fonctions pour enregistrer ou charger des données sont répertoriées ci-dessous.

	Pour les Fichiers MAT	Pour les Fichiers ASCII
Enregistrement	`save('fichier','var1','var2',...)`	`dlmwrite('fichier',M)`
Chargement	`load('fichier')`	`M = dlmread('fichier')`

`fichier` est bien sûr le nom du fichier, `var1`, `var2`, ... sont les variables à enregistrer, `M` est la matrice à enregistrer ou à charger.

✎ **27** Le fichier enzo.csv est un fichier ASCII contenant le suivi d'un manchot royal entre le 13/12/2004 et le 21/03/2005 avec une balise Argos [1]. Il contient trois colonnes qui correspondent respectivement à un indice temporel, la latitude et la longitude. Chargez le fichier puis affichez l'itinéraire que le manchot royal a effectué (c'est-à-dire la latitude en fonction de la longitude).

6 Simulink

Simulink est un logiciel dépendant de Matlab qui propose une interface graphique pour modéliser, simuler et analyser des systèmes dynamiques sous forme de schémas-bloc. Tapez « `simulink` » dans la fenêtre de commande de Matlab pour lancer Simulink.

Les blocs sont regroupés en librairies disponibles dans la fenêtre principale de Simulink. Après avoir créé un nouveau système (Menu *File*, *New*, *Model*), il suffit ensuite de connecter les blocs en traçant des lignes entre eux (ou en cliquant sur les blocs tout en maintenant le bouton *Ctrl* enfoncé). Les blocs les plus communs que vous aurez à utiliser sont résumés à la fin du document. Éventuellement, le bloc *Subsystem* de la librairie *Ports & Subsystems* vous permettra de créer des sous-systèmes à la manière d'une sous-fonction d'un programme Matlab. Les paramètres de chaque bloc (valeur du gain, fréquence de la sinusoïde, nombre de signaux d'entrée, etc.), sont accessibles en double-cliquant sur celui-ci.

Pour lancer une simulation, cliquez sur le menu *Simulation* puis *Start*. Les paramètres de simulation sont accessibles par le menu *Simulation*, *Simulation parameters* puis sur *Solver* dans la fenêtre qui s'affiche.

✎ **28** Représentez une sinusoïde en connectant un bloc *Sine Wave* à un bloc *Scope* (double-cliquez sur *Scope* pour afficher le graphe). Il est d'ailleurs conseillé d'ouvrir la fenêtre des paramètres (icône) puis de cliquer sur l'onglet *History* et de décocher la case *limit data points to last* pour afficher tout le signal. Modifiez ensuite les paramètres de la sinusoïde.

1. Source : http://argonautica.jason.oceanobs.com/html/argonautica/.

✎ **29** Affichez la réponse indicielle (réponse à un échelon) du système de fonction de transfert :

$$H(s) = \frac{1}{2s+1}.$$

✎ **30** Utilisez le bloc *Mux* pour afficher sur le même graphique l'entrée et la sortie du système.

7 Interfaces graphiques

Matlab permet de réaliser des programmes dotés d'une interface graphique. Cela évite d'utiliser la ligne de commande et peut donc simplifier l'utilisation de votre programme par les futurs utilisateurs.
Nous allons réaliser une interface très simple permettant d'afficher une sinusoïde dont la fréquence est définie par l'utilisateur. À partir d'un script vide, procédez en suivant les étapes suivantes. Testez votre code après chaque étape pour vérifier l'absence de bug.

✎ **31** Ajouter l'instruction `function` *`nom`* pour définir le nom de votre programme puis l'instruction `figure` pour créer une fenêtre.

✎ **32** Utilisez ensuite l'instruction `axes` pour y ajouter un graphe. Par exemple :

```
axes('Units','pixels', 'Position',[50 50 450 200]);
```

créé un graphe positionné en (50,50) de largeur 450 et de hauteur 200. Ces coordonnées sont définies en pixels.

✎ **33** Ajoutez ensuite des contrôles avec l'instruction `uicontrol`.

