Calculs élémentaires avec Matlab

S. Labrosse

1 Introduction

Matlab est un logiciel de calcul numérique (à ne pas confondre avec les programmes de calcul formels tels que Mathematica ou Maple) et de visualisation qui contient un grand nombre de fonctions d'analyse numérique couramment utiles. La plupart des outils contenus dans le numerical recipes [*Press et al.*, 1992] ont leur équivalent dans Matlab. On peut ainsi simplement représenter des données ou des résultats de modèles numériques, faire des opérations plus ou moins complexes sur ces données. On peut aussi faire des petits modèles simples qui nécessitent la résolution d'équations différentielles.

Le nom Matlab est une abréviation de matrix laboratory et en effet Matlab est particulièrement construit pour faire des calculs sur les matrices. Matlab est un logiciel propriétaire mais il est à noter qu'il existe des logiciels libres qui s'approchent de Matlab. En particulier, Octave a une syntaxe identique à celle de Matlab. Son rendu graphique est cependant beaucoup moins évolué que celui de Matlab.

Il existe sur la toile de nombreux tutoriaux pour Matlab. Il ne faut bien sûr pas hésiter à les utiliser. La section 2 présente un tours rapide de l'utilisation de Matlab et ceux qui se sentent suffisamment au point peuvent directement aller à la section 3 pour démarrer le premier calcul pratique.

2 Petit guide de Matlab

2.1 Utilisation

Matlab peut s'utiliser de plusieurs manières différentes, et toutes sont utiles à des moments différents du développement de nos outils. On peut directement taper des lignes de commandes, écrire des scripts et fonctions que l'on appelle par ligne de commande ou faire tourner ces scripts depuis un shell (fonctionnement en "batch"). Nous allons surtout nous concentrer sur les deux premières utilisations qui sont celles utilisées lors du développement de script.

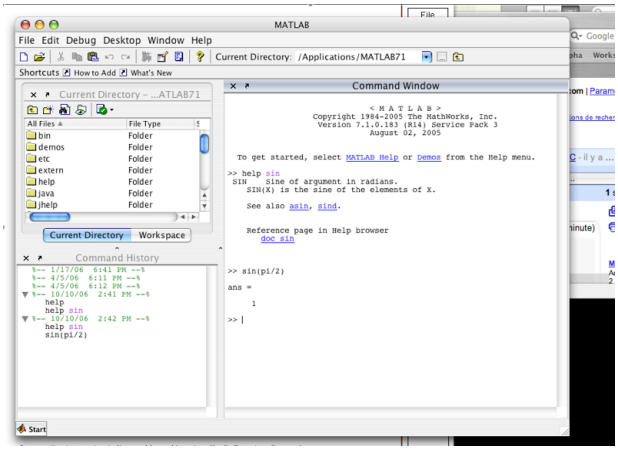

FIG. 1 – Fenêtre Matlab, qui se décompose en une fenêtre de commande sur la partie droite, une fenêtre de navigation en haut à gauche et une fenêtre d'historique en bas à gauche.

On commence par démarrer Matlab, soit en cherchant dans les menus déroulant soit en tapant

```
matlab &
```

dans un terminal. Une fenêtre apparaît composée de plusieurs sous-fenêtres (fig. 1).

2.2 Quelques commandes de base

La première commande à connaître dans Matlab (mais c'est également vrai dans tout autre logiciel) est celle qui permet d'obtenir de l'aide sur les instruction utiles. Sans surprise, il s'agit de

```
help
```

qui si elle est tapée seule donne accès à une liste de rubriques. On peut aussi demander de l'aide sur une fonction particulière :

```
help sin
```

qui vous explique comment utiliser la fonction `sin`. Pour voir s'il on a bien compris, on peut essayer

```
sin(pi/2)
```

et vérifier que le résultat est conforme à notre attente. On a en même temps découvert que le nombre π est connu par Matlab et s'écrit simplement `pi`. Il existe d'autres "nombres" ainsi connus : `e, i, j(=i)`. Ces variables sont ainsi réservées et mieux vaut ne pas en changer la valeur dans vos programmes. On y reviendra.

