Chapitre 8

Tableaux multidimensionnels – Cellules et Structures

I. Tableaux multidimensionnels

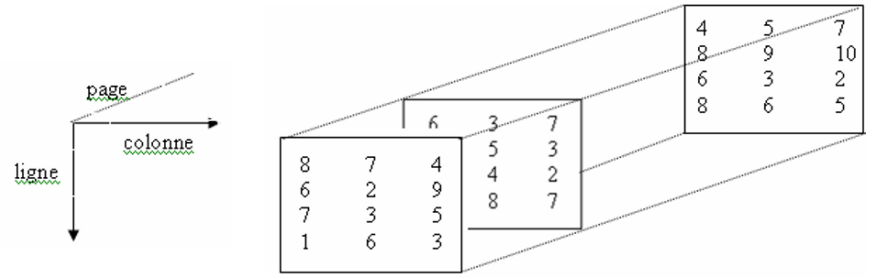
I.1. Définition et génération d'un tableau multidimensionnel

I.1.1. Définition

MATLAB, langage orienté objet, admet des tableaux à plusieurs dimensions, supérieures à 2 pour le cas particulier des matrices.

Un tableau à 3 dimensions est formé de « pages », elles-mêmes constituées de matrices (tableaux à 2 dimensions), avec le même nombre de lignes et de colonnes.

N. Martaj, M. Mokhtari, *MATLAB R2009, SIMULINK et STATEFLOW pour Ingénieurs, Chercheurs et Etudiants*, DOI 10.1007/978-3-642-11764-0_8, © Springer-Verlag Berlin Heidelberg 2010



I.1.2. Création d'un tableau multidimensionnel

Par définir ce tableau, nous utilisons la commande `cat` qui réalise la concaténation des pages selon la dimension 3.

```
>> page1=[8 7 4;6 2 9;7 3 5;1 6 3] ;
>> page2=[6 3 7;4 5 3; 1 4 2;3.14 8 7];
>> page3=[4 5 7;8 9 10;6 3 2;8 6 5];
>> cat(3, page1,page2,page3)
```

```
>> tableau1=cat(3, page1,page2,page3)
tableau1(:,:,1) =
     8     7     4
     6     2     9
     7     3     5
     1     6     3

tableau1(:,:,2) =
    6.0000    3.0000    7.0000
    4.0000    5.0000    3.0000
    1.0000    4.0000    2.0000
    3.1400    8.0000    7.0000

tableau1(:,:,3) =
     4     5     7
     8     9    10
     6     3     2
     8     6     5
```

L'affichage se fait par page, le signe ' :' indique « toutes » les lignes et toutes les colonnes de chaque page.

Pour accéder à l'élément de la 2ème page, 4ème ligne et 1ère colonne :

```
>> tableau1(4,1,2)
ans =
    3.1400
```

On peut créer un tableau multidimensionnel sans utiliser la commande `cat`, en spécifiant directement la page comme dans le cas suivant :

```
>> tableau2(:,:,1)=[2 6;4 5]
tableau2 =
     2     6
     4     5
```

```
>> tableau2(:,:,2)=[5 7;8 3]
tableau2(:,:,1) =
     2     6
     4     5

tableau2(:,:,2) =
     5     7
     8     3
```

On peut étendre un tableau en spécifiant directement les valeurs de la nouvelle page.

```
>> tableau2(:,:,3)=[2 4;1 7]
tableau2(:,:,1) =
     2     6
     4     5

tableau2(:,:,2) =
     5     7
     8     3

tableau2(:,:,3) =
     2     4
     1     7
```

Un tableau à 4 dimensions est un tableau de tableaux.
Les éléments de la 4ème dimension de `tableau3` sont formés de `tableau1` et d'un tableau aléatoire de mêmes dimensions que `tableau1` :

```
>> tableau3(:,:,:,1)=tableau1;
>> tableau3(:,:,:,2)=randn(size(tableau1)) ;
```

```
>> size(tableau3)
ans =
     4     3     3     2
```

Nous pouvons aussi utiliser la commande `cat` en précisant la concaténation selon une autre dimension comme la 4ème dimension, par exemple.

```
>> tableau4=cat(4, tableau1, randn(size(tableau1))) ;
```

Nous pouvons vérifier l'égalité de `tableaux3` et `tableau4` si les valeurs de `randn` sont toujours les mêmes à chaque réalisation.

```
>> all(all(all((tableau3==tableau4))))

ans(:,:,1,1) =
     1

ans(:,:,1,2) =
     0
```

L'égalité est réalisée uniquement pour les éléments de la 1ère page ne contenant par la fonction `randn`.

Comme pour les matrices, on peut créer des tableaux particuliers en utilisant les commandes `ones`, `randn`, `zeros`, etc.

```
>> taille=size(tableau1);
>> tableau5=ones(taille)+randn(taille);
```

Nous créons un tableau de mêmes dimensions que `tableau1`, formés de 1 auxquels on a ajouté des valeurs aléatoires normales, dont nous affichons la 2ème page.

```
>> tableau5(:,:,2)

ans =
    3.9080    0.5314    1.7015
    1.8252    0.7275   -1.0518
    2.3790    2.0984    0.6462
   -0.0582    0.7221    0.1764
```

La commande `ndims` donne le nombre de dimensions qui est 4 pour `tableau3`.

```
>> ndims(tableau3)
ans =
     4
```

La commande `whos` qui liste les variables de l'espace de travail donne :

```
>> whos
  Name          Size              Bytes  Class     Attributes

  ans           1x1                   8  double
  page          0x0                   0  double
  page1         4x3                  96  double
  page2         4x3                  96  double
  page3         4x3                  96  double
  tableau1      4x3x3               288  double
  tableau2      2x2x3                96  double
  tableau3      4-D                 576  double
  taille        1x3                  24  double
```

Pour supprimer la 2ème colonne dans toutes les pages de `tableau3`, on doit la remplacer par un ensemble vide.

```
>> tableau3(:,2,:)=[]
```

La première page de ce nouveau tableau donne :

```
>> tableau3(:,:,1)
ans =
     8     4
     6     9
     7     5
     1     3
```

On peut accéder à une partie du tableau en indexant les lignes, les colonnes et les pages correspondantes.

I.1.3. Extraction d'un sous-tableau

```
>> subtableau3=tableau3(1:2,1,1:2)

subtableau3(:,:,1) =
     8
     6

subtableau3(:,:,2) =
     6
     4
```

Dans ce cas on ne considère que les 2 premières lignes, uniquement la 1ère colonne, des pages 1 à 2. Le tableau est uniquement constitué d'un tableau à 3 dimensions à 2 pages formées de 2 vecteurs. Les conditions sur les valeurs des éléments d'un tableau des tableaux de mêmes dimensions formés de 1 et 0 aux index des éléments où la condition est respectivement vraie et fausse.

```
>> [i,j,k]=find(tableau5>3)
i =
     3
     3

j =
     3
     4
k =
     1
     1
```

On trouve uniquement 2 éléments qui vérifient cette condition, à savoir les éléments (3,3) et (3,4) de la 1ère page.

