Introduction à Matlab

Table des matières

1 Introduction

1.1 Remerciements

A Mathieu Bruchet pour sa relecture et ses conseils ainsi qu'à Marc Lussac pour ses encouragements. Merci aussi à Charles-Albert Lehalle qui m'a beaucoup aidé dans mon apprentissage de Matlab.

1.2 Présentation succinte

MATLAB est une abréviation de Matrix LABoratory. Écrit à l'origine, en Fortran, par C. Moler, MATLAB était destiné à faciliter l'accès au logiciel matriciel développé dans les projets LINPACK et EISPACK. La version actuelle, écrite en C par the MathWorks Inc., existe en version professionnelle et en version étudiant. Sa disponibilité est assurée sur plusieurs plates-formes : Sun, Bull, HP, IBM, compatibles PC (DOS, Unix ou Windows), Macintoch, iMac et plusieurs machines parallèles.

Nous nous intéresserons ici à Matlab comme langage de programmation, tout en essayant de tirer parti de sa spécificité : la vectorisation. En effet, tout comme d'autres langages vectoriels (ou matriciels) tels que R, Scilab, Octave, Gauss, Matlab est très performant dans la manipulation de vecteurs ou de matrices : il est alors dans l'intérêt du programmeur d'éviter un maximum les boucles habituelles dans d'autres langages.

J'ai réalisé ce document pour `http://www.developpez.com`. Je suis à la base un informaticien et non pas un mathématicien, je présente donc Matlab en tant que langage de programmation. J'ai volontairement été succint sur les bases du langage. Tout simplement parce qu'il existe déjà des tonnes de documents d'universitaires sur le sujet, en plus de l'excellente documentation fournie avec le logiciel ; pour arriver plus rapidement à la partie intéressante : la vectorisation et la programmation objet avec Matlab.

Je tiens à disposition les sources LaTeXde ce document pour toute amélioration ou modification, considérez qu'il est sous GFDL (license de documentation libre GNU). Je me tiens aussi à votre disposition pour tout changement license, si vous rencontrez un problème de compatibilité entre la GNU GPL et la GFDL.

Thomas Grin, mai 2006 Contact : `http://www.cerbermail.com/?62iih6MJAW`.

2 Prise en main

2.1 Vecteurs

Pour créer un vecteur, la syntaxe est la suivante :

```
>> v = [1;2;3;4]

v =

     1
     2
     3
     4
```

En séparant les chiffres par des points-virgules, vous obtenez un vecteur colonne, en utilisant des espaces ou des virgules vous aurez un vecteur ligne. Notez que par défaut, chaque élement d'un vecteur est un double.

```
>> v = [1 2 3 4]

v =

     1     2     3     4
```

L'opérateur de transposition, comme en math, est l'apostrophe :

```
>> v'

ans =

     1
     2
     3
     4
```

Petite subtilité : l'opérateur $<\acute{>}$ renvoie le vecteur transconjugué alors que $<.\acute{>}$ renvoie la vraie transposée. Toutefois, si vous restez dans les réels, vous ne verrez pas la différence.

```
>> v=[1,1+i,2]
```

```
v =

   1.0000             1.0000 + 1.0000i   2.0000

>> v'

ans =

   1.0000
   1.0000 - 1.0000i
   2.0000

>> v.'

ans =

   1.0000
   1.0000 + 1.0000i
   2.0000
```

Evidement, nous disposons des 4 opérations classiques :

$$+ - */$$

Vous noterez qu'il existe 2 multiplications possibles entre 2 vecteurs : la multiplication de vecteur et la multiplication membre à membre.

```
>> a = [1:5] , b = [1:5]

a =

     1     2     3     4     5


b =

     1     2     3     4     5

>> a*b
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> a'*b
```

```
ans =

     1     2     3     4     5
     2     4     6     8    10
     3     6     9    12    15
     4     8    12    16    20
     5    10    15    20    25

>> a.*b

ans =

     1     4     9    16    25
```

En écrivant `[1:5]` on crée une suite partant de 1 jusqu'à 5, par pas de 1. De même, `[1:2:10]` va créer un vecteur de 1 à 10 par pas de 2, donc de 5 éléments.

```
>> length(1:2:10)

ans =

     5
```

2.2 Matrices

Les matrices suivent exactement la même syntaxe que les vecteurs : les lignes sont séparés par des espaces ou des virgules et les colonnes par des points-virgules.

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

m =

     1     2     3
     4     5     6
     7     8     9
```

A présent, une fonction extrêmement utile, pour le programmeur comme pour le mathématicien : la fonction `find` permet de renvoyer les indices des éléments vérifiant la condition placée en paramètre.

```
>> m

m =

     1     2     3
     4     5     6
     7     8     9

>> find(m>5)

ans =

     3
     6
     8
     9

>> m(find(m>5))

ans =

     7
     8
     6
     9

>> m(find(m>5 & m<7))

ans =

     6
```

2.3 Opérateurs principaux sur les matrices

```
ones(i,j)       crée une matrice de i lignes j colonnes contenant des 1
zeros(i,j)      crée une matrice de i lignes j colonnes contenant des 0
eye(i,j)        crée une matrice de i lignes j colonnes avec
                     des 1 sur la diagonale principale et 0 ailleurs
diag(U)         extrait la diagonale de la matrice U
triu(A)         renvoie la partie supéerieure de A
tril(A)         renvoie la partie inférieure de A
```

```
linspace(a,b,n) crée un vecteur de n composantes uniformément réparties de a à b
A\b             résolution du système linéaire Ax=b
det(A)          déterminant d'une matrice
rank(A)         rang d'une matrice
inv(A)          inverse d'une matrice
pinv(A)         pseudo inverse d'une matrice
svd(A)          valeurs singulières d'une matrice
norm(A)         norme matricielle ou vectorielle
```

2.4 Les Cell Array

Un Cell Array est en Matlab, une matrice pouvant contenir n'importe quelle type de données sans aucune contrainte : des structures, des vecteurs, des chaines de caractères... Cela permet de manipuler des objets condensés de manière puissante.

