2. Les Matrices

Tous les variables créés dans Matlab sont des matrices. Une matrice est un arrangement des éléments qui sont tous accessibless par des indices. Comme dans tous les langages de programmation, on associe des noms aux variables créés. En Matalb, les noms de variables (ou bien des matrices) s'écrivent

- sur 31ou 63 (selon la versio) caractères au max dont la première est une lettre
- MATLAB différencie les majuscules des miniscules

MATLAB travaille en double précision

2.1 Syntaxe de l'indexation

A(i,j) est un élément de la matrice A qui se trouve sur la i-ème ligne et dans la j-ème colonne. **Dans MATLAB les lignes sont presque toujours prioritaires**.

A= [a_{11}, a_{12}; a_{21} a_{22}]

- les éléments d'une ligne sont separés par des blancs ou des virgules
- les éléments d'une colonne sont separés par un point virgule
- le tout est mis entre [].Les [] sont l'opérateur de **concaténation**

2.2 Créationde différents types de matrices

Exemples

- scalaire >> c = 5;
- vecteur ligne -

  i) >> v = [1 2 3 4 5];

  ii) v = deb: pas: fin

  >>v = 0: 0.01:20 ;

Ainsi on peut recréer le vecteur temps pour des données échantillonnées à 100 Hz

```
>> h = 1:10          (un pas de 1 est pris par défaut)
h =
1   2   3   4   5   6   7   8   9   10
```

- tableau >> T = [1 2 3; 4 5 6; 7 7 7];
- matrices spéciales

magic	Carré magique
zeros	Matrice de 0
ones	Matrice de 1
rand	Matrice de nombres aléatoires : distribution uniforme (0<n<1)
randn	Matrice de nombres aléatoires : distribution normale
eye	Matrice identité

```
» A = magic(5)
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

» R = rand(2,3)
R =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621

» Rn = randn(2,3)
Rn =
   -0.4326    0.1253   -1.1465
   -1.6656    0.2877    1.1909

» I =eye(4)
I =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

2.3 Manipulation des indices

- **création de la matrice A**

» A = [1 2 3;4 5 6;7 8 9];

La matrice W n'existe pas. MATLAB crée une matrice de taille(3,4) dont le seul élément non nul est égal à 5

```
» W (3,4) = 5
W =
     0     0     0     0
     0     0     0     0
     0     0     0     5
```

- **création du vecteur de 5 éléments**

```
» v = [-4.5 sqrt(-5) rand(1,3)]
v =
  Columns 1 through 4
  -4.5000                 0+ 2.2361i   0.4565            0.0185
  Column 5
   0.8214
```

- **lecture d'un élément**

```
» a1 = A(1,3)
a1 =
     3
```

- **lecture d'une ligne**

```
» a2 = A(1,:)
a2 =
     1     2     3
```

- **lecture de certains éléments d'une ligne**

```
» a3 = A(2,[1 3])
a3 =
     4     6
```

- **lecture d'une colonne**

```
» a4 = A(:,3)
a4 =
     3
     6
     9
```

- **convertir en colonne**

```
» acol = A(:)        =>   acol = A(1:1:9)  ou acol = A(1:1:end)
acol =
     1
     4
     7
     2
     5
     8
     3
     6
     9                        ATTENTION a l'ordre des éléments
```

- **suppression d'une ligne**

```
» A(2,:) = []
A =
     1     2     3
     7     8     9
```

- **suppression des elements**

On ne peut pas enlever des éléments individuels d'une matrice

```
» A = [1 2 3;4 5 6;7 8 9];
```

```
» A(1,2) = []
???  Indexed empty matrix assignment is not allowed.
```

Mais si on permet la conversion de la matrice en vecteur, une commande comme la suivante permet de reformer les éléments en vecteur de ligne.

```
» A(2:2:9) = []
A =
     1     7     5     3     9
```

- **ajouter une ligne** (concaténation verticale)

```
»  b= [1 2 3];
» A = [A; b]
A =
     1     2     3
     7     8     9
     1     2     3
```

- **ajouter un colonne** (concaténation horizontale)

```
» c=[10;11;12];
»  A = [A, c]
A =
     1     2     3    10
     7     8     9    11
     1     2     3    12
```

- **concaténation des matrices**

```
» A = [1 2 3;4 5 6;7 8 9];
» B = [A A+10; A-1 A]
B =
     1     2     3    11    12    13
     4     5     6    14    15    16
     7     8     9    17    18    19
     0     1     2     1     2     3
     3     4     5     4     5     6
     6     7     8     7     8     9
```

- **les fonctions length et size**

```
» A = rand(4,6);
» size(A)
ans =
     4     6
» length(A)
```

la commande **length** est plutôt prévue pour déterminer la longeur des vecteurs. Comme A est une matrice elle sort la valeur max de **size**.