- le mot-clé `Style` de cette instruction permet de définir le type de contrôle. L'exemple de la figure 1 utilise une zone de texte (`text`), une zone éditable (`edit`) et un bouton (`pushbutton`) ;
- le mot-clé `String` définit le texte à afficher ;
- dans le cas du bouton, il faut également définir l'action à effectuer lorsque l'utilisateur clique sur le bouton grâce au mot-clé `Callback`. Par exemple :

  ```
  uicontrol('Style','pushbutton', 'String','Dessiner', 'Callback',@draw);
  ```

  permet d'affecter la fonction « draw » au bouton.
- Enfin, la fonction `uicontrol` retourne l'identifiant de celui-ci, ce qui permet notamment d'avoir accès aux propriétés du contrôle. Ainsi, pour pouvoir utiliser ultérieurement la valeur de la fréquence entrée par l'utilisateur, veillez à enregistrer dans une variable l'identifiant de la zone éditable.

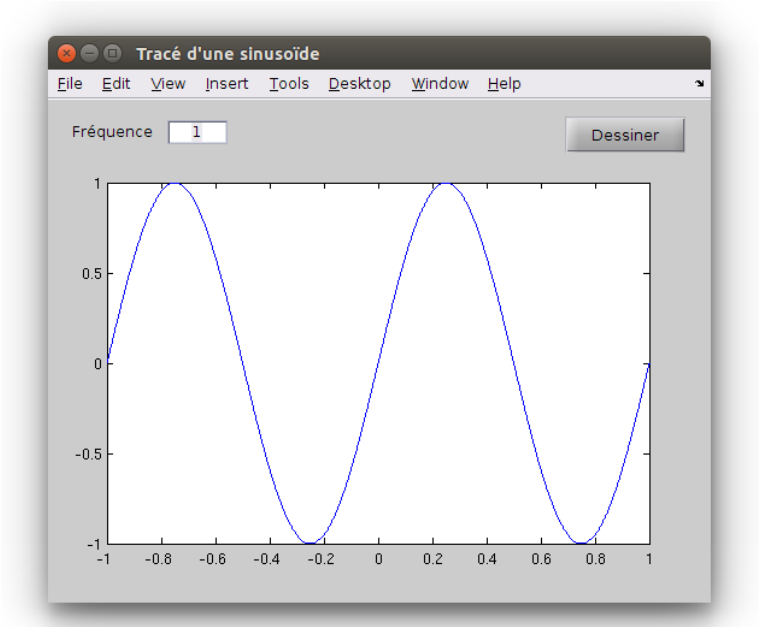


FIGURE 1 – Modèle pour la réalisation d'une interface graphique simple.

✎ **34** Tapez le code ci-dessous pour définir la fonction « draw » qui s'exécute lorsqu'on clique sur le bouton :

```
function draw(source,eventdata)
  f = str2double( get(freq, 'String') );  % Fréquence
  t = -1:1e-3:1;                          % Vecteur des abscisses
  y = sin(2*pi*f*t);                      % Valeur de la sinusoïde
  plot(t,y);                              % Affichage de la sinusoïde
end
```

L'instruction `get(freq, 'String')` permet de récupérer la chaîne de caractères entrée par l'utilisateur dans la zone de texte nommée « freq ». Cette chaîne de caractère est ensuite convertie en nombre avec `str2double`.

✎ **35** Votre programme peut également prendre en argument d'entrée une valeur pour la fréquence. Modifiez la première ligne pour intégrer cette variable (que nous noterons `f`). Modifiez également le texte de la zone éditable pour y afficher la valeur de `f`. À la toute fin du programme, appelez la fonction « draw » pour tracer automatiquement la sinusoïde sans avoir à cliquer que le bouton.

✎ **36** Enfin, il peut être intéressant de laisser l'utilisateur libre de définir ou non la fréquence lors de l'appel du programme. Dans ce cas, il faut affecter une valeur par défaut à `f` si le nombre d'arguments d'entrée (`nargin`) est nul.