On peut créer un tableau formé de ces éléments en indexant directement `tableau5` à l'aide de la commande `find`.

```
>> tableau6=tableau5(find(tableau5>3))
tableau6 =
    3.5784
    3.0349
```

Le `tableau6` est alors une matrice à 2 lignes et 1 colonne.

```
>> size(tableau6)
ans =
     2     1
```

On peut transformer n'importe quel tableau en un seul vecteur colonne en l'indexant par le signe « : ». Si on veut calculer la moyenne de tous les éléments de `tableau5`, nous avons plusieurs possibilités.

I.1.4. Opérations sur les tableaux

I.1.4.1. Fonctions de tableaux

```
>> moy1=mean(tableau5)
moy1(:,:,1) =
    0.0376   -0.0422
moy1(:,:,2) =
    1.1625    1.4848
moy1(:,:,3) =
    0.4590    0.3869
```

Le calcul se fait d'abord par page en calculant la moyenne de chacune de ses colonnes.

En combinant deux fois la commande `mean`, nous obtenons toujours 3 pages contenant la moyenne de tous les éléments de toutes les pages.

```
>> moy2=mean(moy1)
moy2(:,:,1) =
   -0.0023
moy2(:,:,2) =
    1.3237
moy2(:,:,3) =
    0.4229
```

Il faut combiner la commande `mean` en autant de dimensions du tableau pour avoir la moyenne de tous ses éléments.

```
>> moyenne=mean(mean(mean(tableau5)))
moyenne =
    0.5814
```

Le moyen le plus simple de calculer cette moyenne est de l'appliquer sur le vecteur colonne obtenu en indexant le tableau par l'opérateur « : ».

```
>> moyenne=mean(tableau5(:))
moyenne =
    0.5814
```

Ceci est valable pour toutes les commandes telles que `sum`, `prod`, etc.

```
>> size(tableau5(:))
ans =
    18     1
```

Le vecteur obtenu est de type colonne à 18 éléments.

I.1.4.2. Opérations élément par élément

Les opérations élément par élément peuvent s'opérer tableau par tableau, matrice par matrice, vecteur par vecteur ou élément par élément (singleton).

Certaines opérations ne s'appliquent que pour les tableaux à 2 dimensions (matrices) comme `det`, `eig`, etc.

```
>> a = rand(1,2,3,4,5);
>> eig (a)
??? Undefined function or method 'eig' for input arguments
of type 'double' and attributes 'full nd real'.
>> det(a)
??? Undefined function or method 'det' for input arguments
of type 'double' and attributes 'full nd real'.
```

Ces fonctions s'appliquent, par contre, à des pages (matrices) de ces tableaux.

```
>> c=cat(4,cat(3,[1 3;5 6],[2 6;8 9]), cat(3,[7 5; 4 7],[9
2;8 5]))
>> det(c(:,:,2,1))
ans =
   -30
>> inv(c(:,:,2,1))
ans =
   -0.3000    0.2000
    0.2667   -0.0667
```

L'application des fonctions telles que `sinc`, `cos`, `exp`, etc. à des tableaux multidimensionnels, donne des tableaux de mêmes dimensions.

```
>> sinc_a=sinc(a);
>> size(sinc_a)
ans =
     1     2     3     4     5
>> exp_2_sin_a=exp(2*sin(a));
>> size(exp_2_sin_a)
ans =
     1     2     3     4     5
```

I.1.5. Changement des dimensions d'un tableau

`tableau5` de dimensions (3, 2, 3) possède 18 éléments. Il peut être transformé ainsi en une matrice à 6 lignes et 3 colonnes, par exemple.

```
>> reshape(tableau5,[6,3])

ans =
    0.5377   -0.4336    0.7254
    1.8339    0.3426   -0.0631
   -2.2588    3.5784    0.7147
    0.8622    2.7694   -0.2050
    0.3188   -1.3499   -0.1241
   -1.3077    3.0349    1.4897
```

Ou en matrice à 3 lignes et 6 colonnes :

```
>> reshape(tableau5,[3,6])

ans =
    0.5377    0.8622   -0.4336    2.7694    0.7254   -0.2050
    1.8339    0.3188    0.3426   -1.3499   -0.0631   -0.1241
   -2.2588   -1.3077    3.5784    3.0349    0.7147    1.4897
```

On peut la transformer en n'importe quel tableau, pourvu que le nombre d'éléments reste inchangé, soit par exemple le tableau à 3 dimensions de 3 pages, chacune de 2 lignes et 3 colonnes.

```
>> reshape(tableau5,[2,3,3])
ans(:,:,1) =

    0.5377   -2.2588    0.3188
    1.8339    0.8622   -1.3077

ans(:,:,2) =
   -0.4336    3.5784   -1.3499
    0.3426    2.7694    3.0349
ans(:,:,3) =
    0.7254    0.7147   -0.1241
   -0.0631   -0.2050    1.4897
```

I.1.6. Permutation des dimensions d'un tableau

La commande `permute(tableau, dims)` permet de permuter les dimensions de `tableau` selon les dimensions spécifiées dans `dims`.

```
>> a = rand(1,2,3,4,5);

>> b=permute(a,[2,1,3,5,4]);
```

On passe du tableau « a » 5 dimensions au tableau « b » à 4 dimensions tout en respectant le même nombre total d'éléments. Le tableau « a » est formé de 5 tableaux eux-mêmes constitués de 4 tableaux à 3 dimensions (3 pages d'une ligne et 2 colonnes) et le tableau « b » sera formé de 4 tableaux constitués de 5 tableaux tridimensionnels à 3 pages de 2 lignes et 1 colonne.

```
>> size(a)
ans =
     1     2     3     4     5
```

```
>> size(b)
ans =
     2     1     3     5     4
```

On donne ici la première page des tableaux « `a` » et « `b` » de la dernière dimension 5 pour « `a` » et 4 pour « `b` ».

```
>> a(:,:,1,1,1)
ans =
    0.3685    0.6256

>> b(:,:,1,1,1)
ans =
    0.3685
    0.6256
```

Le retour au tableau d'origine se fait à l'aide de la commande `ipermute` avec les mêmes dimensions utilisées par la commande `permute`.

```
>> c=ipermute(b,[2,1,3,5,4]);

>> all(c(:)==a(:))
ans =
     1
```

Le tableau « `c` » obtenu par la commande `ipermute` est identique au tableau « `a` » d'origine.

I.1.7. Utilisation pratique des tableaux multidimensionnels dans l'industrie

Prenons l'exemple d'un banc de mesures dans l'industrie de l'automobile où l'on a souvent besoin de mesurer nombre de signaux issus de capteurs (température, pression, humidité, débit de fluide, etc.). Supposons qu'on fasse ces mêmes mesures pour plusieurs paramètres de l'environnement (ex. vitesse du moteur). Dans ce cas précis, chaque ligne du tableau représente les valeurs issues de ces capteurs à un instant t. Toutes les lignes comporteront ainsi toutes les mesures pour une vitesse du moteur. Chaque colonne représente la valeur d'un capteur pour tous les instants de l'essai. Pour une autre vitesse du moteur, les mesures sont enregistrées dans une autre page du tableau.

