Tutoriel matlab

F. Ravelet[a]
[a] *Arts et Metiers ParisTech, DynFluid,*
151 boulevard de l'Hôpital, 75013 Paris, France.
contact: florent.ravelet@ensam.eu

4 octobre 2016

1 Introduction

1.1 Description de l'environnement matlab

Matlab est l'acronyme de "Matrix Laboratory". Il s'agit d'un logiciel de calcul numérique (et non de mathématiques ou de calcul formel), orienté vers l'utilisation de vecteurs, tableaux et matrices.

Lorsqu'on ouvre le logiciel, on arrive sur la fenêtre principale, la "Command Window" (voir Fig. 1).

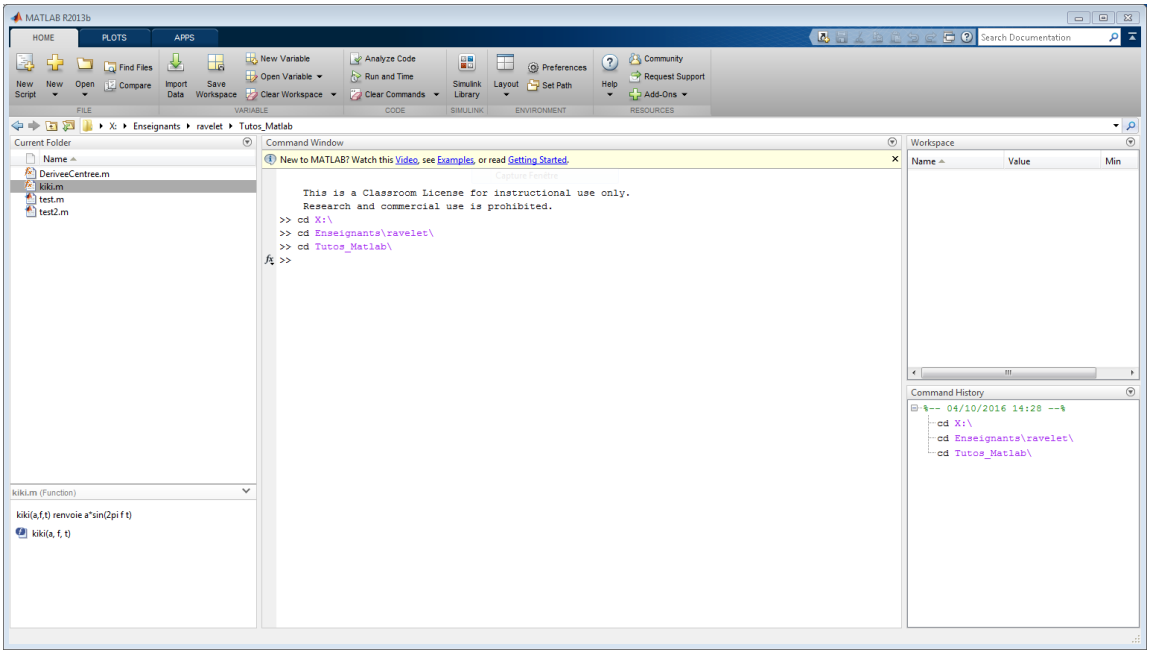

FIGURE 1 – Fenêtre principale de Matlab

Après >>, le prompt, on peut taper des commandes. Le résultat de l'exécution s'inscrit alors dans la fenêtre principale. Pour rentrer une suite complexe d'instructions, on va créer un "script" en utilisant l'éditeur intégré (voir Fig. 2). Une fois le script enregistré, on peut l'exécuter en tapant son nom dans la fenêtre de commande, ou en cliquant dans l'éditeur sur la flèche verte "run".