```
BondData = {'15-Jul-1999'  0.06000   99.93
            '15-Jan-2000'  0.06125   99.72
            '15-Jul-2000'  0.06375   99.70
            '15-Jan-2001'  0.06500   99.40
            '15-Jul-2001'  0.06875   99.73
            '15-Jan-2002'  0.07000   99.42
};
```

Cet exemple, tiré de la documentation de la Financial Toolbox crée un Cell Array contenant une colonne de chaînes de caractères et le reste de flottants. A bien garder en tête : accolade = cell array, crochet = matrice.

On manipule ensuite le Cell Array comme n'importe quelle matrice :

```
>> BondData(:,1)

ans =

    '15-Jul-1999'
    '15-Jan-2000'
    '15-Jul-2000'
    '15-Jan-2001'
    '15-Jul-2001'
    '15-Jan-2002'
```

On accède avec les parenthèses aux cellules du Cell Array, quoi qu'elles contiennent : ici les chaînes de caractères sont renvoyées avec des quotes, à cause de leur type.

```
>> BondData(:,2)

ans =

    [0.0600]
    [0.0612]
    [0.0638]
    [0.0650]
    [0.0688]
    [0.0700]
```

Ici, la deuxième colonne, que l'on avait saisie comme des flottant est renvoyée avec des crochets : en effet, un nombre est un vecteur de nombre à un élément.

Si on souhaite accéder non pas aux cellules, mais à leur contenu directement, voici la syntaxe :

```
>> BondData{1:4,2}

ans =

    0.0600


ans =

    0.0612


ans =

    0.0638


ans =

    0.0650
```

Ici, j'ai demandé le contenu des cellules des 4 premières lignes de la deuxième colonne.

2.5 Les structures

Les structures permettent d'organiser vos données, je ne saurais trop vous conseiller de vous définir un format de structure standard à toutes vos fonctions : par exemple un format pour les signaux avec un vecteur de date, une matrice de valeur et des champs d'informations.

On peut simplement créer une structure par affectation, comme ceci :

```
>> s.chiffre = 1; s.chaine='toto'; s.cellarray={1,'toto'} ; s.vecteur=[1;2;3;4];s

s =

      chiffre: 1
       chaine: 'toto'
    cellarray: {[1]  'toto'}
      vecteur: [4x1 double]
```

Notez à présent, que chaque champs de notre structure est nommé, ce qui explicite la syntaxe suivante :

```
>> s = struct('chiffre',1,'chaine','toto','cellarray',{{1,'toto'}},'vecteur',[1;2;3;4])

s =

      chiffre: 1
       chaine: 'toto'
    cellarray: {[1]  'toto'}
      vecteur: [4x1 double]
```

Il y a deux choses à remarquer ici : tout d'abord, les noms de champs sont entrés en paramètres, dans l'ordre en tant que chaînes de caractères, c'est une syntaxe très courante en Matlab, équivalente dans d'autres langages à ce genre de chose : `function(chiffre=1,chaine='toto',...)`.

D'autre part, j'ai dû doubler les accolades pour <cellarray>, sinon Matlab m'aurait créé un <s> qui aurait été un Cell Array de la structure définie, avec 2 cellules. Pas évidement, essayons...

```
>> s = struct('chiffre',1,'chaine','toto','cellarray',{1,'toto'},'vecteur',[1;2;3;4])

s =

1x2 struct array with fields:
```

```
    chiffre
    chaine
    cellarray
    vecteur

>> s(1),s(2)

ans =

      chiffre: 1
       chaine: 'toto'
    cellarray: 1
      vecteur: [4x1 double]


ans =

      chiffre: 1
       chaine: 'toto'
    cellarray: 'toto'
      vecteur: [4x1 double]
```

3 Structures de contrôle

3.1 IF ELSE/ELSEIF END

Syntaxe :

```
if expression1
    statements1
elseif expression2
    statements2
else
    statements3
end
```

Exemple :

```
if ((attendance >= 0.90) && (grade_average >= 60))
   pass = false;
end;
```

3.2 SWITCH CASE OTHERWISE END

Syntaxe :

```
switch switch_expr
  case case_expr
    statement, ..., statement
  case {case_expr1, case_expr2, case_expr3, ...}
    statement, ..., statement
  otherwise
    statement, ..., statement
end
```

Exemple :

```
method = 'Bilinear';

switch lower(method)
   case {'linear','bilinear'}
      disp('Method is linear')
   case 'cubic'
      disp('Method is cubic')
   case 'nearest'
      disp('Method is nearest')
   otherwise
      disp('Unknown method.')
end
```

3.3 WHILE END

Syntaxe :

```
while expression
    statements
end
```

Exemple :

```
eps = 1;
while (1+eps) > 1
    eps = eps/2;
end
```

3.4 FOR END

Syntaxe :

```
for variable = expression
    statement
    ...
    statement
end
```

Exemple :

```
a = zeros(k,k)  % Preallocate matrix
for m = 1:k
    for n = 1:k
        a(m,n) = 1/(m+n -1);
    end
end
```

4 Création de fonctions

Comme en Java, il est de bon ton d'avoir un fichier <.M> par fonction, et qu'il porte le nom de la fonction. Attention : la casse est importante. Faites aussi attention de bien mettre vos répertoires de travail dans le PATH, sinon Matlab ne pourra jamais trouver vos fonctions.

Quand vous tapez `>> help repmat` vous affichez l'aide de la fonction. Il est très simple de faire pareil : Matlab affichera comme aide tout commentaire (%) qui suivra immédiatement le prototype de la fonction

4.1 Fonctions simples

Syntaxe :

```
function [out1, out2, ...] = funname(in1, in2, ...)
```

Crée une fonction de nom funame qui prend in1, in2, ...paramètres et en renvoie autant. Il n'est pas nécessaire de mettre un end. Exemples :

```
function [mean,stdev] = stat(x)
% STAT - renvoie la moyenne et l'écart-type d'une matrice entrée.
% Use :
```

```
% [mean,stdev] = stat(x);
%
% Example :
% >> x = stat([1 2 3 4 5]);

% ceci ne sera pas affiché dans l'aide
n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2/n));
```

Petite fonction simple, qui calcule la moyenne d'un vecteur ou d'une matrice et son écart-type. Voyez comment on renvoie 2 arguments en sortie. A l'exécution, si on retourne le résultat de la fonction dans une seule variable, nous y trouverons la sortie la plus à gauche, c'est à dire la moyenne ici.

