TP 6 : Calcul Matriciel

Création de matrices

Une matrice c'est un tableau à n lignes et p colonnes; pour créer une matrice sur Matlab on commence par remplir entre deux crochets les p positions de chaque ligne par des nombres ou des fonctions séparés par des virgules ou des blancs; les n lignes sont séparées par des points-virgules.

Exemple

```
>> m1 = [ 1 2 3 ; 4 5 6 ; 7 8 9]          % on sépare les lignes par des point-virgules
m1 =
     1     2     3
     4     5     6
     7     8     9
```

La variable m1 est donc une matrice de dimension 3x3.

- Une matrice avec une seule ligne ou une seule colonne est un vecteur , et une matrice 1x1 est un scalaire.
- Les éléments d'une matrice peuvent être référencés par leurs indices placés entre parenthèses.
- La commande size(A) fournit le nombre de lignes et le nombre de colonnes de A.

On peut étendre aux matrices les autres manières de définir des vecteurs.

L'opérateur deux points

Il sert fondamentalement à construire un vecteur dont les valeurs des éléments sont incrémentées séquentiellement.

Exemple

```
>> x = 3:9
```

x=

3 4 5 6 7 8 9

L'incrément est de 1 par défaut. Pour avoir un autre incrément, tapez des commandes telles que :

>> x = 1:0.5:4

>> x = 6:-1:0

La plupart des fonctions Matlab acceptent des entrées vectorielles et produisent des sorties vectorielles.

Exemple

>> y = sqrt(1:10)

→ Construit un vecteur d'entiers de 1 à 10 et prend la racine carrée de chacun d'entre eux.

Matrices particulières

>> ones(3) % ones(n,m) renvoie une matrice à n lignes et m colonnes dont tous les coefficients sont des 1

ans =

1 1 1

1 1 1

1 1 1

>> zeros(2,5) % zeros(n,m) matrice à n lignes et m colonnes dont tous les coefficients sont des 0 ; elle sert beaucoup pour les initialisations.

ans =

0 0 0 0 0

0 0 0 0 0

>> eye(4) % renvoie la matrice identité

```
ans =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>> diag([1 : 4])
ans =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
>> rand(1,7)          % nombres aléatoires entre 0 et 1
ans =
0.9355    0.9169    0.4103    0.8936    0.0579    0.3529    0.8132
>> logspace(0, 2, 11) % crée un vecteur à 11 composantes entre 100 et 10 2
>> linspace(0, pi, 11) % génère un vecteur (11 valeurs. réparties de 0 à pi)
```

Manipulation des Matrices

Les dimensions d'une matrice ou d'un vecteur peuvent être augmentées en introduisant de nouveaux éléments

Exemples :

```
>> x = [1 3 5]
>> x = [x 6 8 10]
x=
     1     3     5     6     8    10
```

```
>> y = [x;1:6]

y=

     1     3     5     6     8    10

     1     2     3     4     5     6

>> m2 = [1:1:3 ; 11:1:13]

m2 =

     1     2     3

    11    12    13

>> m3 = [1:1:3 ; logspace(0, 1, 3)]

m3 =

    1.0000    2.0000    3.0000

    1.0000    3.1623   10.0000
```

Extraction d'une sous-matrice

On peut aussi utiliser les deux points pour extraire une sous-matrice d'une matrice A. A(:,j) extrait la jème colonne de A. On considère successivement toutes les lignes de A et on choisit le jème élément de chaque ligne.

- A(i,:) extrait la ième ligne de A.
- A(:) reforme le matrice A en un seul vecteur colonne en concaténant toutes les colonnes de A.
- A(j:k) extrait les éléments j à k de A et les stocke dans un vecteur ligne
- A(:,j:k) extrait la sous-matrice de A formée des colonnes j à k.
- A(j:k,:) extrait la sous-matrice de A formée des lignes j à k.
- A(j:k,q:r) extrait la sous-matrice de A formée des éléments situés dans les lignes j à k et dans les colonnes q à r.

Ces définitions peuvent s'étendre à des pas d'incrémentation des lignes et des colonnes différents de 1.