Pour la vitesse 1 :

Température (°C)	Débit (l/mn)	Pression (bar)
20.1	0.48	1.67
21.0	0.50	1.20
21.2	0.52	1.18
21.5	0.55	1.16
21.8	0.57	1.12
21.7	0.60	1.08
21.9	0.62	1.08
22.1	0.64	1.07
22.5	0. 67	1.04
22.8	0.68	1.02

Pour la vitesse 2:

Température (°C)	Débit (l/mn)	Pression (bar)
30	0.54	1.04
30.6	0.62	1.03
30.8	0.65	1.01
31	0.67	0.96
31.2	0.69	0.94
31.5	0.71	0.93
31.7	0.81	0.91
31.9	0.91	0.89
32.2	1.01	0.84
32.5	1.1	0.81

Le tableau multidimensionnel sera le suivant : 2 pages de 3 colonnes et 10 colonnes.

```
>> temp_vit1=[20.1 21 21.2 21.5 21.8 21.7 21.9 22.1 22.5
22.8]';
>> pression_vit1=[1.67 1.20 1.18 1.16 1.12 1.08 1.08 1.07
1.04 1.02]';
>> debit_vit1=[0.48 0.50 0.52 0.55 0.57 0.60 0.62 0.64 0.67
0.68]';
```

La concaténation selon la 2ème dimension se fera pour ces vecteurs colonnes.

```
>> vit1=cat(2,temp_vit1,pression_vit1,debit_vit1)
vit1 =
   20.1000    1.6700    0.4800
   21.0000    1.2000    0.5000
   21.2000    1.1800    0.5200
   21.5000    1.1600    0.5500
   21.8000    1.1200    0.5700
   21.7000    1.0800    0.6000
   21.9000    1.0800    0.6200
   22.1000    1.0700    0.6400
   22.5000    1.0400    0.6700
   22.8000    1.0200    0.6800
```

On fait de même pour les mesures concernant la vitesse 2 du moteur :

```
vit2 =
   30.0000    1.0400    0.5400
   30.6000    1.0300    0.6200
   30.8000    1.0100    0.6500
   31.0000    0.9600    0.6700
   31.2000    0.9400    0.6900
   31.5000    0.9300    0.7100
   31.7000    0.9100    0.8100
   31.9000    0.8900    0.9100
   32.2000    0.8400    1.0100
   32.5000    0.8100    1.1000
```

Le tableau multidimensionnel des 2 séries de mesures s'obtient par concaténation selon la dimension 3.

```
>> vit12= cat(3, vit1,vit2)

vit12(:,:,1) =
   20.1000    1.6700    0.4800
   21.0000    1.2000    0.5000
   21.2000    1.1800    0.5200
   21.5000    1.1600    0.5500
   21.8000    1.1200    0.5700
   21.7000    1.0800    0.6000
   21.9000    1.0800    0.6200
   22.1000    1.0700    0.6400
   22.5000    1.0400    0.6700
```

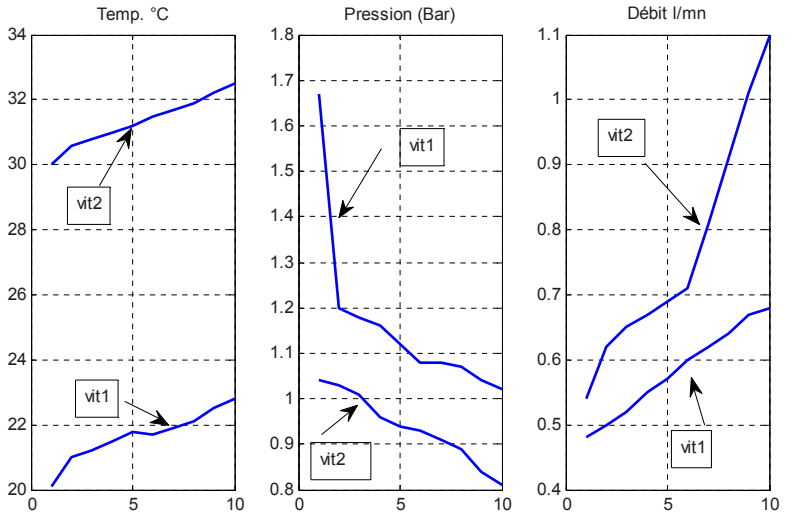
```
   22.8000    1.0200    0.6800

vit12 (:,:,2) =
   30.0000    1.0400    0.5400
   30.6000    1.0300    0.6200
   30.8000    1.0100    0.6500
   31.0000    0.9600    0.6700
   31.2000    0.9400    0.6900
   31.5000    0.9300    0.7100
   31.7000    0.9100    0.8100
   31.9000    0.8900    0.9100
   32.2000    0.8400    1.0100
   32.5000    0.8100    1.1000
```

On désire tracer la courbe des mesures de chaque capteur sur le même graphique pour visualiser l'effet de la vitesse du moteur sur cette mesure (température, pression, débit).

fichier mesures_capteurs.m

```
% Tracé des mesures de capteurs
vit1 =[20.1000    1.6700    0.4800
   21.0000    1.2000    0.5000
   21.2000    1.1800    0.5200
   21.5000    1.1600    0.5500
   21.8000    1.1200    0.5700
   21.7000    1.0800    0.6000
   21.9000    1.0800    0.6200
   22.1000    1.0700    0.6400
   22.5000    1.0400    0.6700
   22.8000    1.0200    0.6800];

vit2 =[30.0000    1.0400    0.5400
   30.6000    1.0300    0.6200
   30.8000    1.0100    0.6500
   31.0000    0.9600    0.6700
   31.2000    0.9400    0.6900
   31.5000    0.9300    0.7100
   31.7000    0.9100    0.8100
   31.9000    0.8900    0.9100
   32.2000    0.8400    1.0100
   32.5000    0.8100    1.1000];

vit12= cat(3, vit1,vit2)
subplot(1,3,1)

plot(1:10,vit12(:,1,1))
grid on
hold on
plot(1:10,vit12(:,1,2))
title('Temp. °C')
```

```
subplot(1,3,2)
plot(1:10,vit12(:,2,1))
hold on
plot(1:10,vit12(:,2,2))

title('Pression (Bar)'), grid on
subplot(1,3,3)
plot(1:10,vit12(:,3,1))
grid on
hold on
plot(1:10,vit12(:,3,2))
title('Débit l/mn')

% courbe de la pression en fonction de la température
hold off
figure(2)
plot(vit12(:,1,1), vit12(:,2,1),'s')
title('Pression en fonction de la température - Vitesse 1 du
moteur')
```

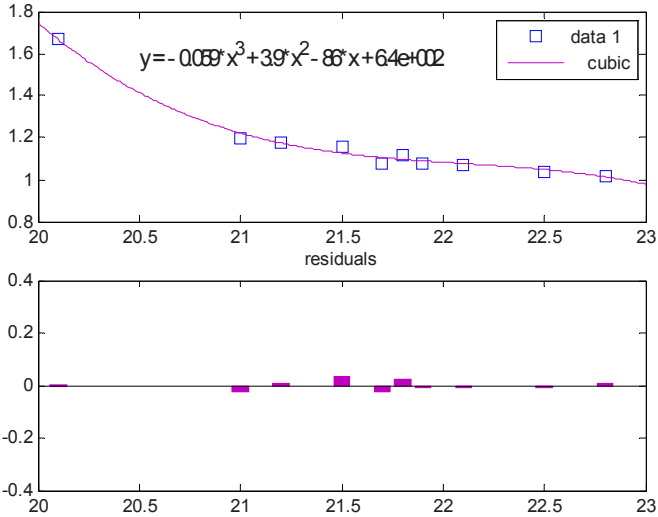


On réalise également la modélisation de la courbe donnant la pression en fonction de la température. On trace la courbe donnant la pression en fonction de la température.