```
function x = fnan(mode, x, y)
% FNAN - replace nan by a value
% use:
%  x = fnan('replace',x,value_to_replace_nan)
% or
%  x = fnan('revert', x,value_to_revert_to_nan)

if iscell( x)
   x = cellfun(@(t)(fnan(mode,t,y)), x, 'UniformOutput',false);
   return
end

switch lower(mode)
    case {'replace','nan->value'}
        idx    = isnan(x);
        if sum(idx)
            x(idx) = y;
        end
    case {'revert','reverse','value->nan'}
        idx = (abs(x-y)<eps);
        if sum(idx)
            x(idx) = nan;
        end
    otherwise
        error('fnan:mode','unknown mode <%s>',mode);
end
```

Cette fonction remplace la valeur NaN (valeur manquante) dans une matrice par une autre numérique. Elle me sert généralement avant une insertion dans une base de données où je gère une table de correspondances entre le NaN de matlab et une valeur représentant la valeur manquante dans ma table.

4.2 Fonctions anonymes

Il existe quantité de fonctions sous Matlab, qui prennent en paramètre un pointeur (handler) vers une autre fonction. Si cette fonction est courte, nous n'aurons peut-être pas l'envie de créer un fichier .M. Les fonctions anonymes, vous vous en doutez, n'ont pas de nom. Une fois déclarée, elles renvoient un handler sur elle même.

```
>> t = @(x,y) x+y;
>> t(2,3)

ans =

     5
```

Dans cette exemple, je crée une fonction anonyme et je récupère son handler dans la variable t. Ensuite j'utilise t comme une fonction. C'est le seul moyen de définir une fonction dans l'interpréteur sans créer un fichier <.M>.

```
% Keep only workdays
estNull  = cellfun(@isnan,res(:,2:(end-1)));
res      = res(~estNull,:);
```

Dans cet exemple, je supprime de la matrice <res> toutes les lignes qui ont des NaN sur toutes les colonnes autre que la première. Notez le <...> qui permet de mettre un retour à la ligne dans une instruction.

Détaillons : j'ai une matrice de 5 colonnes : la première est la date, les autres sont des données. C'est pour cela que j'exclus à chaque fois la première colonne avec <`(:,2:(end-1))`>. Avec <cellfun> j'applique une fonction sur chaque cellule de <`res(:,2:(end-1))`>, cette fonction est une fonction Matlab standard dont je donne la référence. J'aurais très bien pu définir ici une fonction anonyme. J'obtiens donc une matrice logique (de 0 et de 1). Si estNull vaut <true>, il s'agit d'une ligne que je veux supprimer. C'est pourquoi dans la dernière ligne, je ne garde que les lignes de res (et toutes les colonnes), dont l'indice en ligne vaut <false> dans <estNull>.

4.3 Fonctions à arguments variables

Malheureusement, Matlab ne permet pas en natif de gérer des fonctions avec un nombre variable d'arguments. Il existe juste des variables spéciales, appelées <nvarargin> et <varargin> : la première donne le nombre d'arguments et la deuxième sert à mettre des arguments optionnels, mais tout est à implémenter à chaque fois. Toutefois il existe une solution pour gérer ses arguments optionnels très simplement ; programmée par Charles-Albert Lehalle, elle se trouve ici : `http://literateprograms.org/Swiss_army_knife_Matlab_programs_for_quantitative_finance`.

5 Accès aux bases de données

Matlab est un langage vectoriel, particulièrement à son aise avec les gros volumes de données.Il est donc particulièrement avantageux d'attaquer des bases de données depuis Matlab. On peut ainsi prendre ses données sous Oracle, Access et Excel ou chez d'autres fournisseurs comme Bloomberg, Reuters et Datastream, si l'on est abonné.

5.1 Exécution d'un Select sur Oracle

Récupérer des données sur une base Oracle est extrêment simple, comme un exemple vaut mieux qu'un long discours, voici :

```
function out = execSQL(base,login,password,query)
% EXECSQL - Execute an SQL query into the specified database.
% use :
% >> data = execSQL(base, login, password, query);

conn=database(base,login,password);
cur = exec(conn, query);
res = fetch(cur);
close(cur);
close(conn);
out = res.Data;
```

Il manque à cette fonction une gestion des erreurs : une faute dans la requête serait fatale. La variable <base> est ici le nom d'une connexion ODBC. Il est aussi possible d'utiliser JDBC :

```
conn=database('oracle','scott','tiger',
   'oracle.jdbc.driver.OracleDriver','jdbc:oracle:oci7:')
```

5.2 Récupération de données depuis Bloomberg

La récupération de données chez un provider nécessite la Datafeed Toolbox qui permet de se connecter chez un provider et de l'interroger. Cet exemple ne peut marcher que sur un poste connecté au provider en question.

```
bloom = bloomberg;
data = fetch(b,'IBM US Equity','GETDATA', {'Open';'Last_Price'})
close(bloom);
```

Comme vous le voyez, c'est très simple, si l'on connait le "ticker" et les noms des colonnes que l'on souhaite : dans ce cas rien ne vaut l'expérimentation sur un terminal.

6 Gestion des erreurs

6.1 TRY CATCH END

Syntaxe :

```
try,
   statement,
   ...,
   statement,
catch,
   statement,
   ...,
   statement,
end
```

Comme on peut s'y attendre, tout comme en Java entre autres, toute instruction sous le <try> qui échoue lance l'exécution des instructions sous le <catch>. L'erreur est récupére dans <lasterror>. On peut aussi relancer l'erreur avec <rethrow>.

7 Vectorisation

Vectoriser ses traitements, voila enfin le vif du sujet. Cela demande beaucoup de réflexion, pour écrire 2 lignes qui feront autant que 10 écrites rapidement avec des boucles imbriquées. Toutefois, vous verrez à l'usage qu'il n'est pas

toujours souhaitable de vectoriser toutes les opérations : on revient au vieux dilemmes de l'informatique : vitesse contre mémoire. En effet, certaines matrices intermédiaires seront tellement énormes que Matlab renverra une erreur, faute de mémoire vive. Il faut alors "vectoriser par paquets de données". Personnellement, je travaille avec 512 Mo de RAM, mais certains collègues se plaignent encore parfois avec leurs 2 Go.

7.1 Le pour et le contre

Vectoriser vos traitements a aussi du mauvais, par exemple la portabilité de vos codes Matlab vers d'autres langages est fortement amoindrie si vous avez correctement vectorisé votre code : il est bien plus simple de porter vers Java ou C un code "bas niveau" principalement à base de boucles imbriquées, plutôt que 3 lignes de Matlab ultra condensées. Certains vous dirons qu'un code Matlab bien vectorisé est plus clair, car moins verbeux que le même code "bas niveau". Je suis moyennement d'accord ; à moins de connaître Matlab sur le bout des doigts, un code non vectorisé me semble plus clair à lire moins difficile à écrire.

7.2 Combinaison de fonctions de bases

Matlab combine déjà par lui même 6 de ses fonctions de bases qu'il est bon de connaître :