```
ans =
     6
```

```
» length(A(:))      on crée un vecteur des valeurs de A
ans =
   24
```

Exercises

1. Créez un vecteur de 12 éléments pour extraire les éléments imapaires dans un autre vecteur

2. Créez une matrice A de (3,4) et un vecteur ligne de 4 éléments. Ajoutez la ligne entre les lignes 1 et 2 de A

3. Créez une matrice de 6 lignes et 13 colonnes et extraire dans une autre matrice les colonnes paires de la matrice d'origine

2.4 Transformations

triu	Partie triangulaire supérieure
tril	Partie triangulaire inférieure
diag	Extraire la diagonale/création de la matrice diagonale
rot90	Rotation de 90 degrés
flipud	Symétrie axiale horizontale - haut-bas
fliplr	Symétrie axiale verticale - gauche-droite

- **Transposée**

```
» A= rand(5,3)
A =
   0.4447    0.1763    0.8936
   0.6154    0.4057    0.0579
   0.7919    0.9355    0.3529
   0.9218    0.9169    0.8132
   0.7382    0.4103    0.0099
» A'                                    la transposée de A
ans =
   0.4447    0.6154    0.7919    0.9218    0.7382
   0.1763    0.4057    0.9355    0.9169    0.4103
   0.8936    0.0579    0.3529    0.8132    0.0099
```

(Attention si les valeurs sont complexes………

```
» F=randn(3,2) + randn(3,2)*i
F =
  -0.6918- 0.3999i  -1.5937+ 0.7119i
   0.8580+ 0.6900i  -1.4410+ 1.2902i
```

```
  1.2540+ 0.8156i   0.5711+ 0.6686I
» F2 = F'                          transposée avec le conjugué
F2 =
 -0.6918+ 0.3999i   0.8580- 0.6900i   1.2540- 0.8156i
 -1.5937- 0.7119i  -1.4410- 1.2902i   0.5711- 0.6686i

» F3 = F.'                         transposée sans le conjugué
F3 =
 -0.6918- 0.3999i   0.8580+ 0.6900i   1.2540+ 0.8156i
 -1.5937+ 0.7119i  -1.4410+ 1.2902i   0.5711+ 0.6686I …… )
```

3.5 Opérations sur les matrices

- **Addition**

```
» A = [1 2 3;4 5 6;7 8 9];
» B = A'
B =
     1     4     7
     2     5     8
     3     6     9

» C= A + B            les matrices sont de taille identique
C =
     2     6    10
     6    10    14
    10    14    18

» D = C + 100         le scalaire est ajouté à chaque élément
D =
   102   106   110
   106   110   114
   110   114   118
```

- **Multiplication**

```
» AB = A*B            le produit matriciel - ATTENTION aux tailles des matrices
AB =
    14    32    50
    32    77   122
    50   122   194
```

- **Puissance**

```
»  A = [1 2 3;4 5 6;7 8 9];
» A^2
ans =
    30    36    42
    66    81    96
```

```
  102   126   150
```

- **Division**

Le symbole pour la division 'standard' est le / .

(Pour plutôt les matheux, il existe aussi le \ . Les deux formes de division correspondent aux cas suivants:

X = A\B est la solution de A*X = B

Y = W/Z est la solution de Y*Z = W

Exemple : la résolution d'un système d'équations

$$
\begin{aligned}
x + 2y &= 5 \\
3x + 4y &= 6
\end{aligned}
$$

sol = [x; y]

donc, on peut définir les matrices A et B comme A*sol = B, ou

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

donc….

```
» sol = A\B

sol =
  -4.0000
   4.5000
```

La matrice A étant inversible, la solution peut être également obtenue de la manière suivante.

```
» inv(A)
ans =
  -2.0000    1.0000
   1.5000   -0.5000

» sol = inv(A) * B
sol =
  -4.0000
   4.5000
```

Si on compare les deux méthodes, la première est la meilleure numériquement comme elle peut éviter le calcul de l'inverse d'une matrice mal conditionnée…..)

2.6 Les Opérations Terme à Terme

2.6.1 Opérations arithmétiques

Il est possible de réaliser des opérations arithmétiques terme à terme entre matrices. Les matrices sont alors assimilées à des tableaux.

Les deux opérateurs + et - travaillent déjà terme à terme. Pour les autres opérateurs *, /, \ et ^, un point est mis devant l'opérateur pour préciser que les opérations se font terme à terme

Exemple:

```
» x = [1 2 3]; y = [4 5 6];
» z= x.*y
z =
     4    10    18

» z = x.\y                  (identique à  z = y./x)
z =
    4.0000    2.5000    2.0000
```

2.6.2 Opérations relationnels et logiques

< >	Inférieur et supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal
==	Egal
~=	Différent
&	ET logique
¦	OU logique
~	Complément logique (NOT)

Exemples:

```
» A = rand(4,3)                     création de la matrice A
A =
    0.9501    0.8913    0.8214
    0.2311    0.7621    0.4447
    0.6068    0.4565    0.6154
    0.4860    0.0185    0.7919

» B= (A>0.2)&(A<0.8)               recherche des éléments de A supérieurs
                                   à 0.2 et inférieurs à 0.8
```

```
B =
     0     0     0
     1     1     1
     1     1     1
     1     0     1
```

» C = A.*B — dans C sont stockés les valeurs qui remplissent la condition

```
C =
        0         0         0
   0.2311    0.7621    0.4447
   0.6068    0.4565    0.6154
   0.4860     0         0.791
```

Exercises

1. Mettre les éléments de A (déjà définie) qui sont inférieures à 0.2 à zero.

2. Recherchez des éléments dans la matrice A(déjà définie) qui sont différents du premier élément de la matrice A.

2.6.3 Fonctions mathématiques

sin, asin,sinh,asinh cos, tan…….	Fonctions trigonométriques
exp, log,log10,sqrt	Fonctions exponentielles
abs, angle, conj, imag, real	Fonctions complexes
fix, floor, ceil, round, rem,sign	Fonctions numériques

Exemples:

	0.8	-2.3	-3.7	
round	1	-2	-4	
fix	0	-2	-3	(en direction 0)
floor	0	-3	-4	(en direction -inf)
ciel	1	-2	-3	(en direction +inf)

2.6.4 Fonctions logiques pour déterminer des caractéristiques

any	Egal 1 si l'un des éléments est différent de 0	(colonne par colonne)
all	Egal 1 si tous les éléments sont différents de 0	(colonne par colonne)
find	Donne les indices des éléments différents de 0	
isnan	Egal 1 si l'élément est égal à NaN	
isinf	Egal 1 si l'élément est égal à Inf	

Exemples:

```
»  A = [0 0 6;0 0 4;0 1 2];
» any(A)
ans =
    0     1     1
» all(A)
ans =
    0     0     1
» find(A)
ans =
    6
    7
    8
    9
```

Exemples de valeurs infinies et de valeurs non définies:

```
» x= [0 0 6]; y= [0 1 2];
» d = y./x
Warning: Divide by zero.
d =
      NaN       Inf    0.3333          NaN (Not a Number) = élément non défini
                                       Inf = élément infini
» e= 1./d                              MATLAB travaille sur les NaN et les Inf
e =
  NaN     0     3
» f = d-d
f =
  NaN   NaN     0
» isinf(d)
ans =
    0     1     0
» isnan(d)
ans =
    1     0     0
```

2.6.5 Analyse de données

Les fonctions suivantes travaillent sur des vecteurs. Lorsqu'on leur applique une matrice en entrée, elles travaillent indépendamment sur chacune des colonnes . ie elles travaillent sur la première dimension qui n'est pas un singleton

max, min	maximum, minimum
sort	classement par ordre croissant
mean,std	moyenne et écart type
sum	somme des éléments
prod	produit des éléments
diff	différence entre les valeurs consécutives
gradient	gradient numérique
sign	1 pour positive, 0 pour zéro, -1 pour negative

```
» A = rand(1,5)
A =
   0.9218    0.7382    0.1763    0.4057    0.9355

» max(A)                max du vecteur A
ans =
   0.9355

» A = rand(6,3)
A =
   0.9169    0.0099    0.1988
   0.4103    0.1389    0.0153
   0.8936    0.2028    0.7468
   0.0579    0.1987    0.4451
   0.3529    0.6038    0.9318
   0.8132    0.2722    0.4660


» max(A)
ans =
                0.9169   0.6038    0.9318

» [m,i] = max(A)                 on peut aussi sortir les indices des éléments max
m =
          0.9169    0.6038    0.9318
i =
          1     5     5
```

Exercises :

1. Comment peut-on touver la valeur moyenne de toute la matrice A ?

2. Comment déterminer les max des lignes ?

3. La commande **std** donne les écart-types colonne par colonne. Vérifiez que c'est bien le cas avec une matrice comme A. Comment peut-on trouver l'écart-type de toutes les valeurs dans A. ?

4. Si x est le vecteur rand(10,1) calculez la somme des x carrés

5. Si A = rand(4), calculez le nombre d'éléments de A supérieur à 0.5.

les commandes `diff` et `gradient`

» D = rand(1,20); pour montrer une différence entre diff et gradient
» df = diff(D);
» gr = gradient(D);
» whos

Name	Size	Bytes	Class
D	1x20	160	double array
df	1x19	152	double array
gr	1x20	160	double array