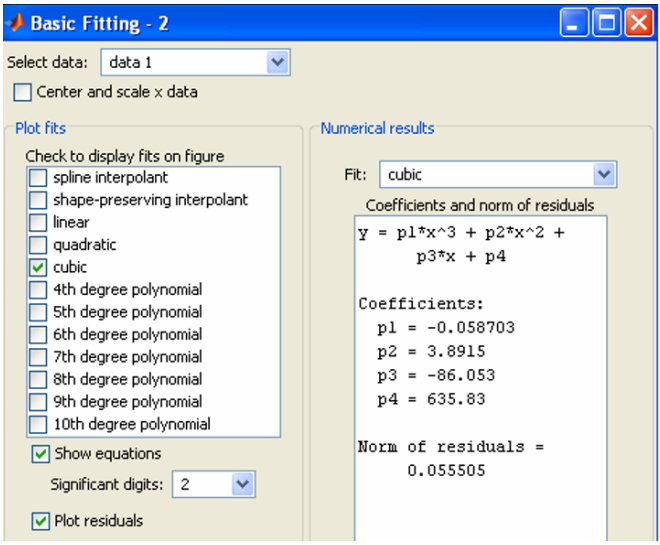
```
>> plot(vit12(:,1,1), vit12(:,2,1))
```

Le menu `Tools` de la fenêtre graphique possède l'outil de lissage de courbes `Basic Fitting`.

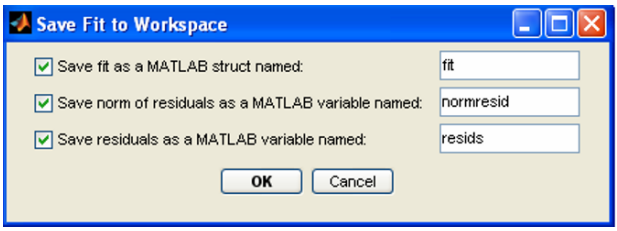
Cet outil permet de tracer la courbe du polynôme d'interpolation, d'afficher les coefficients du polynôme de lissage et de les sauvegarder sous une forme d'une structure.



On choisit un lissage cubique en cochant la case `cubic`, soit un polynôme d'interpolation du 3ème degré. En cochant la case `Show equations`, on obtient l'affichage des paramètres du coefficient avec le nombre de digits significatifs spécifié dans le menu `Significant digits`.



En cliquant sur le bouton `Save to workspace`, le polynôme est sauvegardé dans l'espace de travail MATLAB sous la forme d'une structure.



La structure aura pour nom `fit`, la norme des résidus sous le nom `normresid` et les résidus sous le nom `resids`.

Ces noms peuvent être, bien sûr, modifiés.

Dans le cas où on garde ces noms, nous avons :

```
>> fit
fit =
     type: 'polynomial degree 3'
    coeff: [-0.0587 3.8915 -86.0531 635.8273]

>> fit.type

ans =
polynomial degree 3

>> fit.coeff
ans =
   -0.0587    3.8915  -86.0531  635.8273
```

```
>> normresid
normresid =
    0.0555
```

Les valeurs des résidus sont :

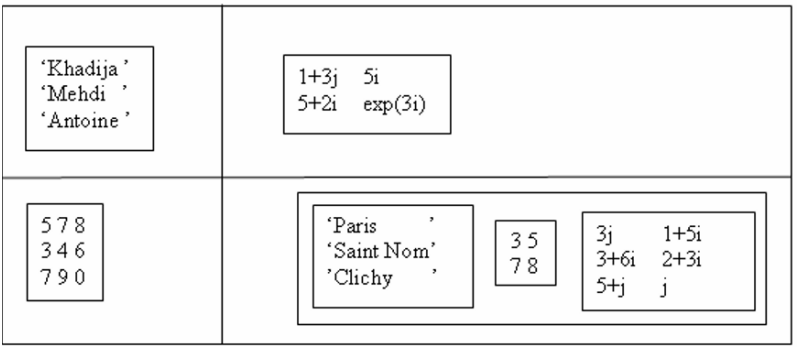
```
>> resids

resids =
    0.0026
   -0.0220
    0.0055
    0.0338
    0.0232
   -0.0251
   -0.0095
   -0.0068
   -0.0084
    0.0066
```

II. Tableaux multidimensionnels de cellules

II.1. Cellules, Tableaux de cellules

Les cellules sont des tableaux pouvant contenir des éléments de différents types de données : des chaînes de caractères, des nombres, des tableaux, etc.

La cellule suivante est un tableau 2x2 contenant des tableaux de chaînes de caractères, des nombres complexes, des tableaux d'entiers et une autre cellule.



II.1.1. Construction de cellules

Les éléments d'une cellule doivent être mis entre accolades « { } »

La cellule ci-dessus doit être écrite de façon suivante :

```
>>  Cell2={['Khadija';'Mehdi    ';'Antoine']  [1+3j  5i;5+2i
exp(3i)]; ...
[5 7 8 ;3 4 6 ;7 9 0] Cell1}
```

Avec `Cell1 précédemment définie par`:

```
>> Cell1={['Paris    ';'Saint Nom';'Clichy   '] [3 5;7 8] ...
[3j 1+5i;3+6i 2+3i ;5+j j]}
```

```
Cell1 =
    [3x9 char]    [2x2 double]    [3x2 double]
```

```
Cell2 =
    [3x7 char  ]    [2x2 double]
    [3x3 double]    {1x3 cell  }
```

Nous remarquons que `Cell1` est constituée d'un tableau d'une ligne et 3 colonnes.

- Le premier élément est du type chaîne de caractères (3 lignes de 9 colonnes ou caractères),
- Le deuxième est de type réel (matrice carrée 2x2),

- Le troisième est un tableau de type double de 3 lignes et 2 colonnes.

Les chaînes d'une même colonne doivent avoir la même longueur (même nombre de caractères). Dans le cas précédent, nous avons du rajouter autant de blancs aux chaînes les plus courtes afin d'avoir la même longueur que la plus grande.
Pour s'affranchir de ce calcul fastidieux de longueur des chaînes, nous utilisons la commande `strvcat` (concaténation verticale des chaînes de caractères).

```
>> Cell1={[strvcat('Paris','Saint Nom','Clichy')] [3 5;7 8]
...
[3j 1+5i;3+6i 2+3i ;5+j j]}
>>    Cell2={[strvcat('Khadija','Mehdi','Antoine')]    [1+3j
5i;5+2i exp(3i)]; ...
[5 7 8 ;3 4 6 ;7 9 0] Cell1}

Cell1 =
    [3x9 char]    [2x2 double]    [3x2 double]

Cell2 =
    [3x7 char  ]    [2x2 double]
    [3x3 double]    {1x3 cell  }
```

Un autre moyen de construire des cellules est d'utiliser la commande `cell`.

```
>> x=cell(2,3)
x =
     []      []      []
     []      []      []
```

Cette commande crée une cellule vide taille 2x3.