✎ **37** Modifiez votre programme pour inclure d'autres fonctionnalités, par exemple : la vérification de saisies correctes, des options pour modifier la couleur ou le tracé de la courbe, etc.

8 Exercices complémentaires

✎ 38 Manipulation de vecteur

On considère le vecteur ligne x de longueur 100 dont les 10 premiers éléments sont égaux à un et les 90 suivants à zéro.

1. Créez ce vecteur.
2. Créez ensuite le vecteur $y = [x\ x\ x]$ par concaténation.
3. Ce vecteur correspond à un signal créneau ; affichez ce signal (avec l'instruction `stem`).

✎ 39 Manipulation de matrice

On considère la matrice suivante :

$$A = \begin{pmatrix} 3 & 9 & -1 \\ 6 & 0 & 7 \\ 5 & 2 & -5 \end{pmatrix}.$$

1. Calculez, en utilisant les fonctions de Matlab, son déterminant, sa matrice inverse et sa diagonale.
2. Calculez puis afficher $B = AA^T$. Déterminez sa taille.

✎ 40 Vecteurs logiques

Créez le vecteur x défini ci-dessous où t va de -98 à 71 avec un pas de $1/3$.

$$x_i = \begin{cases} 0 & \text{si } t_i < 0, \\ 5 & \text{si } t_i \geq 0. \end{cases}$$

Si vous êtes très fort, vous êtes capable d'écrire le code en une unique instruction de 19 caractères !

✎ 41 Programmation

Créez la matrice carrée de taille N (à définir) dont les termes sont définis par $H_{i,j} = i/j$ avec $i,j \in \{1,\ldots,N\}$. Utilisez deux boucles `for` imbriquées. Il est préférable d'initialiser la matrice H au début : `H = zeros(N)` car cela évite à Matlab de redimensionner la matrice pendant l'exécution du programme et permet donc de gagner du temps.

Si vous êtes vraiment très fort, vous êtes capable d'écrire le code avec une seule boucle `for`, voire carrément aucune !

✎ 42 Fonction

Écrivez une fonction Matlab qui calcule la trace d'une matrice donnée en argument (la trace est la somme des termes diagonaux). Utilisez `size` pour connaître la taille de la matrice. La fonction doit vérifier que la matrice est carrée et retourner une erreur sinon (fonction `error`).

✎ 43 Tracé d'une sinusoïde

Tracez le signal sinusoïdal $x(t)$ de période $T_0 = 500$ ms, d'amplitude $A = 2$ et de moyenne $m = 0{,}5$ sur 2 secondes :

$$x(t) = m + A\sin(2\pi f_0 t) \qquad \text{où} \qquad f_0 = \frac{1}{T_0}.$$

Pour ce faire, on calculera 50 points par période.

✎ 44 Polynômes

On considère le polynôme $p(x) = 4x^2 - 3x + 5$.

1. Évaluer ses valeurs (`polyval`) aux points $x_1 = 3{,}6$ et $x_2 = \sqrt{2} + 3j$.
2. Évaluer ses racines (`roots`).

✎ 45 Quizz Matlab

Entraînez vous sur le quizz disponible sur http://users.polytech.unice.fr/∼strombon/SSI/3.matlab/quizmatlab.htm [3].

9 Bibliographie & webographie

[1] Aide Matlab (`doc`).

[2] Jean-Thierry Lapresté, *Introduction à MATLAB*, Ellipses Marketing, 2009.

[3] Jean-Paul Stromboni, Quizz Matlab, http://users.polytech.unice.fr/∼strombon/SSI/3.matlab/quizmatlab.htm.

[4] *Getting Started with MATLAB*, http://www.mathworks.com/videos/matlab/getting-started-with-matlab.html (vidéo en anglais).

[5] *Step-by-Step Guides to MATLAB and Simulink*, http://fr.mathworks.com/academia/student_center/tutorials/ (vidéos en anglais).

10 Fonctions principales de Matlab

Consultez l'aide pour avoir des informations détaillées sur ces instructions.