```
  length(x) est équivalent à max(size(x))
   ndims(x) est équivalent à length(size(x))
   numel(x) est équivalent à prod(size(x))
 isempty(x) est équivalent à numel(x) == 0
   isinf(x) est équivalent à abs(x) == Inf
isfinite(x) est équivalent à abs(x) ~= Inf
```

7.3 Calculer la norme L2 d'un vecteur

Tout simplement : `m = sum(x.^2);`

7.4 Utiliser les index

Utilisez le plus possible les index, quand vous désirez sélectionner des lignes ou des colonnes dans vos matrices, comme par exemple ici :

```
X = 1:10;
index = X>6;
```

```
X = X(index); % retire de X toutes les valeurs <=6
```

Avec les index, vous pouvez ainsi extraire exactement en une instruction, les données qui vous intéressent. Ici nous allons extraire grâce aux index les valeurs sur la diagonales d'une matrice :

```
>> x = magic(5); x(logical(eye(5)))

ans =

    17
     5
    13
    21
     9
```

Evidement, on aurait pu simplement faire `diag(x)` !

Plus simplement, on peut extraire le 1er le second et le dernier élement d'un vecteur : `v([1 2 end])`. Si on veut les 10 derniers éléments : `v(end-10:end)`

7.5 Initialisation d'une matrice

Afin de gagner en vitesse, on peut très bien initialiser une matrice avec une valeur quelconque. Par exemple `A = ones(5,5);` va créer une matrice de (5x5) de 1 et `B = zeros(3,5);` une matrice de zéros. Si on souhaite une autre valeur, on peut simplement faire `C = ones(5,5)*7` pour avoir une matrice de 7.

Autre instruction gadget permettant surtout de faire des tests, `O = magix(5);` va créer un carré magique de taille 5. Plus intéressant, avec `Id = eye(4,4)` permet de créer une matrice identité de la taille souhaitée.

7.6 Les opérations matricielles

. Vous n'êtes pas en C ! Vous perdez votre énergie à imbriquer les boucles pour parcourir vos Matrices, Matlab le fait déjà bien mieux que vous ne pourriez jamais le faire. Voici un exemple de ce qu'il en faut pas faire :

```
for i = 1:n
  z(i) = x(i) * y(i);
end

for i = 1:n
```

```
 for j = 1:m
  if A(i,j) > B(i,j)
   C(i,j) = D(i,j);
  else
   C(i,j) = F(i,j);
  end
 end
end

for i = 1:n-2
 Z(i) = 4*X(i+2) + 2*X(i+1) + X(i);
end
```

Voici maintenant le même code, vectorisé :

```
z = x .* y;

J = A > B;
C = D .* J + F .* (~J);

W = [4 2 1];
Z = filter(W,1,X);
Z = Z(3:n);
```

Cet exemple est tiré de la FAQ du forum usenet comp.soft-sys.matlab.

Voici maintenant la mauvaise et la bonne façon de mettre les éléments d'un vecteur ou d'une matrice au carré :

```
% mauvais
x = [ 1 2 3 4 5 ];
y = zeros(size(x));
for i = 1 : numel(x)
  y(i) = x(i)^2;
end

% bon !
x = [ 1 2 3 4 5 ];
y = x.^2;
```

7.7 Suppression des lignes NaN

Voici un petit exemple, qui permet de supprimer les lignes d'une matrice si elles contiennent une valeur manquante :

```
out = [];
for i=1:size(in,1)
   isAllNaN = true;
   for j=2:size(in,2)
       if ~isnan(in{i,j})
          isAllNaN = false;
       end
   end
   if ~isAllNaN
      out = [out ; in(i,:)];
   end
end
```

Voyons maintenant une version vectorisée du même traitement, déjà utilisé pour illustrer les fonctions anonymes :

```
notnan  = sum(cellfun(@(x) ~isnan(x),in(:,2:end)),2);
out     = in(notnan~=0,:);
```

Non seulement ce code est plus cours, mais il est aussi plus robuste, plus flexible et 2 fois plus rapide que le premier. Certains benchs mettent Matlab devant le C++ pour les calculs matriciels. C'est exemple a déjà été explicité en détail dans cet article, dans le paragraphe sur les fonctions anonymes.

7.8 Comment remplacer tous les NaN par zéro ?

Si vous avez une version ancienne de Matlab (< R14) ceci fonctionne :

```
>> x =[1 ; 3 ;NaN; 4 ; 5]

x =

     1
     3
   NaN
     4
     5

>> x(isnan(x)) = zeros(size(x(isnan(x))))

