Pour définir un vecteur la syntaxe est une des suivantes :

```
>> b=[1 2]   %un vecteur ligne de dimension 1x2
>> b=[1,2]   %la même chose
>> c=[3;4;5]   %un vecteur colonne de dimension 3x1
```

Les matrices suivent la même syntaxe que les vecteurs. Les composantes des lignes sont séparées par des virgules et chaque ligne est séparée de l'autre par un point virgule.

```
>> A=[1 2;3 4]   %une matrice de dimension 2x2
```

On peut facilement faire la concaténation de tableaux, par exemple:

```
>> A=[A;b]
>> B=[A c]
```

Le double point (:) est l'opérateur d'incrémentation dans MATLAB

```
>> x=1:10        % création d'un vecteur ligne formé de 10 entiers
>> x=10:-1:1     % à rebours
>> x=0:0.1:0.5   % un incrément de 0.1 plutôt que 1
```

- On peut accéder aux éléments de Matrice par leurs coordonnées ou par leur ordre dans la matrice.

```
>> A21=A(2,1)
A21 =  3
>> A4=A(4)
A4 =  2
```

Il est aussi possible de stocker dans un vecteur une ligne (ou colonne) d'une matrice. Ainsi, si l'on veut stocker la deuxième colonne de la matrice A dans un vecteur V, la commande est la suivante :

```
>> V=A(:,2) %(:) signifier toutes les lignes
V =
     2
     4
     2
```

De la même manière, si l'on veut stocker les lignes 2 et 3:

```
>> V2=A(2:3,:) % (2:3) signifier les lignes 2 et 3
               % (:) signifier toutes les colonnes
V2 =
     3     4
     1     2
```

Dans certaines applications, il est parfois utile de connaître les dimensions d'une matrice, et la longueur d'un vecteur. Dans ce cas, on utilise les fonctions length et size.

```
>> V = [0:0.1:10];
>> n = length(V)
>> M = [1 2 3; 4 5 6];
>> [n,m] = size(M)
```

Utilisation de la fonction length sur une matrice donne la plus grande dimension :

```
>> dim = length(M)
```

Pour appliquer les opérations de base sur les matrices on va procéder comme suit :

```
>> B*C        % produit matrice-matrice
>> B^2        % l'opération B*B
>> B\C        % les divisions matricielles
>> inv(B)*C   % qui est équivalent à B\C
>> b/B        % qui est équivalent à b*inv(B)
```

Noter que les fonctions scalaires courantes, (sin, exp, etc...) peuvent aussi s'appliquer à des matrices, composante par composante, comme dans l'exemple suivant:

```
>>u=[0:1:4]
u =  0     1     2     3     4
>>v=sin(u)
v =  0    0.8415    0.9093    0.1411   -0.7568
```

Graphisme :

Etant donné deux vecteurs de même taille, x et y, la fonction plot(x,y) trace le graphe de y en fonction de x. En fait Matlab relie les points de coordonnées (x(k),y(k)) pour $1 \leq k \leq$ length (x). En prenant un grand nombre de points dans le vecteur x et en définissant en suite y=f(x) pour une certaine fonction f, la fonction plot(x,y) nous donnera le graphe de la fonction f.

```
>> x=[0:0.01:4*pi];
>> y=cos(x);
>> plot(x,y)
```

On s'aperçoit que ces commandes ne tracent pas deux graphes, mais un seul. En fait, le deuxième plot(x,z) vient remplacer le premier plot(x,y). Pour remédier à cela, Matlab propose plusieurs méthodes suivant si l'on désire que les courbes apparaissent dans une ou plusieurs fenêtres. Pour voir les graphiques sur deux fenêtres, il suffit de dire à Matlab de construire une nouvelle fenêtre avec la commande figure.

```
>> z=sin(x);
>> plot(x,y)
>> plot(x,z)
```

Pour avoir les deux courbes dans la même fenêtre, on va procéder comme suit :

```
>> plot(x,y)
>> figure
>> plot(x,z)
```

```
>> plot(x,y,x,z)
```

Les couleurs et le style du tracé peuvent également être modifiés. Pour cela, il suffit d'ajouter à plot une chaîne de caractères spécifiant le style; voir **help plot** pour toutes les possibilités.

```
>> x=[0.5:0.1:5];y=log(x);
>> plot(x,y,'co')
>>
>> x=[0:0.05:1];y=exp(x);z=log(y);
>> plot(x,y,'mo',y,z,'g>')
```