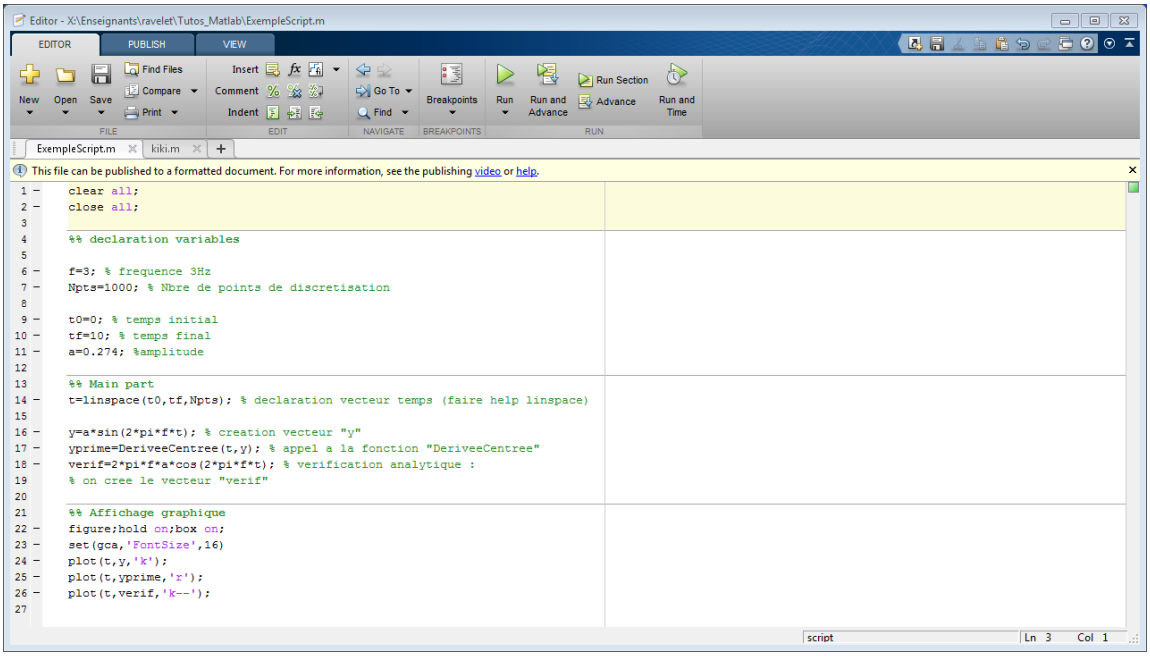

FIGURE 2 – Editeur de scripts et de fonctions de Matlab

1.2 Prise en main

Tapez dans la fenêtre de commande la ligne suivante

```
>>Matrice = ones(2,3)
```

On obtient :

```
Matrice =
1 1 1
1 1 1
```

On a donc déclaré une variable appelée “Matrice”, de taille 2×3, contenant des 1. L’aide en ligne de Matlab vous sera très utile : tapez

```
>>help ones
 ONES   Ones array.
    ONES(N) is an N-by-N matrix of ones.

    ONES(M,N) or ONES([M,N]) is an M-by-N matrix of ones.

    ONES(M,N,P,...) or ONES([M N P ...]) is an M-by-N-by-P-by-... array of
    ones.

    ONES(SIZE(A)) is the same size as A and all ones.

    ONES with no arguments is the scalar 1.

    ONES(M,N,...,CLASSNAME) or ONES([M,N,...],CLASSNAME) is an M-by-N-by-...
```

```
    array of ones of class CLASSNAME.

    Note: The size inputs M, N, and P... should be nonnegative integers.
    Negative integers are treated as 0.

    Example:
       x = ones(2,3,'int8');

    See also eye, zeros.
```

On a alors une description de la fonction, ainsi que de sa syntaxe, plus des exemples, et un "see also" qui peut être très utile. Essayez ! Regardez aussi ce que donne :

```
>>doc ones
```

Notez enfin que si on ne veut pas afficher le résultat d'une commande, il faut mettre un " ;" à la fin de la ligne.

1.3 Vecteurs, tableaux

Pour créer un vecteur :

```
>>a = [1,2,4,7,8]
```

Essayez :

```
>>a'
```

Pour accéder au premier élément :

```
>>a(1)
```

Pour le dernier élément :

```
>>a(end)
```

Pour déclarer une matrice :

```
>>B=[2,4,6,8,10 ; 1,2,3,4,5]
```

Que fait :

```
>>B*a
```

? et

```
>>B*a'
```

?