x =
```

```
1
3
0
4
5
```

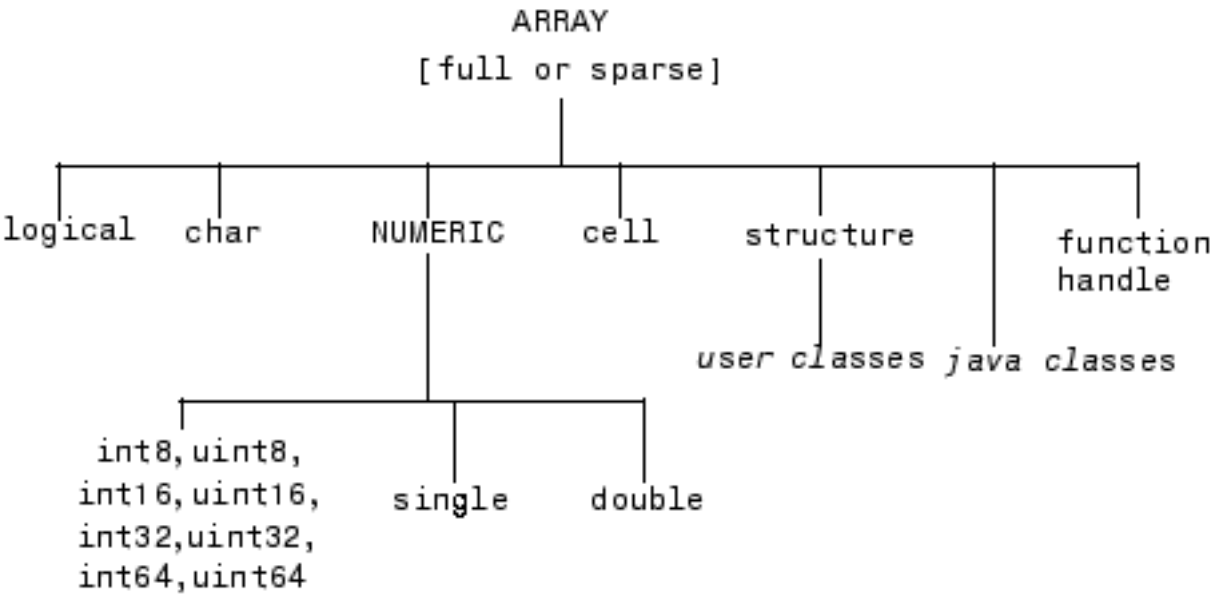
Avec la 2006a, on peut tout simplement faire `x(isnan(x)) = 0;`

8 Programmation Objet

Matlab est étroitement lié à Matlab, il est possible d'utiliser les API java depuis Matlab : on trouve par exemple des implémentations Matlab de tables de hashages s'appuyant sur l'API Java. Par exemple mhashtable `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=9162&objectType=file` développée par Mikhail Poda Razumtsev.

8.1 L'objet selon Matlab

8.1.1 Les types de base



Tous les types de bases dérivent les un des autres comme vous pouvez le voir sur ce schéma. Ainsi vous pouvez utiliser `class(x)` pour connaître la classe d'un objet.

8.1.2 Création d'une classe

Il vous faut un répertoire dont le nom commencera par @ suivi du nom de la classe. A l'intérieur vous devrez mettre un répertoire nommé <private>. Chaque méthode devra être écrire dans une fichier <.M> séparé et placé soit à la racine (si la méthode est publique) soit dans le répertoire <private>. Le constructeur doit être dans un fichier <.M> du même nom que la classe et du même nom que le répertoire en @. Je suis d'accord, ce n'est pas pratique du tout.

8.1.3 Surcharge d'opérateurs

Par contre, grâce à ce type d'objets (contrairement à la solution pratique de la section suivante), vous pourrez surcharger des opérateurs (oui comme en C++ ou en Python). Il vous suffira de créer un fichier <.M> appelé "plus.m" qui contiendra une fonction prenant 2 arguments, pour surcharger l'addition. Voici la liste de ces noms d'opérateurs :

- plus(a,b)
$$a + b$$
- minus(a,b)
$$a - b$$
- uminus(a)
$$-a$$
- uplus(a)
$$+a$$
- times(a,b)
$$a. * b$$
- mtimes(a,b)
$$a * b$$
- rdivide(a,b)
$$a./b$$
- ldivide(a,b)
$$a.\backslash b$$
- mrdivide(a,b)
$$a/b$$
- mldivide(a,b)
$$a./b$$
- power(a,b)
$$a\ .\widehat{}\ b$$

- mpower(a,b)

$$a \hat{} b$$

- lt(a,b)

$$a < b$$

- gt(a,b)

$$a > b$$

- le(a,b)

$$a <= b$$

- ge(a,b)

$$a >= b$$

- ne(a,b)

$$a \ = b$$

- eq(a,b)

$$a == b$$

- and(a,b)

$$a \& b$$

- or(a,b)

$$a | b$$

- not(a)

$$a$$

- colon(a,d,b)

$$a : d : b$$

- colon(a,b)

$$a : b$$

- ctranspose(a)

$$a'$$

- transpose(a)

$$a.'$$

- display(a) : affichage, sortie à l'écran.
- horzcat(a,b,...)

$$[a\ b]$$

- vertcat(a,b,...)

$$[a; b]$$

- subsref(a,s)

$$a(s1, s2, ...sn)$$

– subsasgn(a,s,b)

$$a(s1, ..., sn) = b$$

– subsindex(a)

$$b(a)$$

8.1.4 Héritage simple et multiple

Je lis dans la documentation de Matlab que la syntaxe est la suivante : `derivedObj = class(derivedObj, 'der`
avec comme exemple : `c = class(c, 'derivedClassname', baseObject);`.
Ce n'est malheureusement pas très clair, mais c'est possible !

Ici : `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=2915&objectType=file` Stijn Helsen a mis les sources d'exemples d'héritage avec Matlab. Il crée 2 classes : samplebase et samplederived. Dans le constructeur de samplederived (donc dans le fichier `\ldots\ backslash @samplederived\backslash sample` il écrit juste :

```
function c=samplederived
% samplederived-constructor

c=class(struct([]),'samplederived',samplebase);
```

Il s'agit donc, d'une sorte d'équivalent au `super()` de Java qui en plus défini l'héritage.

8.2 Une classe plus simplement

Finalement, qu'est ce qu'un objet, sinon des données et des pointeurs sur fonctions ? C'est exactement ce que j'utilise ici, pour créer une classe pour la gestion des tables de hachage :

```
function h = myhashtable()
% MYHASHTABLE - my hashtable
% use :
%  myhash = myhashtable % create
% methods :
%  - get('key')
%  - add('key', value)
%  - remove('key')
%  - isKey('key')
%  - isValue('value')
```

```
% example :
%  myhash = myhashtable
%  myhash.add('toto',1);
%  myhash.add('tata',2);
%  [v,k] = myhash.get()
```

Donc, nous allons créer une classe <myhashtable> avec des nested tables et une structure. N'oubliez pas que l'usage des nested tables rend le <end> de fin de fonction obligatoire.

```
this = struct('keys', {{}}, 'values', {{}});
```

Voici la structure dans laquelle nous allons stocker notre table de hachage. Pourquoi les doubles accolades ? Si vous ne voyez vraiment pas retournez à la section sur les Cells Arrays ! Petite indication : je veux une structure contenant 2 Cells Arrays, et pas autant de structures qu'il n'y a d'éléments (ici vides !) dans les Cells Arrays. Je l'ai appelé <this> pour ne pas perturber les programmeurs Java mais on aurait aussi pu l'appeler <self> pour les programmeurs Python.

```
h = struct('get',@get_, ...
    'getKey',@getKey, ...
    'add',@add , ...
    'remove',@remove, ...
    'isKey',@iskey, ...
    'isValue',@isvalue, ...
    'isEmpty',@isempty_);
```

Voila notre structure, qui va nous permettre de créer une classe à base de nested tables. Notre fonction renverra cette structure faite de références sur fonctions. Notez la fonction <@get_>, car le mot "get" est déjà utilisé par une fonction Matlab. Le reste du code est trivial, je vous encourage toutefois si vous débutez, d'essayer d'implémenter vous mêmes chacune des méthodes.