On peut spécifier après chaque élément de la cellule.

```
>> x{1,1}=1
x =
    [1]      []      []
     []      []      []
```

```
>> x{2,3}='MATLAB R2009a'
x =
    [1]      []                  []
     []      []    'MATLAB R2009a'
```

La commande `cell` permet de définir le type d'objet cellule avant de spécifier leurs valeurs.

```
>> A = cell(3,1);
>> for n = 1:3
   A{n} = randn(n);
   A{n}
end
```

```
ans =
   -2.2023
```

```
ans =
    0.9863    0.3274
   -0.5186    0.2341


ans =
    0.0215   -0.3744    1.4725
   -1.0039   -1.1859    0.0557
   -0.9471   -1.0559   -1.2173
```

Le tableau A possède 3 cellules.

```
>> A
A =
    [   -2.2023]
    [2x2 double]
    [3x3 double]
```

Chaque cellule est aussi un tableau de taille `n`. Pour accéder à la valeur d'indices (2,3) de la 3ème cellule, on fait l'indexation suivante :

```
>> A{3}(2,3)
ans =
    0.0557
```

II.1.2. Accès aux éléments des cellules, indexation

Pour accéder à un élément d'une cellule, on l'indexe en utilisant les accolades :

```
>> Cell1{1}
ans =
Paris
Saint Nom
Clichy
```

```
>> Cell1{1:2}
ans =
Paris
Saint Nom
Clichy

ans =
     3     5
     7     8
```

Pour avoir tous les éléments de la cellule, on met l'opérateur « : » entre les accolades, comme pour les tableaux de réels.

```
>> Cell1{:}
ans =
Paris
Saint Nom
Clichy
ans =
     3     5
     7     8

ans =
        0 + 3.0000i   1.0000 + 5.0000i
   3.0000 + 6.0000i   2.0000 + 3.0000i
   5.0000 + 1.0000i        0 + 1.0000i
```

Nous remarquons que l'élément 2x2 de la cellule `Cell2` est aussi une autre cellule de taille 1x3 (1 ligne et 3 colonnes).

```
>> x=Cell2{2,2}

x =
    [3x9 char]    [2x2 double]    [3x2 double]
```

```
>> x=Cell2{2,2}

x =
    [3x9 char]    [2x2 double]    [3x2 double]

>> x(3)
ans =
    [3x2 double]

>> iscell(x)
ans =
     1
```

L'indexation peut se faire soit entre crochets soit entre accolades.

Comme nous venons de le voir, `Cell2{2,2}` retourne le contenu de la cellule `Cell2` spécifié par l'index (2ème ligne et 2ème colonne).

On utilise les parenthèses pour indexer les cellules d'un tableau (voir tableaux de cellules).

II.1.3. Concaténation de cellules

Nous créons 2 cellules, `Cell3` et `Cell4` que nous allons concaténer en une cellule `Cell5`.

```
>>  Cell3={[1  3;5  9]  ['2  Sources  chaudes';'Saint  Nom
';'Saint Lazare     ']}
Cell3 =
    [2x2 double]    [3x17 char]

>> Cell3{2}
ans =
2 Sources chaudes
Saint Nom
Saint Lazare

>> Cell4={'Thermodynamique à échelle finie' 8667}
Cell4 =
    [1x30 char]    [8667]
```

• *Concaténation des cellules*

```
>> Cell5={Cell3 Cell4}
Cell5 =
    {1x2 cell}    {1x2 cell}
```

On remarque bien que `Cell5` est constituée de 2 cellules.

Pour accéder au 1er élément de la 1ère cellule de `Cell5`, nous devons faire deux fois l'indexation.

```
>> Cell5{1}
ans =
    [2x2 double]    [3x17 char]
```

Le premier élément de `Cell5` est une autre cellule à 2 éléments.

```
>> Cell5{1}{1}
ans =
     1     3
     5     9
```

On peut aussi concaténer les éléments de ces deux cellules pour en faire les éléments d'une seule. Pour ça, on utilise les crochets « [] » au lieu des accolades « {} ».

```
>> Cell6=[Cell3 Cell4]
Cell6 =
    [2x2 double]    [3x17 char]    [1x31 char]    [8667
```

On obtient bien une seule cellule avec un élément supplémentaire que la cellule Cell5.

```
>> Cell6{:}
ans =
     1     3
     5     9

ans =
2 Sources chaudes
Saint Nom
Saint Lazare

ans =
Thermodynamique à échelle finie
ans =
        8667
```

- *2ème élément de la 1ère cellule*

```
>> Cell5{1}{2}
ans =
2 Sources chaudes
Saint Nom
Saint Lazare
```

II.2. Tableaux de cellules

II.2.1. Tableaux bidimensionnels

Un tableau de cellules contient des cellules de mêmes dimensions.

```
>> x1 = {['MATLAB R2009a' ; 'exergie      '] ,8667}
x1 =
    [2x13 char]    [8667]
```

```
>> x2= {randn(2),'SIMULINK 7.8.5'}
x2 =
    [2x2 double]    'SIMULINK 7.8.5'
```

```
>> x12 =
    [2x13 char  ]    [          8667]
    [2x2  double]    'SIMULINK 7.8.5'
```

L'indexation entre parenthèses retourne les cellules d'un tableau spécifiées par l'index.

```
>> x12(1)
ans =
    [2x13 char]
```

```
>> x12(1:2)
ans =
    [2x13 char]    [2x2 double]
```

L'indexation entre accolades retourne le contenu des cellules d'un tableau spécifiées par l'index.

```
>> x12{1}
ans =
MATLAB R2009a
exergie
```

```
>> x12{1:2}
ans =
MATLAB R2009a
exergie

ans =
    0.3188   -0.4336
   -1.3077    0.3426
```

Considérons le tableau de cellules suivant dont nous rentrons les valeurs par lignes.

```
>> x(1,:) = {'tissu rouge', {'transparent', [3193, 8667]}};
>> x(2,:) = {'2 étoiles vertes', {'vertes',  [3, 2]}};
>> x(3,:) = {'brushing film du Dimanche',  {'Octobre', [13
10]}};
>> x
x =
    'tissu rouge'          {1x2 cell}
    '2 étoiles vertes'     {1x2 cell}
           [1x25 char]     {1x2 cell}
```

Ce tableau est constitué de 3 cellules contenant chacune une cellule ayant 2 éléments (une chaîne de caractères et un vecteur ligne de taille 2).

Le deuxième élément de chacune de ces cellules internes vaut :

```
>> x{:,2}

ans =
    'transparent'    [1x2 double]
ans =
    'vertes'    [1x2 double]
ans =
    'Octobre'    [1x2 double]
```

Le 4ème élément du tableau `x`*,* `x{4}`**,** *étant une cellule :*

```
>> x{4}

ans =
    'transparent'    [1x2 double]
```

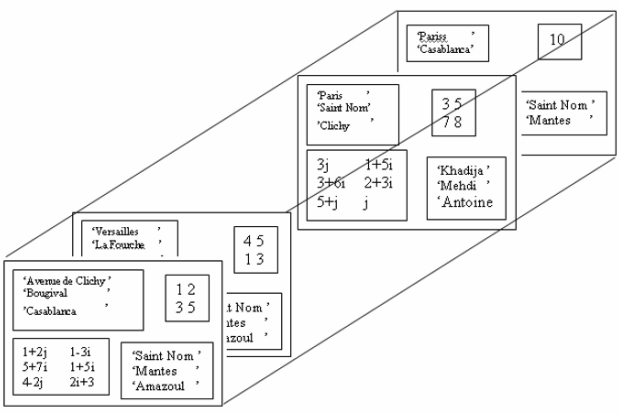
Le premier élément de cette cellule peut être indexé comme suit pour pouvoir extraire sa valeur :

```
>> x{4}{1}
ans =
transparent
```

C'est la 1ère valeur du 4ème élément du tableau de cellules x.