Le produit "*" est donc le produit matriciel. Pour effectuer le produit terme-à-terme (utile pour « vectoriser » un code), on utilisera ".*". Par exemple, pour calculer le carré de chaque élément de a :

```
>>acarre = a.*a
>>acarre = a.^2
```

1.4 Boucles

L'exemple suivant permet de comprendre la syntaxe générale :

```
>>a = linspace(0,10,21);
>>zouzou=zeros(size(a));
>>for i=1:length(zouzou)
zouzou(i)=a(i)^2;
end
>>zouzou
```

Notez que les lignes précédentes font exactement la même chose que l'instruction

```
>>acarre = a.^2
```

vue précédemment : la différence est que cette forme est plus optimisée en temps CPU.

2 Exercices

2.1 Exercice 1

Comprendre ce que fait le code suivant, contenu dans le script "ExempleScript.m" et faisant un appel à la fonction "DeriveeCentree.m" . Pour cela, éditer ces deux fichiers :

```
>> cd X:\
>> cd Enseignants\ravelet\Tutos_Matlab\
>> edit ExempleScript.m
>> edit DeriveeCentree.m
```

ExempleScript.m

```
clear all;
close all;

%% declaration variables

f=3; % frequence 3Hz
Npts=1000; % Nbre de points de discretisation

t0=0; % temps initial
tf=10; % temps final
a=0.274; %amplitude

%% Main part
t=linspace(t0,tf,Npts); % declaration vecteur temps (faire help linspace)

y=a*sin(2*pi*f*t); % creation vecteur "y"
yprime=DeriveeCentree(t,y); % appel a la fonction "DeriveeCentree"
verif=2*pi*f*a*cos(2*pi*f*t); % verification analytique :
% on cree le vecteur "verif"
```

```
%% Affichage graphique
figure;hold on;box on;
set(gca,'FontSize',16)
plot(t,y,'k');
plot(t,yprime,'r');
plot(t,verif,'k--');
```

DeriveeCentree.m

```
function df=DeriveeCentree(x,y)

df=zeros(size(x));
for i=2:length(x)-1
    dx=x(i+1)-x(i-1);
    dy=y(i+1)-y(i-1);
    df(i)=dy/dx;
end

df(1)=(y(2)-y(1))/(x(2)-x(1));
df(end)=(y(end)-y(end-1))/(x(end)-x(end-1));
```

2.2 Exercice 2

1. Créer un script qui permet de trouver une racine de :

$$f(x) = x^3 - 5x^2 + x + 2$$

   en utilisant la méthode itérative de Newton, initialisée à $x = 5$. Votre script devra donner le nombre d'itérations nécessaires à obtenir convergence à 10^{-12} près.
2. Essayer avec une autre valeur initiale.
3. Créer un script utilisant la méthode de la sécante.
4. Comparer leur efficacité.

Description de la méthode de Newton La méthode de Newton est une méthode de résolution de l'équation $f(x) = 0$. Si x_0 est proche de la racine on peut faire un développement de Taylor à l'ordre 1 de la fonction f en x_0 :

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + O((x - x_0)^2)$$

Pour trouver une valeur approchée de la racine, on ne garde que la partie linéaire du développement, on résout :

$$f(r) = 0 \approx f(x_0) + (r - x_0)f'(x_0)$$

donc (si $f'(x0) \neq 0$) :

$$r \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

Graphiquement, cela revient à tracer la tangente à la courbe représentative de f et à chercher où elle coupe l'axe des x. On considère donc la suite récurrente définie par une valeur initiale u_0 proche de la racine et par la relation :

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$$

On itère jusqu'à convergence.

Description de la méthode de la sécante La méthode de la sécante est une méthode dérivée de celle de Newton où l'on remplace $f'(x)$ par $(f(x+dx) - f(x))/(dx)$. On obtient la relation de récurrence :

$$u_{n+1} = u_n - \frac{u_n - u_{n-1}}{f(u_n) - f(u_{n-1})} f(u_n)$$

Il faut donc deux valeurs initiales u_0 et u_1.