```
    % Ajoute un élément à notre table de hachage
    function idx = add(key, value)
        if isempty(this.keys)
            this.keys{end+1}   = key;
            this.values{end+1} = value;
            idx = length(this.keys);
```

```
        return
    end
    [v, idx] = get_(key);
    if isempty(idx)
        this.keys{end+1}   = key;
        this.values{end+1} = value;
        idx = length(this.keys);
    else
        this.values{idx} = value;
    end
end

% récupère la valeur de l'élément dont la clé est key
% ainsi que son indice
function [value, idx] = get_(key)
    if nargin == 0
        value = {this.keys,this.values};
        return
    end
    value = [];
    idx = strmatch(key,this.keys,'exact');
    if ~isempty(idx)
        value = this.values(idx);
    end
end

% renvoie la clé ou les clés correspondantes à une valeur
function [key, idx] = getKey(value)
    if nargin == 0
        idx = this.values;
        key = this. keys;
        return
    end
    key = [];
    idx = strmatch(value,this.values,'exact');
    if ~isempty(idx)
        key = this.keys(idx);
    end
end

% Supprime l'élément dont la clé est key
```

```
    function idx = remove(key)
        [v,idx] = get_(key);
        if ~isempty(idx)
            this.keys{idx} = [];
            this.values{idx} = [];
        end
    end


    % est ce que key est déjà utilisé comme clé ?
    function out = iskey(key)
        if nargin == 0
            out = 0;
            return;
        end
        out = strmatch(key,this.keys,'exact');
    end

    % est ce qu'on a déjà val comme valeur dans la table ?
    function out = isvalue(val)
        if nargin == 0
            out = 0; return;
        end
        out = strmatch(val,this.values);
    end