II.2.2. Tableaux multidimensionnels de cellules

La même notion de pages est définie pour les tableaux multidimensionnels de cellules où la même commande `cat` peut être utilisée selon une dimension.



Considérons les tableaux de cellules suivants, basés sur la même structure que le schéma précédent :

```
>> % Cellule A

% Cellule A
>> A{1,1} = 'Saint Nom';
>> A{1,2} = ones(2);
>> A{2,1} = i;
>> A{2,2} = 'Casablanca';

 % Cellule B
>> B{1,1} = 'Antony';
>> B{1,2} = [1 3;5 7];
>> B{2,1} = exp(j*[pi pi/4]);
>> B{2,2} = 3;

>> % Tableau à 3 dimensions de cellules
>> C = cat(3, A, B);
```

Nous obtenons le tableau de structures à 3 dimensions suivant :

```
>> C
C(:,:,1) =
    'Saint Nom'           [2x2 double]
    [0.6324 + 1.0000i]    'Casablanca'

C(:,:,2) =
    'Antony'         [2x2 double]
    [1x2 double]     [          3]
```

Chacune de ces 2 pages est un tableau bidimensionnel de cellules, qui peut être remplie directement sans utiliser la commande `cat` comme pour les tableaux ordinaires.

```
>> D(:,:,1)={'Saint Nom' ones(2);i+rand 'Casablanca'};
>> D(:,:,2)={'Antony' [1 3;5 7]; exp(j*[pi pi/4]) 3};

>> D

D(:,:,1) =
    'Saint Nom'           [2x2 double]
    [0.7382+ 1.0000i]     'Casablanca'
D(:,:,2) =
    'Antony'         [2x2 double]
    [1x2 double]     [          3]
```

On obtient le même tableau de cellules à 3 dimensions sans utiliser la commande `cat`.

II.3. Fonctions propres aux cellules et tableaux de cellules

- *cellfun*

Applique des fonctions MATLAB à des tableaux de cellules.

```
>> CellNombres = {1:5, [1; 3; 2], [] inf};
>> CellNombres{:}
ans =
     1     2     3     4     5

ans =
     1
     3
     2

ans =
     []
ans =
   Inf
```

Le calcul de la moyenne et de la variance des 4 cellules du tableau se fait par application des commandes `mean` et `cov`.

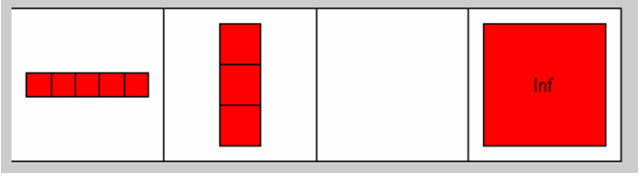
```
>> MoyCellNombres = cellfun(@mean, CellNombres)
MoyCellNombres =
     3     2   NaN   Inf
```

```
>>     VarCellNombres     =     cellfun(@std,     CellNombres,
'UniformOutput', false)

>>     VarCellNombres     =     cellfun(@std,     CellNombres,
'UniformOutput', false)
VarCellNombres =
    [1.5811]    [1]    [NaN]    [NaN]
```

Si on veut récupérer la taille du tableau de cellules, nous utilisons la commande `size`.

```
>> [NbLignes,NbCols] = cellfun(@size, CellNombres)

NbLignes =
     1     3     0     1
NbCols =
     5     1     0     1
```

```
>> cellplot(CellNombres)
```



La cellule vide, représentée par un carré blanc, possède 0 ligne et 0 colonne.

- *iscell*

Pour déterminer si une donnée est du type cellule, on utilise la commande `iscell` qui retourne 1 si l'argument est une cellule et 0 dans le cas contraire.

```
>> Cell5{1}
ans =
    [2x2 double]    [3x17 char]
```

```
>> iscell(ans)
ans =
     1
```

Le premier élément de la cellule `Cell5` est aussi une cellule.

- *num2cell*

Transforme un tableau numérique en tableau de cellules.

```
>> A=[1 3;5 7];
>> B=num2cell(A)
B =
    [1]    [3]
    [5]    [7]
```

```
>> size(B)
ans =
     2     2
```

```
>> iscell(B)
ans =
     1
```

Le tableau B, à 2 lignes et 2 colonnes est du type cellule.

`C = num2cell(A,dims)` convertit la matrice `A` en tableau de cellules `C` en créant des cellules séparées selon la dimension spécifiée dans `dims`.

```
>> C=num2cell(A,1) % contenu des colonnes dans cellules
                   % séparées

C =
    [2x1 double]    [2x1 double]
```

```
>> C(1)

ans =
    [2x1 double]
```

Le contenu de la 1ère cellule est :

```
>> C{1}
ans =
     1
     5

>> D=num2cell(A,2) %contenu des lignes dans cellules séparées
D =
    [1x2 double]
    [1x2 double]
```

Cette commande s'applique aussi pour des tableaux multidimensionnels.

- *iscellstr*

Retourne 1 pour un tableau de cellules sous forme de chaînes de caractères et 0 dans le cas contraire.

```
>> CellChaine={'MATLAB R2009a','Amel','SIMULINK', 'Khadija'}

CellChaine =
    'MATLAB R2009a'    'Amel'    'SIMULINK'    'Khadija'

>> iscellstr(CellChaine)

ans =
     1
```

- *cellstr*

Cette commande crée une structure de chaînes de caractères à partir d'un tableau de chaînes.

```
>>  TabChaines=['MATLAB  R2009a';'Amel              ';'SIMULINK
'; …
'Khadija      ']

TabChaines =
MATLAB R2009a
Amel
SIMULINK
Khadija
```

```
>> CellChaines=cellstr(TabChaines)

CellChaines =

    'MATLAB R2009a'
    'Amel'
    'SIMULINK'
    'Khadija'
```

```
>> whos
 Name             Size             Bytes  Class    Attributes

  CellChaines      4x1                304  cell
  TabChaines       4x13               104  char
  ans              4x1                304  cell
```

Les mêmes chaînes de caractères occupent 304 octets en tant que cellule contre 104 en tant que tableau de chaînes de caractères.

- *celldisp*

Affiche la structure donnée en argument.

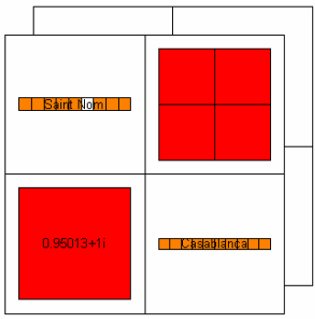
```
>> celldisp(CellChaine)

CellChaine{1} =
MATLAB R2009a

CellChaine{2} =
Amel

CellChaine{3} =
SIMULINK

CellChaine{4} =
Khadija
```

- *cellplot*

Affiche graphiquement un tableau de cellules.