Pour avoir une idée de la précision des calculs de Matlab, on peut calculer $\sin \pi$ et on obtient

```
>> sin(pi)
ans =
     1.2246e-16
```

Cette précision ne doit pas être confondue avec le nombre de décimales utilisées pour l'affichage. Celui-ci peut-être modifié en utilisant l'instruction `format` :

```
>> pi
ans =
     3.1416
>> format long
>> pi
ans =
     3.14159265358979
```

La commande `help` permet d'obtenir un guide d'utilisation d'une instruction que l'on connaît déjà. Comment procéder quand on ne connaît pas l'instruction en question et que l'on cherche à effectuer une opération particulière ? L'instruction `lookfor` permet de chercher des mots clés dans la première ligne de l'aide de toutes les instructions existantes. Enfin, l'instruction `helpdesk` permet de lancer un navigateur de la toile sur une page de Matlab contenant de nombreuses ressources utiles. Par exemple, il existe une rubrique appelée "Getting started" qui permet de faire un tour rapide des fonctionnalités de base de Matlab.

La première instruction indispensable est bien sûr celle d'affectation d'une valeur :

```
a = 3
```

Comme dans tous les langages de programmation le signe `=` n'a pas le même sens qu'en mathématique. L'instruction précédente signifie juste : "donne à la variable `a` la valeur 3". Cette variable gardera sa valeur jusqu'à ce qu'elle soit modifiée par une autre affectation. Par exemple

```
a = a + 1
```

prend la valeur de `a` (3), lui ajoute 1, et affecte le résultat à `a` qui prend alors la valeur 4. Ceci montre bien la différence entre le signe `=` en Matlab et une égalité mathématique. L'égalité mathématique apparaît régulièrement lorsque lorsque l'on teste une condition et s'écrit

```
test = a == 2
```

dont le résultat est une variable booléenne qui vaut 1 si l'équation mathématique $a = 2$ est vraie et 0 dans le cas contraire.

Dans tous les cas précédents, le résultat du calcul effectué était affiché à l'écran juste après l'opération effectuée. Cela peut s'avérer ennuyeux (on le verra très vite) dans le cas où le

nombre d'opérations devient grand. Pour éviter cela, on ajoute un `;` à la fin de l'instruction. L'opération est effectuée mais le résultat ne s'affiche pas dans la fenêtre de calcul. Pour voir la valeur contenue dans une variable, par exemple `test`, il suffit de taper le nom de cette variable. Pour savoir quelles variables sont attribuées dans la mémoire de Matlab, il suffit de taper

```
whos
```

qui permet de lister les variables utilisée et un certain nombre de leurs attributs. On voit alors que toutes les variables déjà utilisées existent encore. Si l'on veut les effacer de la mémoire de Matlab, il suffit d'utiliser l'instruction

```
clear
```

qui, utilisée sans argument, efface toute la mémoire. On peut aussi l'utiliser pour effacer la mémoire associée à une variable particulière. Supposons que par erreur on ait tapé

```
pi = 2
```

La variable `pi` prend alors la valeur 2 et perd son sens habituel. On peut effacer cette erreur en tapant

```
clear pi
```

qui libère la variable `pi`. On peut le vérifier en tapant

```
pi
```

qui nous renvoie la valeur standard de π.

2.3 Fichiers scripts et fonctions

Avant d'aller plus loin dans les opérations compliquées, il est désirable de pouvoir travailler sur un fichier qui contient les instructions que l'on veut exécuter, sans avoir à les retaper à chaque fois. On appellera un tel fichier un script Matlab, auquel on donnera un nom avec l'extension `.m`. Si par exemple on appelle notre fichier `toto.m`, son exécution dans Matlab s'effectuera en tapant

```
toto
```

Pour que cela fonctionne il faut bien sûr que Matlab sache où se trouve le fichier `toto.m`. C'est le cas si le fichier se trouve dans le répertoire de travail d'où l'instruction `matlab` a été lancée. C'est également le cas si le fichier se trouve dans un répertoire `Matlab` ou `matlab` dans votre répertoire racine.

