Tutorial Matlab

Alexandra Bac

Méthodes numériques
Polytech Marseille - IRM 3A

Matlab (ou son clône gratuit Octave) a été construit pour la manipulation de vecteurs et de matrices. Il permet donc leur manipulation de manière simple et compacte (même si les notations peuvent parfois surprendre).

Scripts / chargement

Définition d'un vecteur / matrice La définition se fait par lignes. Le séparteur de colonnes et `,` et celui de lignes `;`

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

Quelques fonctions sur les matrices :

- *size*

```
>> A = gallery(3)
A =
  -149   -50  -154
   537   180   546
   -27    -9   -25
>> size(A)
ans =
     3     3
>> size(A,1)
ans =
     3
```

- *reshape*

```
>> A = [1,2,3;4,5,6]
A =
     1     2     3
     4     5     6
>> reshape(A,3,2)
ans =
     1     5
     4     3
     2     6
```

Transposée Directement accessible avec `'`

```
>> U
U =
     1     2     3
>> U'
ans =
     1
     2
     3
```

Opérations matricielles Les opérateurs `+` et `-` permettent de

sommer / soustraire des matrices et vecteurs de même taille. On peut également multiplier une matrice/vecteur par un nombre.

```
>> A+A
ans =
     2     4     6
     8    10    12
>> 3*A
ans =
     3     6     9
    12    15    18
```

Pour la multiplication/puissance :

- `*` est la multiplication matricielle et `^` la puissance matricielle
- `.*` et `.^` désignent la mutiplication et puissance terme à terme (i.e. coefficient par coefficient de la matrice/vecteur)

```
>> A
A =
     1     2     3
     4     5     6
>> B
B =
     0     1
     1     1
     1     0
>> A*B
ans =
     5     3
    11     9
>> A*A
??? Error using ==> mtimes
Inner matrix dimensions must agree.
>> A.*A
ans =
     1     4     9
    16    25    36
>> A.^3
ans =
     1     8    27
    64   125   216
```

Vecteurs séquences On peut générer aisément des vecteurs contenant des suites de nombres :

- *n1 : n2* Nombres de `n1` à `n2` par pas de 1
- *n1 : p : n2* Nombres de `n1` à `n2` par pas de `p`

Manipulation des lignes/colonnes La manipulation des lignes et colonnes se fait de manière directe :

- Accès à un coefficient : *()*
- Permet également d'accéder à une plage de coefficients (ligne/colonne/sous-matrice)

```
>> A
A =
     1     2     3
     4     5     6
>> A(2,3)
ans =
     6
>> A(1:2,2)
ans =
     2
     5
>> A(:,1)
ans =
     1
     4
>> A(:,1)=[11;44]
A =
    11     2     3
    44     5     6
>> A(3,:) = [7,8,9]
A =
    11     2     3
    44     5     6
     7     8     9
```

Quelques fonctions utiles Quelques fonction, pêle-mêle :

- Valeurs aléatoires :
  A = rand(n,m)
  tire une matrice `n x m` aléatoire (répartition uniforme)
  A = randn(n,m)
  tire une matrice `n x m` aléatoire (répartition normale)
- Matrice identité :
  A = eye(n)
- Matrice de zéros :
  A = zeros(n)
- Matrice diagonale / diagonale d'une matrice :
  A = diag(b) si `b` est un vecteur, on obtient la matrice de diagonale `b`
  b = diag(A) si `A` est une matrice, on obtient le vecteur diagonale de `A`
- Norme matricielle :
  norm(A,type)
  où `type` vaut 1, 2, inf ou 'fro'
- Valeurs propres :
  [P,D] = eig(A)
- Valeurs singulières :
  [U,S,V] = svd(A)
- Gauss / décomposition LU :
  [L,U] = lu(A) `A = L*U`
  [L,U,P] = lu(A) `P*A = L*U`
- Householder / décomposition QR :
  [Q,R] = qr(A)
- Résolution du système linéaire `Au = b` :
  u = A\b
  Gauss, Cholesky ou Householder en fonction de `A`

Le graphique 2D Principales fonctions :

- Superposer les tracés :
  hold on hold off
- Nettoyer la fenêtre graphique :
  clf
- Tracé 2D :
  plot(X,Y,linespec)
  trace la suite de points (X_i, Y_i) reliés par des lignes. `linespec` code le type de lignes :
  - `'b'`, `'r'`, `'g'` ... couleurs
  - `'.'` tracé seulement des points (pas lignes)
  - `'--'` pointillés ...
- Tracé d'une courbe 2D :
  ezplot(fun) où `fun` peut être une équation paramétrique ou implicite (nombre de variables)

Le graphique 3D Fonctions principales :

- Tracé 3D :
  plot3(X,Y,Z,linespec)
  trace la suite de points (X_i, Y_i, Z_i) reliés par des lignes. `linespec` code le type de lignes :
- Tracé d'une courbe paramétrique 3D :
  ezplot3(funx, funy, funz)
- Tracé d'une surface paramétrique 3D :
  ezsurf(...)
- Générer une grille 2D à partir de deux vecteurs `x` et `y` :
  [X,Y] = meshgrid(x,y)
- Tracé 3D :
  mesh(X,Y,Z)
  surf(X,Y,Z)

Structures de contrôle, fonctions Bien entendu, les structures de contrôle classiques sont disponibles. Quelques exemples qui suffiront. Pour le `if`, classique :

```
if (x == 3)
   y = x ;
end
```

Pour le `for`, on utilise un vecteur :

```
x = [] ;
for i = 1:10
   x = [x,i] ;
end
```

Pour le `while`, le classique :

```
x = 0 ;
while (x < 10)
   x = x+1 ;
end
```

Et enfin, un exemple de fonction:

```
function [y,z] = f(x)
% fonction renvoyant deux résultats y et z
y = x.^3-2*x-5;
z = 2*x.^3-x.^2;
end
```

Attention, toute fonction est stockée dans un fichier du même nom, donc ici, dans "f.m". Donc un fichier par fonction !

Aspects fonctionnels Matlab dispose d'un noyau fonctionnel. On peut donc définir des objets de type fonctionnel en utilisant `@`. Syntaxe :

```
@(arg1, arg2, ...) valeur
```

qui est équivalente à :

$$(\text{arg}_1, \text{arg}_2, \ldots) \mapsto \text{valeur}$$

Ces objets peuvent ensuite être affecté dans des variables, passés en argument ...
Exemple :

```
f = @(x) x.^2+x+1 ;
y = f(2.5) ;
```

Equations (symbolique) Eléments principaux :

- Une équation s'écrit entre *' '*
- Substituer :
  subs(eq, old, new)

Images 2D

- Chargement d'une image (dans une matrice) :
  A = imread('file')
- Conversion d'une image en flottants simple précision :
  im2single(A)
- Affichage d'une image (matrice) :
  imshow(A)
- Plusieurs subplots :
  subplot(n,m,i)

Vidéos

- Exportation d'une vidéo à partir d'images :

```
video_f= VideoWriter('video_file.avi');
open(video_f);
for i=1:n
    writeVideo(video_f,Flist{i});
end
close(video_f);
```

ACP

- ACP de points :
  coeff = pca(X);

Statistiques

- Moyenne de points :
  `S = mean(X);`
- Ecart type :
  `Y = std(X);`