    % est ce que la table est vide ?
    function out = isempty_()
        if isempty(h.values) && isempty(h.keys)
            out = 1;
        else
            out = 0;
        end
    end
end % fin de la fonction principale (ie de la classe)
```

En conclusion, si vous avez besoin d'héritage, utilisez l'objet Matlab officiel, sinon utilisez les structures et les références vers fonctions.

9 Licence

GNU Free Documentation License Avertissement

Cette traduction n'a rien d'officiel. La seule copie officielle est celle disponible sur le site de la copyleft. Ce document n'a qu'un caractère indicatif, afin de vous permettre de mieux comprendre les subtilités de cette Licence. N'hésitez pas à nous signaler toute incohérence ou mauvaise traduction par rapport au texte original. Historique

6 mars 2001. Je passe le Petit Journal en GFDL et en entreprends une première traduction. 8 mars 2001. Jean-Luc Fortin m'envoie la traduction qu'il en avait faite en juillet 2000, numérotée 1.0 FR 11 mars 2001. Christian Casteyde a repris le bébé et a entièrement relu et corrigé le texte. 12 mars 2001. Je publie ici la version 1.1.1 FR que j'ai relue et corrigée ce jour (quelques fautes d'orthographe et omissions). 13 mars 2001. Après relecture par Christian, quelques corrections mineures. 14 mars 2001. Voici la version "stable" de cette traduction numérotée 1.1.2 FR Licence de documentation libre GNU Disclaimer

This is an unofficial translation of the GNU Free Documentation License into French. It was not published by the Free Software Foundation, and does not legally state the distribution terms for documentation that uses the GNU FDL– only the original English text of the GNU FDL does that. However, we hope that this translation will help French speakers understand the GNU FDL better.

Ceci est une traduction française non officielle de la Licence de documentation libre GNU. Elle n'a pas été publiée par la Free Software Foundation, et ne fixe pas légalement les conditions de redistribution des documents qui l'utilisent – seul le texte original en anglais le fait. Nous espérons toutefois que cette traduction aidera les francophones à mieux comprendre la FDL GNU.

Traduction française non officielle de la GFDL Version 1.1 (Mars 2000)
Copyright original :

Copyright (C) 2000 Free Sofware Foundation, inc

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Pour la traduction : Version 1.0 FR (Jean-Luc Fortin, juillet 2000) Version 1.1 FR (Christian Casteyde, mars 2001) Version 1.1.1 FR (César Alexanian, mars 2001) Version 1.1.2r2 FR (Christian Casteyde et César Alexanian, juin 2001)

Chacun est libre de copier et de distribuer des copies conformes de cette Licence, mais nul n'est autorisé à la modifier.
0 - PRÉAMBULE

L'objet de cette Licence est de rendre tout manuel, livre ou autre document écrit « libre » au sens de la liberté d'utilisation, à savoir : assurer à chacun la liberté effective de le copier ou de le redistribuer, avec ou sans modifications,

commercialement ou non. En outre, cette Licence garantit à l'auteur et à l'éditeur la reconnaissance de leur travail, sans qu'ils soient pour autant considérés comme responsables des modifications réalisées par des tiers.

Cette Licence est une sorte de « copyleft », ce qui signifie que les travaux dérivés du document d'origine sont eux-mêmes « libres » selon les mêmes termes. Elle complète la Licence Publique Générale GNU, qui est également une Licence copyleft, conçue pour les logiciels libres.

Nous avons conçu cette Licence pour la documentation des logiciels libres, car les logiciels libres ont besoin d'une documentation elle-même libre : un logiciel libre doit être accompagné d'un manuel garantissant les mêmes libertés que celles accordées par le logiciel lui-même. Mais cette Licence n'est pas limitée aux seuls manuels des logiciels ; elle peut être utilisée pour tous les documents écrits, sans distinction particulière relative au sujet traité ou au mode de publication. Nous recommandons l'usage de cette Licence principalement pour les travaux destinés à des fins d'enseignement ou devant servir de documents de référence.
1 - APPLICABILITÉ ET DÉFINITIONS

Cette Licence couvre tout manuel ou tout autre travail écrit contenant une notice de copyright autorisant la redistribution selon les termes de cette Licence. Le mot « Document » se réfère ci-après à un tel manuel ou travail. Toute personne en est par définition concessionnaire et est référencée ci-après par le terme « Vous ».

Une « Version modifiée » du Document désigne tout travail en contenant la totalité ou seulement une portion de celui-ci, copiée mot pour mot, modifiée et/ou traduite dans une autre langue.

Une « Section secondaire » désigne une annexe au Document, ou toute information indiquant les rapports entre l'auteur ou l'éditeur et le sujet (ou tout autre sujet connexe) du document, sans toutefois être en rapport direct avec le sujet lui-même (par exemple, si le Document est un manuel de mathématiques, une Section secondaire ne traitera d'aucune notion mathématique). Cette section peut contenir des informations relatives à l'historique du Document, des sources documentaires, des dispositions légales, commerciales, philosophiques, ou des positions éthiques ou politiques susceptibles de concerner le sujet traité.

Les « Sections inaltérables » sont des sections secondaires considérées comme ne pouvant être modifiées et citées comme telles dans la notice légale qui place le Document sous cette Licence.

Les « Textes de couverture » sont les textes courts situés sur les pages de couverture avant et arrière du Document, et cités comme tels dans la mention légale de ce Document.

Le terme « Copie transparente » désigne une version numérique du Document

représentée dans un format dont les spécifications sont publiquement disponibles et dont le contenu peut être visualisé et édité directement et immédiatement par un éditeur de texte quelconque, ou (pour les images composées de pixels) par un programme de traitement d'images quelconque, ou (pour les dessins) par un éditeur de dessins courant. Ce format doit pouvoir être accepté directement ou être convertible facilement dans des formats utilisables directement par des logiciels de formatage de texte. Une copie publiée dans un quelconque format numérique ouvert mais dont la structure a été conçue dans le but exprès de prévenir les modifications ultérieures du Document ou dans le but d'en décourager les lecteurs n'est pas considérée comme une Copie Transparente. Une copie qui n'est pas « Transparente » est considérée, par opposition, comme « Opaque ».

Le format de fichier texte codé en ASCII générique et n'utilisant pas de balises, les formats de fichiers Texinfo ou LaTeX, les formats de fichiers SGML ou XML utilisant une DTD publiquement accessible, ainsi que les formats de fichiers HTML simple et standard, écrits de telle sorte qu'ils sont modifiables sans outil spécifique, sont des exemples de formats acceptables pour la réalisation de Copies Transparentes. Les formats suivants sont opaques : PostScript, PDF, formats de fichiers propriétaires qui ne peuvent être visualisés ou édités que par des traitements de textes propriétaires, SGML et XML utilisant des DTD et/ou des outils de formatage qui ne sont pas disponibles publiquement, et du code HTML généré par une machine à l'aide d'un traitement de texte quelconque et dans le seul but de la génération d'un format de sortie.

La « Page de titre » désigne, pour les ouvrages imprimés, la page de titre elle-même, ainsi que les pages supplémentaires nécessaires pour fournir clairement les informations dont cette Licence impose la présence sur la page de titre. Pour les travaux n'ayant pas de Page de titre comme décrit ci-dessus, la « Page de titre » désigne le texte qui s'apparente le plus au titre du document et situé avant le texte principal.

2 - COPIES CONFORMES

Vous pouvez copier et distribuer le Document sur tout type de support, commercialement ou non, à condition que cette Licence, la notice de copyright et la notice de la Licence indiquant que cette Licence s'applique à ce Document soient reproduits dans toutes les copies, et que vous n'y ajoutiez aucune condition restrictive supplémentaire. Vous ne pouvez pas utiliser un quelconque moyen technique visant à empêcher ou à contrôler la lecture ou la reproduction ultérieure des copies que vous avez créées ou distribuées. Toutefois, vous pouvez solliciter une rétribution en échange des copies. Si vous distribuez une grande quantité de copies, référez-vous aux dispositions de la section 3.

Vous pouvez également prêter des copies, sous les mêmes conditions que celles suscitées, et vous pouvez afficher publiquement des copies de ce Document.

3 - COPIES EN NOMBRE

Si vous publiez des copies imprimées de ce Document à plus de 100 exemplaires et que la Licence du Document indique la présence de Textes de couverture, vous devez fournir une couverture pour chaque copie, qui présente les Textes de couverture des première et dernière pages de couverture du Document. Les première et dernière pages de couverture doivent également vous identifier clairement et sans ambiguïté comme étant l'éditeur de ces copies. La première page de couverture doit comporter le titre du Document en mots d'importance et de visibilité égales. Vous pouvez ajouter des informations complémentaires sur les pages de couverture. Les copies du Document dont seule la couverture a été modifiée peuvent être considérées comme des copies conformes, à condition que le titre du Document soit préservé et que les conditions indiquées précédemment soient respectées.

Si les textes devant se trouver sur la couverture sont trop importants pour y tenir de manière claire, vous pouvez ne placer que les premiers sur la première page et placer les suivants sur les pages consécutives.