.

```
>> A(:,:,1)={'Saint Nom' ones(2);i+rand 'Casablanca'}
>> A(:,:,2)={'Antony' [1 3;5 7]; exp(j*[pi pi/4]) 3};
>> cellplot(A)
```



- *cell2mat*

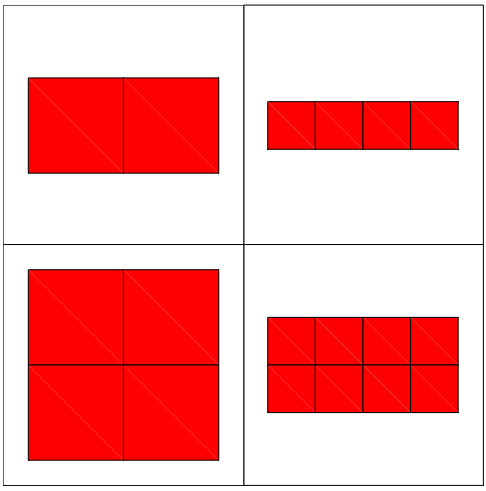
Convertit un tableau multidimensionnel de cellules en une seule matrice (tableau 2D).

```
>> C = {[1 5] [2 3 4 7]; [5 8; 9 7] [4 8 9 3; 1 12 20 5]};

>> C{:}
```

```
ans =
     1     5
ans =
     5     8
     9     7
ans =
     2     3     4     7
ans =
     4     8     9     3
     1    12    20     5

>> cell2mat(C)
ans =
     1     5     2     3     4     7
     5     8     4     8     9     3
     9     7     1    12    20     5
```

La seule condition est que les contenus de la cellule doivent pouvoir être concaténés dans un hyperrectangle.

Si on trace la figure donnant la forme du tableau de cellules, nous remarquons que chaque cellule possède toujours le même nombre de lignes que celle qui lui est voisine horizontalement et le même nombre de colonnes que celle qui lui est voisine verticalement.



- *sort*
  Arrange les chaînes d'un tableau de cellules par ordre alphabétique.

```
>> TabChaines={'MATLAB R2009a','Amel','SIMULINK', 'Khadija'}
TabChaines =
    'MATLAB R2009a'    'Amel'    'SIMULINK'    'Khadija'

>> CellOrd=sort(TabChaines)
CellOrd =
    'Amel'    'Khadija'    'MATLAB R2009a'    'SIMULINK'
>> iscell(CellOrd)
ans =
     1
```

Le résultat de la commande `sort` est aussi un tableau de cellules de chaînes de caractères ordonnées par ordre alphabétique.

III. Tableaux multidimensionnels de structures

III.1. Structures

Une structure avec des champs vides est obtenue par la commande `struct` sans remplir les tableaux de cellules contenant les valeurs des éléments des différents champs.

```
>> StructVide = struct('Champ1',{},'Champ2',{})
StructVide =
0x0 struct array with fields:
    Champ1
    Champ2
```

Considérons la description de personnes par certaines caractéristiques qui sont :

- la taille,
- la couleur des yeux,
- la date de naissance,
- l'adresse,
- lieu de naissance.

Une personne dénommée `NomPers` sera décrite par une structure avec les 5 champs précédents.

```
>> NomPers = struct('Poids',{50},...
                    'CouleurDesYeux',{'noirs'},...
                    'DateDeNaissance', {'10 Mai 1977'},...
                    'Adresse',  {'Allée  Les  Erables  Saint
Nom'},...
                    'LieuDeNaissance',{'Sidi        Bernoussi
Casablanca'})
NomPers =
             Poids: 50
```

```
    CouleurDesYeux: 'noirs'
     DateDeNaissance: '10 Mai 1977'
            Adresse: 'Allée Les Erables Saint Nom'
 LieuDeNaissance: 'Sidi Bernoussi Casablanca'
```

La description de 2 personnes à la fois se fait en spécifiant 2 valeurs à chaque fois pour chaque champ de la structure `NomPers`.

```
>> NomPers = struct('Poids',{50 75},...
   'CouleurDesYeux',{'noirs' 'marrons'},...
   'DateDeNaissance', {'10 Mai 1977' '13 Octobre 1954'},...
   'Adresse', {'Allée Les Erables, Saint Nom' 'Avenue de Clichy
Paris'},...
   'LieuDeNaissance',{'Sidi Bernoussi Casablanca' 'Amazoul'})
```

```
>> NomPers(1)
ans =
            Poids: 50
    CouleurDesYeux: 'noirs'
 DateDeNaissance: '10 Mai 1977'
            Adresse: 'Allée Les Erables, Saint Nom'
LieuDeNaissance: 'Sidi Bernoussi Casablanca'

>> NomPers(2)
ans =
            Poids: 75
    CouleurDesYeux: 'marrons'
    DateDeNaissance: '13 Octobre 1954'
            Adresse: 'Avenue de Clichy Paris'
    LieuDeNaissance: 'Amazoul'
```

On peut accéder à n'importe quel champ d'une personne définie dans cette structure.

```
>> NomPers.Adresse
ans =
Allée Les Erables, Saint Nom

ans =
Avenue de Clichy Paris
```

Pour obtenir l'adresse de la première personne :

```
>> Adress1=NomPers(1).Adresse

Adress1 =
Allée Les Erables, Saint Nom
```

Et la date de naissance de la deuxième personne :

```
>> NomPers(2).DateDeNaissance
ans =
13 Octobre 1954
```

On peut étendre la structure en affectant directement d'autres valeurs aux champs de la structure `NomPers`.

L'extension de la structure peut se faire en affectant uniquement un champ, comme le champ Poids.

```
>> NomPers(3).Poids=78
```

```
>> NomPers
NomPers =
1x3 struct array with fields:
    Poids
    CouleurDesYeux
    DateDeNaissance
    Adresse
    LieuDeNaissance
```

Les autres champs non affectés, retournent des valeurs vides.

```
>> NomPers(3).Adresse
ans =
     []
```

On le voit directement en affichant les champs de `NomPers(3)`.

```
>> NomPers(3)
ans =
            Poids: 78
      CouleurDesYeux: []
     DateDeNaissance: []
            Adresse:  []
     LieuDeNaissance: []
```

Tous les champs sont vides sauf le champ `Poids`, seul auquel on a affecté la valeur 78.
L'affectation directe d'une valeur à un champ est la deuxième façon de créer une structure. La simple instruction suivante permet de créer la structure `a` en affectant la valeur 5 au champ `b`.

```
>> a.b=5 ;
>> isstruct(a)
ans =
     1
```

La variable `a` désigne bien une structure qui possède jusqu'à présent un seul champ `b` qui vaut 5.

```
>> a(:)
ans =
    b: 5
```

III.2. Tableaux de structures

Un tableau multidimensionnel peut posséder comme éléments des structures.