Instructions particulières

`% commentaires`	commentaires.
`clear all`	supprime toutes les variables.
`close all`	ferme toutes les fenêtre graphiques.
`clc`	efface la fenêtre de commande.
`format`	format d'écriture des nombres.
`tic, toc`	chronomètre.

Opérations

`+, -, *, /, ^`	addition, soustraction, multiplication, division, puissance.
`.*, ./, .^`	multiplication, division, puissance élément par élément.

Variables spéciales et constantes

`pi`	π.
`i, j, 1i ou 1j`	nombre complexe i.
`Inf`	nombre infini.
`NaN`	« *not a number* » : exprime une indétermination.
`ans`	dernière résultat.

Graphiques

`figure`	nouvelle fenêtre graphique.
`plot`	trace une courbe en reliant les points.
`hold('on')`	permet de tracer plusieurs courbes l'un sur l'autre.
`subplot`	divise la fenêtre en plusieurs graphiques.
`xlabel, ylabel`	légende de l'axe des abscisses, des ordonnées.
`title`	titre du graphique.

Manipulation de fichier

`save, load`	enregistre ou charge les données d'un fichier MAT.
`dlmread, dlmwrite`	enregistre ou charge les données d'un fichier ASCII.

Opérations sur les matrices

`'` (apostrophe)	transposée.
`linspace`	génère un vecteur dont les éléments sont régulièrement espacés.
`ones`	crée une matrice remplie de 1.
`zeros`	crée une matrice remplie de 0.
`size`	taille d'une matrice.
`length`	longueur d'un vecteur.
`det`	déterminant.

Fonctions mathématiques

`exp`	exponentielle.
`cos, sin, tan`	cosinus, sinus, tangente.
`sinc`	sinus cardinal : $\mathrm{sinc}(x) = \sin(\pi x)/\pi x$.
`log`	logarithme népérien (ln).
`log10`	logarithme en base 10 ($\log_{10}$).
`sqrt`	racine carrée.
`real, imag`	partie réelle, imaginaire.
`abs`	valeur absolue, module.
`angle`	argument.
`conj`	conjugué.

Conversion en chaîne de caractères

`num2str`	convertit un nombre en chaîne de caractères.
`mat2str`	convertit une matrice en chaîne de caractères.

Aide

`help`	aide simple (documentation en ligne).
`doc`	aide détaillée (documentation hypertexte).
`lookfor`	recherche dans les mots-clés de l'aide.
`demo`	démonstrations.

Programmation

`function`	fonction Matlab.
`if`, `else`, `elseif`	condition.
`end`	fin d'un boucle ou d'une condition.
`for`	boucle (sur un nombre déterminé de fois).
`while`	boucle (sur un nombre indéterminé de fois).
`break`	termine l'exécution d'un boucle.
`error`	affiche un message d'erreur et arrête l'exécution.

Interaction avec l'utilisateur

`disp`	affiche une matrice ou un texte.
`fprintf`	affiche un texte formaté à l'écran ou dans un fichier.
`input`	pose une question à l'utilisateur.
`pause`	pause dans l'exécution d'un programme.

11 Blocs principaux de Simulink

Librairie Sources

Sine Wave	signal sinusoïdal.
Pulse Generator	signal créneau.
Constant	signal constant.
Step	échelon.

Librairie Math Operations

Gain	gain.
Product	produit de plusieurs signaux.
Sum	somme de plusieurs signaux.

Librairie Discrete

Unit Delay	retarde le signal d'une valeur à définir.

Librairie Continuous

Transfer Fcn	fonction de transfert.

Librairie Sinks

Scope	graphe affichant un ou plusieurs signaux.
To Workspace	permet d'exporter le vecteur des valeurs prises par le signal vers Matlab.

Librairie Signal Routing

Mux	permet de multiplexer deux signaux (pour afficher deux signaux sur le même graphe).
Switch	permet de choisir l'un des deux signaux d'entrée à l'aide d'une condition.