Transposition :

l'opérateur apostrophe (') utilisé pour créer un vecteur colonne est en fait l'opérateur transposition: sert à désigner la transposée d'une matrice. Les lignes de A' sont les colonnes de A, et vice versa. Si A est une matrice complexe, A' est sa transposée conjuguée. Pour obtenir une transposée non conjuguée, il faut employer les deux caractères point-prime (.').

Exemples :

```
>> A = [1 2 3;4 5 6;7 8 9]'     % produit la matrice
A=
     1     4     7
     2     5     8
     3     6     9
>> m2'
ans =
     1    11
     2    12
     3    13
```

Opérations scalaires-matrices

Additions +, soustractions -, multiplications * , puissances ^

Une telle opération agit sur chaque élément de la matrice:

Exemples :

```
>> x = [1 2 3 4], x = x-1
x=
```

```
        1     2     3     4

x=

        0     1     2     3

>> m2' * 10          % de même:  4*m2  m2-10  m2/4

ans =

        10    110
        20    120
        30    130
```

Une exception:

```
>> m2^2

??? Error using ==> ^

Matrix must be square.
```

Dans ce cas, Matlab veut calculer le produit matriciel m2 * m2

La solution est l'usage du point qui force l'opération sur chaque élément:

```
>> m2 .^ 2

ans =

        1     121
        4     144
        9     169
```

Opérations entre matrices

Attention aux dimensions : on ne peut ajouter ou soustraire que des matrices de même taille.

Multiplications

La multiplication de deux matrices n'a de sens que si leurs dimensions ‘’internes’’ sont égales.

$$(A * B)_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

où n est le nombre de colonnes de A et aussi le nombre de lignes de B.

Exemples :

```
>> A = [1 2 3;4 5 6]; B = [7;8;9]; A*B
ans =
        50
        112
>> m1 * m2'          % le produit matriciel n’est possible que lorsque les dimensions
                          sont cohérentes
ans =
        14      74
        32      182
        50      290
>> m1 * m2
??? Error using ==> *
Inner matrix dimensions must agree.
```

L’operateur « point »

En calcul matriciel, on a souvent besoin d'effectuer une opération élément par élément. Dans Matlab, ces opérations sont appelées opérations sur des réseaux ou des tableaux (array operations). Bien entendu, l'addition et la soustraction sont des opérations qui se font élément par élément. L'opération A.*B désigne la multiplication, élément par élément, des matrices A et B.

Exemples :

```
>> A = rand(3)
>> A_square = A.^2
```

Où le point indique qu'une opération doit être effectuée sur chaque élément de A. Sans ce point, A est multipliée par A suivant les règles usuelles de la multiplication des matrices, ce qui conduit à un résultat totalement différent

```
>> A^2
>> 2.^A % pour l'exponentielle de matrice
```

Multiplication élément par élément:

```
>> m2 .*m3    % (m2 et m3 ont les mêmes dimensions)
ans =
    1.0000    4.0000    9.0000   11.0000   37.9473  130.0000
```

Divisions

```
>> m2/m3       % division matricielle
ans =
    1.0000   -0.0000
    9.5406   -1.5960
```

Division élément par élément:

```
>> m2./m3      % chaque élément de m2 est divisé par l'élément équivalent de m3
ans =
    1.0000    1.0000    1.0000   11.0000    3.7947    1.3000
>> m2.\m3     % chaque élément de m3 est divisé par l'élément équivalent m2
ans =
    1.0000    1.0000    1.0000
```

0.0909 0.2635 0.7692

>> m3./m2 % chaque élément de m3 est divisé par l'élément équivalent m2

ans =

1.0000 1.0000 1.0000

0.0909 0.2635 0.7692

Les fonctions vectorielles et matricielles

Fonctions élémentaires

Les fonctions telle que abs, sin, cos, log, exp s'appliquent à des vecteurs ou matrices de taille quelconque. Les opérateurs max et min rendent un vecteur ligne contenant le max (min) de chaque colonne. Les opérateurs logiques rendent des matrices de coefficient 0 ou 1.