Un type particulier de fichier script est très utile : ceux contenant des fonctions. Par exemple, on peut créer un fichier `plusetmoins.m` contenant

```
function [plus moins] = plusetmoins(x,y)
plus = x+y ;
moins = x-y ;
endfunction
```

et utiliser cette fonction très intéressante dans votre fenêtre de commande en tapant

```
plusetmoins(5,3)
```

qui renvoie les résultats attendus 8 et 2. Il est à noter que plusieurs arguments peuvent être utilisés en entrée et que parmi les résultats qui sont donnés entre `[ ]`, seul le premier est le vrai résultat de la fonction, les autres étant des résultats secondaires. On s'en rend compte facilement en tapant successivement

```
a = plusetmoins(5,3)
[a b] = plusetmoins(5,3)
```

L'utilisation des `[ ]` nous amène naturellement à introduire les matrices.

2.4 Matrices

Matlab est particulièrement utile pour travailler avec des matrices. Lorsque l'on utilise l'instruction `whos` on voit par exemple

```
  >> whos
  Name      Size                    Bytes  Class
  a         1x1                         8  double array
  ans       1x1                         8  double array

Grand total is 2 elements using 16 bytes
```

c'est à dire que même les variables scalaires comme `a` sont considérées comme des matrices 1x1.

Pour définir une matrice, on utilise les crochet, en séparant les éléments d'une ligne par des virgules ou des espaces et en séparant les lignes par des passage à la ligne ou des `;`. Par exemple

```
>>  A=[1,2
    3,4
    5,6]

A =
```

```
     1     2
     3     4
     5     6
```

est équivalent à

```
>>  A=[1 2;3 4;5 6]

A =

     1     2
     3     4
     5     6
```

Lorsqu'une matrice est crée, on peut en connaître la taille en utilisant les instruction `size` et `length` qui ont elles-mêmes pour résultats des matrices :

```
>> size(A)

ans =

     3     2

>> length(A)

ans =

     3
```

On peut assembler des vecteurs et des matrices pour en faire des matrices plus grandes. Vous pouvez par exemple essayer

```
a=[-1,-2]
b=-3
v=[a,b] % les colonnes les unes a cotes des autres
vv=[v;v] % les lignes les unes  sous   les autres
```

pour voir le résultat. La transposée de `A` s'écrit simplement `A'`. Pouvez vous prédire le résultat de

```
B=[A,[a';b]]
```

Par ailleurs, il existe des fonctions pour remplir des matrices systématiquement. Les plus simples remplissent une matrice de 0 (instruction `zeros`) ou de 1 (`ones`) ou bien créent la matrice identité (`eye`). On peut ensuite utiliser une boucle implicite pour remplir un vecteur de valeurs incrémentées. Par exemple,

```
>> x=[0:5]

x =

     0     1     2     3     4     5
```

permet d'obtenir des valeurs régulièrement espacées de 1, qui est la valeur par défaut de l'incrément. Pour changer cette valeur, il suffit de préciser l'incrément voulu :

```
>> y=[0:.2:1]

y =

         0    0.2000    0.4000    0.6000    0.8000    1.0000
```

On peut de la même manière se créer une échelle logarithmique avec `x=logspace(-2,2,5)`.

Les éléments d'une matrice peuvent être utilisés et sont référencés simplement comme `A(i,j)` pour la ligne `i` et la colonne `j`. On peut également utiliser les boucles implicites introduites précédemment pour accéder à une zone de la matrice. Par exemple, `A(3,1 :2)` correspond aux éléments de la ligne 3, des colonnes 1 à 2. Pour accéder à un ligne ou une colonne complète, il suffit d'omettre les bornes de la boucle. Ainsi, `A( :,2)` donne la colonne 2 entière. Enfin, pour accéder à la diagonale, on utilise l'instruction `diag` qui sert également à construire une matrice diagonale à partir d'un vecteur. Ainsi, `d=diag(A)` donne un vecteur formé à partir de la diagonale de `A` et `diag(d)` donne une matrice carrée diagonale avec `d` pour diagonale.