Considérons la structure `NomPers` étudiée précédemment.

```
>> NomPers = struct('Poids',{50},...
                    'CouleurDesYeux',{'noirs'},...
                    'DateDeNaissance', {'10 Mai 1977'},...
                    'Adresse',  {'Allée  Les  Erables  Saint,
Nom'},...
                    'LieuDeNaissance',{'Sidi       Bernoussi,
Casablanca'})
NomPers =

              Poids: 50
     CouleurDesYeux: 'noirs'
    DateDeNaissance: '10 Mai 1977'
            Adresse: 'Allée Les Erables Saint Nom'
    LieuDeNaissance: 'Sidi Bernoussi Casablanca'
```

La commande suivante spécifie cette structure comme étant l'élément (2,3) de la 5ème page du tableau `TablStruct` à 3 dimensions.

```
>> TablStruct(2,3,5)=NomPers
TablStruct =
2x3x5 struct array with fields:
    Poids
    CouleurDesYeux
    DateDeNaissance
    Adresse
    LieuDeNaissance
```

```
>> isstruct(TablStruct)
ans =
     1
```

Ce tableau est lui-même de type structure.

```
>> size(TablStruct)
ans =
     2     3     5
```

Si on veut afficher les éléments de ce tableau, on obtient des structures avec des champs vides, sauf l'élément (2, 3, 5).

```
>> TablStruct(1,2,4)
ans =
              Poids: []
     CouleurDesYeux: []
    DateDeNaissance: []
            Adresse: []
    LieuDeNaissance: []
```

```
>> TablStruct(2,3,5)
ans =
              Poids: 50
     CouleurDesYeux: 'noirs'
    DateDeNaissance: '10 Mai 1977'
            Adresse: 'Allée Les Erables, Saint Nom'
    LieuDeNaissance: 'Sidi Bernoussi, Casablanca'
```

III.3. Convertir un tableau de cellules en tableau de structures et inversement

Les fonctions `struct2cell` et `cell2struct` convertissent, respectivement, un tableau de structures en tableau de cellules et inversement.

```
>> TabCell=struct2cell(TablStruct);
```

Le nouveau tableau `TabCell` est bien du type cellules.

```
>> isstruct(TabCell)
ans =
     0
```

```
>> iscell(TabCell)
ans =
     1
```

Cette commande convertit un tableau (`m,n`) de structures à `p` champs en un tableau de cellules de dimensions (`p,m,n`).

```
>> size(TabCell)
ans =
     5     2     3     5
```

Seul le dernier élément de `TabCell` possède une cellule non vide.

```
>> TabCell(:,:,3,5)
ans =
     []    [         50]
     []    'noirs'
     []    '10 Mai 1977'
     []      [1x27 char]
     []      [1x25 char]
```

La conversion d'un tableau multidimensionnel à un tableau de cellules se fait par la commande `cell2struct` qu'on applique au tableau `Tabcell`.

```
>> Tabtruct=cell2struct(TabCell,{'Poids','CouleurDesYeux', …
'DateDeNaissance', 'Adresse','LieuDeNaissance'},1)

Tabtruct =
    2x3x5 struct array with fields:
    Poids
    CouleurDesYeux
    DateDeNaissance
    Adresse
    LieuDeNaissance
```

III.4. Fonctions propres aux tableaux de structures

- *isstruct*

Retourne 1 si l'argument est une structure et 0 autrement.

```
>> isstruct(NomPers)
ans =
     1
```

- *rmfield*

```
>> NomPers2=rmfield(NomPers,'Poids')

NomPers2 =
     CouleurDesYeux: 'noirs'
    DateDeNaissance: '10 Mai 1977'
            Adresse: 'Allée Les Erables Saint, Nom'
    LieuDeNaissance: 'Sidi Bernoussi, Casablanca'
```

On obtient une structure sans les champs supprimés par la commande `rmfield`.

- *fieldnames*

Cette commande retourne les champs de la structure donnée en argument.

```
>> Champs = fieldnames(NomPers)
Champs =
    'Poids'
    'CouleurDesYeux'
    'DateDeNaissance'
    'Adresse'
    'LieuDeNaissance'
```

- *getfield*

Retourne la valeur d'un champ d'une structure.
Si on veut avoir le lieu de naissance dans la structure `NomPers` :

```
>> ValNaissance=getfield(NomPers,'LieuDeNaissance')
ValNaissance =
Sidi Bernoussi, Casablanca
```

Le résultat est le même que celui donné par la commande suivante en pointant le champ correspondant :

```
>> NomPers.CouleurDesYeux
ans =
noirs
```

- *isfield*

Retourne 1 si la chaîne spécifiée est un champ de la structure et 0 autrement.

```
>> isfield(NomPers,'CouleurDesYeux')
ans =
     1
```

On peut entrer comme argument plusieurs chaînes de caractères sous forme d'une cellule.

```
>> isfield(NomPers,{'Taille','DateDeNaissance'})
ans =
     0     1
```

La chaîne '`Taille`' n'est pas un champ de la structure `NomPers`.

- *orderfields*

Les champs sont ordonnés par ordre alphabétique.

```
>> OrdNomPers=orderfields(NomPers)
OrdNomPers =
            Adresse: 'Allée Les Erables Saint, Nom'
      CouleurDesYeux: 'noirs'
     DateDeNaissance: '10 Mai 1977'
    LieuDeNaissance: 'Sidi Bernoussi, Casablanca'
              Poids: 50
```

On obtient ainsi une nouvelle structure dont la liste des champs est ordonnée.

```
>> isstruct(OrdNomPers)
ans =
     1
```

- *setfield*

Permet de spécifier la valeur d'un champ d'une structure.

Ci-après, nous modifions la valeur du champ '`Adress'` de la structure `NomPers` (c'est le cas où la personne déménage de Saint Nom à l'avenue de Clichy, Paris 17ème).

```
>> NomPers2=setfield(NomPers,'Adresse','90 Avenue de Clichy,
75017 Paris')
NomPers2 =
              Poids: 50
     CouleurDesYeux: 'noirs'
    DateDeNaissance: '10 Mai 1977'
            Adresse: '90 Avenue de Clichy, 75017 Paris'
    LieuDeNaissance: 'Sidi Bernoussi, Casablanca'
```

La même chose est obtenue par :

```
>> NomPers.Adresse='90 Avenue de Clichy, 75017 Paris'
```

- *struct2array*

Transforme une structure en vecteur.

```
>> VectNomPers=struct2array(NomPers)
VectNomPers =
2noirs10 Mai 197790 Avenue de Clichy, 75017 ParisSidi
Bernoussi, Casablanca
```

```
>> size(VectNomPers)
ans =
     1    75
```

- *structfun*

Applique une fonction MATLAB à chaque champ d'une structure.

La mise en majuscules des champs de la structure NomPers se fait par la ligne de commandes suivante :

```
>> MajChamps = structfun(@(x) ( upper(x) ), NomPers,
'UniformOutput', false)
MajChamps =
              Poids: 50
     CouleurDesYeux: 'NOIRS'
    DateDeNaissance: '10 MAI 1977'
            Adresse: 'ALLÉE LES ERABLES SAINT, NOM'
    LieuDeNaissance: 'SIDI BERNOUSSI, CASABLANCA'
```

Ci-après, on transforme le contenu des champs par le code ASCII de leurs valeurs

```
>>   CodeAscii=   structfun(@(x)   (   abs(x)   ),   NomPers,
'UniformOutput', false)

CodeAscii =
             Poids: 50
     CouleurDesYeux: [110 111 105 114 115]
    DateDeNaissance: [49 48 32 77 97 105 32 49 57 55 55]
            Adresse: [1x28 double]
    LieuDeNaissance: [1x26 double]
```