Exemple :

>> a=rand(3,6), (a>0.2) & (a<0.8)

Produits internes et externes

Le produit interne, ou scalaire, de deux vecteurs colonnes x et y est le scalaire défini comme le produit x'*y ou, de manière équivalente y'*x

Exemple :

>> x = [1;2],y = [3;4], x'*y

ans =

11

Le produit externe, ou antiscalaire, de deux vecteurs colonnes est la matrice antiscalaire x*y'. De même, le produit externe de deux vecteurs lignes est la matrice x'*y.

Fonctions vectorielles

Ces fonctions sont plutôt destinées à agir sur des vecteurs, lignes ou colonnes. Citons par exemple

:max, sum, mean, sort, min , prod, std...dont vous trouverez l'objet par la commande help lorsqu'il n'est pas évident. Elles agissent aussi sur les matrices, mais colonne par colonne.

exemple :

```
>> b = rand(3,3) ; c = max(b) ; d = max(max(b));
>> b,c,d
>> sort(c)
>> sort(b)
```

Fonctions matricielles

- **eig** : valeurs et vecteurs propres d'une matrice
- **expm** : exponentielle de matrice,
- **norm** : norme
- **rank** : rang.
- **inv** : L'inverse d'une matrice (inv(A) n'est valide que si A est carrée.)

Caractéristiques des matrices

```
>> size(m3)     % dimensions
ans =
        2        3
>> length(m3)            % equivalent à max(size(m3)) : dimension maximum
ans =
        3
```

Fonctions de manipulation des matrices

```
>> A = [1 2 3 ; 4 5 6 ; 7 8 9 ]
A =
```

```
     1     2     3
     4     5     6
     7     8     9
>> flipud(A)          % flip up-down
ans =
     7     8     9
     4     5     6
     1     2     3
>> fliplr(A)          % flip left-right
ans =
     3     2     1
     6     5     4
     9     8     7
>> rot90(A,2)         %2 rotations de 90 degres (sens trigo)
ans =
     9     8     7
     6     5     4
     3     2     1
>> reshape(A,1,9)     % change la forme de la matrice
ans =
     1     4     7     2     5     8     3     6     9
>> diag(A)            % extrait la diagonale de A
ans =
     1
```

```
        5
        9
>> diag (ans)          % diag travaille dans les 2 sens !
ans =
        1     0     0
        0     5     0
        0     0     9
>> triu(A)             % extrait le triangle supérieur de A
ans =
        1     2     3
        0     5     6
        0     0     9
>> tril(A)      % triangle inférieur
ans =
        1     0     0
        4     5     0
        7     8     9
```

Matrices clairsemées

Lorsque seulement quelques éléments d'une matrice sont non-nuls, on peut la définir comme une sparse matrix. Sa description contient seulement les éléments non nuls.

```
>> A_normal = [0 1 0 ; 1 0 0; 0 0 1] % matrice normale
A_normal =
        0     1     0
```

```
    1     0     0

    0     0     1
```

```
>> A_sparse = sparse(A_normal) % matrice clairsemée

A_sparse =

    (2,1)         1

    (1,2)         1

    (3,3)         1
```

sparse peut aussi être utilisé pour créer directement une matrice clairsemée:

```
>> S = sparse([2 1 3 4], [1 2 3 1], [4 5 6 7], 4, 3)

S =

    (2,1)         4

    (4,1)         7

    (1,2)         5

    (3,3)         6
```

Exercices

1) Créer une matrice A de nombre aléatoire de dimension 5x5.
2) Extraire la sous matrice A1 formée par les deux dernières lignes et les 3 premières colonnes .
3) Extraire la sous matrice A2 formée par les deux dernières lignes et toutes les colonnes
4) Extraire la sous matrice A3 formée par toutes les lignes et les 4 premières colonnes .
5) Exprimer la deuxième ligne de A .
6) Supprimer maintenant la première et la troisième colonne de A .
7) Soient les matrices suivantes :

$$A = \begin{bmatrix} 1 & 2 & 2 \\ 4 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} \quad \text{et} \quad B = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 2 & 3 \end{bmatrix}$$

Comparer les opérations :

- Ct=A.*B, et Cm=A.B
- Dt=A./B, et Dm=A.B
- Et=A.', et Em=A'
- Ct=A.^2, et Fm=A^2

8) Créer une matrice B1= diag(-m:m) avec m une valeur que vous choisissez.

9) Créez B2 en prenant les lignes de A en sens inverse