Maintenant que l'on sait générer des matrices, on veut pouvoir jouer avec. Les opérateur usuels `+, *` sont interprétés au sens des matrices par Matlab. Ainsi,

```
A=[1 2 ; 3 4];
B=[1 2 ; 3 4 ; 5 6];
B*A
ans =
```

```
   7  10
  15  22
  23  34
```

permet de calculer le produit de matrice `B*A`. On peut alors vérifier que le produit de matrices n'est généralement pas commutatif puisque `A*B` ne peut être effectué.

Parfois, il est nécessaire de faire un produit terme à terme entre deux matrices, au lieu d'un produit de matrices. Il est alors nécessaire de modifier l'opérateur `*` en le précédent d'un point. Comparons les résultats :

```
C=[4 5 ; 7 8];
A*C
ans =

  18  21
  40  47

 A.*C
ans =

   4  10
  21  32
```

La même modification s'applique à tous les opérateurs usuel (sans effet sur l'addition bien sûr), `*, /, ^`.

Parmi les opérations les plus utiles sur les matrices, le calcul de l'inverse (instruction `inv`) et des valeurs et vecteurs propres (instruction `eig`) se font de façon routinière en Matlab. Il est à noter que les fonctions mathématiques usuelles (`sin, exp`, etc.) son appliquées à chaque élément de la matrice. Pour calculer l'exponentielle d'une matrice, utiliser `expm`.

2.5 Représentations graphiques

Matlab permet de faire des représentations graphiques très sophistiquées. Nous allons nous contenter ici de graphiques bi-dimensionnels pour représenter des fonctions d'une variable. Pour ce faire, il est d'abord nécessaire de calculer les valeurs de la fonction en des points déterminés. On utilise pour cela les boucles implicites introduites précédemment pour remplir des vecteurs contenant les valeurs qui nous intéressent. Par exemple

```
t=[0:0.05:4]';
x=sin(3*t);
```

```
plot(t,x)
```

permet de calculer la fonction $\sin 3t$ pour t variant de 0 à 4 avec un pas de 0.05 et de représenter ces valeurs. Par défaut, les points ne sont pas représentés mais ils sont reliés par des traits. On peut changer ce comportement en ajoutant des options à la fonction `plot` (toutes les options sont bien sûr décrites dans l'aide en ligne). Par exemple, essayez `plot(t,y,'r+')` ou `plot(t,y,'^')`.

Lorsque l'on utilise plusieurs fois l'instruction `plot`, chaque nouveau graphique remplace le précédent. Si l'on veut ajouter une figure sur une déjà existante, on utilise l'instruction `hold on`. Tous les appels à l'instruction `plot` qui suivront ajouterons le résultat à la figure en cours. Pour remplacer (au lieu d'ajouter) la figure précédente, il faut utiliser l'instruction `hold off`. Enfin, pour faire apparaître une nouvelle figure sans perdre la précédente, on utilise l'instruction `figure`.

Entre autres instruction utiles pour modifier l'apparence d'une figure, les instruction `xlabel`, `ylabel` et `title` sont importantes. L'instruction `subplot` permet de découper la figure en sous-figures.

Enfin, si l'on veut sauvegarder une figure dans un fichier pour, par exemple, l'inclure dans un document texte, on utilise l'instruction `print`.

3 Calcul des pressions dans la Terre

L'objectif de ce TP est de mettre en œuvre le calcul numérique d'intégrales dans Matlab pour calculer les pressions et la gravité dans la Terre. La donnée de départ est un profil de densité en fonction du rayon, obtenu par sismologie. On utilisera ici le modèle PREM dont les valeurs de densité sont données dans le fichier `PREM_dens.dat` de l'espace de formation.

La lecture du fichier de données se fait avec l'instruction `load` qui va charger ces données dans une matrice portant le nom du fichier. Avant toute opération, il est utile de représenter ces données pour être sûr de partir sur de bonnes bases.

La distribution radiale de densité étant connue, on calcule l'accélération de la gravité en utilisant la solution formelle à l'équation de Poisson :

$$g(r) = \frac{4\pi G}{r^2} \int_0^r u^2 \rho(u) du. \tag{1}$$

Cette équation semble poser problème en 0, et elle en pose effectivement pour le calcul numérique. À vous de déterminer le comportement de g au centre et d'agir en conséquence. Le calcul numérique de l'intégrale peut se faire avec l'instruction `cumtrapz` de Matlab ou en programmant vous même une fonction d'intégration par méthode trapèzes ou autre. Ne pas hésiter à vérifier le résultat par une représentation graphique.

Le calcul de la pression se fait alors en supposant un équilibre hydrostatique,

$$\frac{dP}{dr} = -\rho g. \tag{2}$$

Là encore, il s'agit de calculer une intégrale. Attention à ne pas oublier la condition limite à la surface de la Terre !

À titre de vérification, représenter la densité, la gravité et la pression en fonction du rayon et donner la pression au centre de la Terre et à la frontière noyau-manteau, la masse de la Terre, sa densité moyenne, et la gravité à la surface.

4 Régression linéaire

L'objectif de ce TP est d'écrire un petit programme de régression linéaire en Matlab. Nous allons utiliser la méthode du χ^2, que l'on va d'abord brièvement expliquer ici.

Nous disposons d'une série de N points de mesures $(x_i, y_i \pm \delta y_i)$ pour lesquels on cherche à obtenir un ajustement linéaire[1] sous la forme $y = ax + b$. On cherche donc à trouver les valeurs de a et de b qui permettent de représenter le "mieux" possible les données. Le concept de mieux nécessite définition et nous utiliserons ici le critère des moindres carrés qui correspond à la minimisation de la norme L_2 de l'erreur introduite par l'ajustement. En d'autres termes, on cherche les coefficients a et b qui minimisent la valeur de

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - ax_i - b}{\delta y_i} \right)^2 . \tag{3}$$

Les points de mesure étant connus, χ^2 est une fonction de a et de b et le problème est simplement d'en déterminer le minimum. On peut graphiquement représenter $\chi^2(a, b)$ pour se faire une idée mais on peut déjà dire, vu sa définition, que $\chi^2(a, b)$ est un polynôme du degré 2 en a et b. Comme par ailleurs il est toujours positif, il existe un minimum unique et ce minimum est définit par deux équations

$$\frac{\partial \chi^2}{\partial a} = 0 \tag{4}$$

$$\frac{\partial \chi^2}{\partial b} = 0 \tag{5}$$

$\chi^2(a, b)$ étant de degré 2 en a et b, ces deux équations sont de degré 1 en a et b, c'est à dire des droites. La détermination des coefficients a et b est donc simple. Nous allons appliquer cette approche au calcul d'une régression de la densité en fonction du rayon dans le noyau.

[1] L'approche présentée ici peut en fait fonctionner avec un ajustement polynomial de tout ordre et la généralisation est très simple.

On peut démontrer que la densité dans le noyau ρ, en dehors de la discontinuité à la limite graine-noyau liquide (FGN), varie avec le rayon r comme

$$\rho = \rho_c \left[1 - \left(\frac{r}{L} \right)^2 \right], \tag{6}$$

expression étant valable à l'ordre 3 en r/L. Nous allons déterminer les valeurs de ρ_c et L qui permettent le meilleur ajustement du modèle PREM. Aucune incertitude sur les densités n'est donnée par PREM mais on peut supposer que 5% d'incertitude sur chaque valeur.

Voici la démarche à suivre :

1. Écrire le χ^2 dans le cas de l'expression (6) et déterminer la forme explicite et générale des équations (4) et (5).
2. Après avoir lu les données de densité de PREM, produire un modèle sans graine en retranchant le saut de densité à la FGN de toute la graine. Produire un vecteur contenant les incertitudes (5%) sur les densités.
3. Déterminer les valeurs de ρ_c et L et discuter. Représenter sur un graphe les valeurs de densité avec leurs incertitudes et la régression obtenue.
4. Pour se faire une idée de l'incertitude associée aux deux coefficients de la régression linéaire, représenter $\chi^2(a, b)$ autour du minimum. Pouvez-vous proposer une expression pour l'incertitude sur a et b ?

Références

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in Fortran*, 2nd ed., Cambridge University Press, Cambridge, 1992.