Si vous publiez plus de 100 Copies opaques du Document, vous devez soit fournir une Copie transparente pour chaque Copie opaque, soit préciser ou fournir avec chaque Copie opaque une adresse réseau publiquement accessible d'une Copie transparente et complète du Document, sans aucun ajout ou modification, et à laquelle tout le monde peut accéder en téléchargement anonyme et sans frais, selon des protocoles réseau communs et standards. Si vous choisissez cette dernière option, vous devez prendre les dispositions nécessaires, dans la limite du raisonnable, afin de garantir l'accès non restrictif à la Copie transparente durant une année pleine après la diffusion publique de la dernière Copie opaque (directement ou via vos revendeurs).

Nous recommandons, mais ce n'est pas obligatoire, que vous contactiez l'auteur du Document suffisamment tôt avant toute publication d'un grand nombre de copies, afin de lui permettre de vous donner une version à jour du Document.

4 - MODIFICATIONS

Vous pouvez copier et distribuer une Version modifiée du Document en respectant les conditions des sections 2 et 3 précédentes, à condition de placer cette Version modifiée sous la présente Licence, dans laquelle le terme « Document » doit être remplacé par les termes « Version modifiée », donnant ainsi l'autorisation de redistribuer et de modifier cette Version modifiée à quiconque en possède une copie. De plus, vous devez effectuer les actions suivantes dans la Version modifiée :

Utiliser sur la Page de titre (et sur la page de couverture éventuellement présente) un titre distinct de celui du Document d'origine et de toutes ses versions antérieures (qui, si elles existent, doivent être mentionnées dans la section « His-

torique » du Document). Vous pouvez utiliser le même titre si l'éditeur d'origine vous en a donné expressément la permission.

Mentionner sur la Page de titre en tant qu'auteurs une ou plusieurs des personnes ou entités responsables des modifications de la Version modifiée, avec au moins les cinq principaux auteurs du Document (ou tous les auteurs s'il y en a moins de cinq).

Préciser sur la Page de titre le nom de l'éditeur de la Version modifiée, en tant qu'éditeur du Document.

Préserver intégralement toutes les notices de copyright du Document.

Ajouter une notice de copyright adjacente aux autres notices pour vos propres modifications.

Inclure immédiatement après les notices de copyright une notice donnant à quiconque l'autorisation d'utiliser la Version modifiée selon les termes de cette Licence, sous la forme présentée dans l'annexe indiquée ci-dessous.

Préserver dans cette notice la liste complète des Sections inaltérables et les Textes de couverture donnés avec la notice de la Licence du Document.

Inclure une copie non modifiée de cette Licence.

Préserver la section nommée « Historique » et son titre, et y ajouter une nouvelle entrée décrivant le titre, l'année, les nouveaux auteurs et l'éditeur de la Version modifiée, tels que décrits sur la Page de titre, ainsi qu'un descriptif des modifications apportées depuis la précédente version.

Conserver l'adresse réseau éventuellement indiquée dans le Document permettant à quiconque d'accéder à une Copie transparente du Document, ainsi que les adresses réseau indiquées dans le Document pour les versions précédentes sur lesquelles le Document se base. Ces liens peuvent être placés dans la section « Historique ». Vous pouvez ne pas conserver les liens pour un travail datant de plus de quatre ans avant la version courante ou si l'éditeur d'origine vous en accorde la permission.

Si une section « Dédicaces » ou une section « Remerciements » sont présentes, les informations et les appréciations concernant les contributeurs et les personnes auxquelles s'adressent ces remerciements doivent être conservées, ainsi que le titre de ces sections.

Conserver sans modification les Sections inaltérables du Document, ni dans leurs textes, ni dans leurs titres. Les numéros de sections ne sont pas considérés comme faisant partie du texte des sections.

Effacer toute section intitulée « Approbations ». Une telle section ne peut pas être incluse dans une Version modifiée.

Ne pas renommer une section existante sous le titre « Approbations » ou sous un autre titre entrant en conflit avec le titre d'une Section inaltérable.

Si la Version modifiée contient de nouvelles sections préliminaires ou de nouvelles annexes considérées comme des Sections secondaires et que celles-ci ne contiennent aucun élément copié à partir du Document, vous pouvez à votre convenance en désigner une ou plusieurs comme étant des Sections inaltérables. Pour ce faire, ajoutez leurs titres dans la liste des Sections inaltérables au sein de la notice de Licence de la version Modifiée. Ces titres doivent êtres distincts des titres des autres sections.

Vous pouvez ajouter une section nommée « Approbations » à condition que ces approbations ne concernent que les modifications ayant donné naissance à la Version modifiée (par exemple, comptes rendus de revue du document ou acceptation du texte par une organisation le reconnaissant comme étant la définition d'un standard).

Vous pouvez ajouter un passage comprenant jusqu'à cinq mots en première page de couverture, et jusqu'à vingt-cinq mots en dernière page de couverture, à la liste des Textes de couverture de la Version modifiée. Il n'est autorisé d'ajouter qu'un seul passage en première et en dernière pages de couverture par personne ou groupe de personnes ou organisation ayant contribué à la modification du Document. Si le Document comporte déjà un passage sur la même couverture, ajouté en votre nom ou au nom de l'organisation au nom de laquelle vous agissez, vous ne pouvez pas ajouter de passage supplémentaire ; mais vous pouvez remplacer un ancien passage si vous avez expressément obtenu l'autorisation de l'éditeur de celui-ci.

Cette Licence ne vous donne pas le droit d'utiliser le nom des auteurs et des éditeurs de ce Document à des fins publicitaires ou pour prétendre à l'approbation d'une Version modifiée.

5 - FUSION DE DOCUMENTS

Vous pouvez fusionner le Document avec d'autres documents soumis à cette Licence, suivant les spécifications de la section 4 pour les Versions modifiées, à condition d'inclure dans le document résultant toutes les Sections inaltérables des documents originaux sans modification, et de toutes les lister dans la liste des Sections inaltérables de la notice de Licence du document résultant de la fusion.

Le document résultant de la fusion n'a besoin que d'une seule copie de cette Licence, et les Sections inaltérables existant en multiples exemplaires peuvent être remplacées par une copie unique. S'il existe plusieurs Sections inaltérables portant le même nom mais de contenu différent, rendez unique le titre de chaque section en ajoutant, à la fin de celui-ci, entre parenthèses, le nom de l'auteur ou de l'éditeur d'origine, ou, à défaut, un numéro unique. Les mêmes modifications

doivent être réalisées dans la liste des Sections inaltérables de la notice de Licence du document final.

Dans le document résultant de la fusion, vous devez rassembler en une seule toutes les sections « Historique » des documents d'origine. De même, vous devez rassembler les sections « Remerciements » et « Dédicaces ». Vous devez supprimer toutes les sections « Approbations ».

6 - REGROUPEMENTS DE DOCUMENTS

Vous pouvez créer un regroupement de documents comprenant le Document et d'autres documents soumis à cette Licence, et remplacer les copies individuelles de cette Licence des différents documents par une unique copie incluse dans le regroupement de documents, à condition de respecter pour chacun de ces documents l'ensemble des règles de cette Licence concernant les copies conformes.

Vous pouvez extraire un document d'un tel regroupement et le distribuer individuellement sous couvert de cette Licence, à condition d'y inclure une copie de cette Licence et d'en respecter l'ensemble des règles concernant les copies conformes.

7 - AGRÉGATION AVEC DES TRAVAUX INDÉPENDANTS

La compilation du Document ou de ses dérivés avec d'autres documents ou travaux séparés et indépendants sur un support de stockage ou sur un média de distribution quelconque ne représente pas une Version modifiée du Document tant qu'aucun copyright n'est déposé pour cette compilation. Une telle compilation est appelée « agrégat » et cette Licence ne s'applique pas aux autres travaux indépendants compilés avec le Document s'ils ne sont pas eux-mêmes des travaux dérivés du Document.

Si les exigences de la section 3 concernant les Textes de couverture sont applicables à ces copies du Document, et si le Document représente un volume inférieur à un quart du volume total de l'agrégat, les Textes de couverture du Document peuvent être placés sur des pages de couverture qui n'encadrent que le Document au sein de l'agrégat. Dans le cas contraire, ils doivent apparaître sur les pages de couverture de l'agrégat complet.

8 - TRADUCTION

La traduction est considérée comme une forme de modification, vous pouvez donc distribuer les traductions du Document selon les termes de la section 4. Vous devez obtenir l'autorisation spéciale des auteurs des Sections inaltérables pour les remplacer par des traductions, mais vous pouvez inclure les traductions des Sections inaltérables en plus des textes originaux. Vous pouvez inclure une traduction de cette Licence à condition d'inclure également la version originale en anglais. En cas de contradiction entre la traduction et la version originale en anglais, c'est cette dernière qui prévaut.

9 - RÉVOCATION

Vous ne pouvez pas copier, modifier, sous-licencier ou distribuer le Document autrement que selon les termes de cette Licence. Tout autre acte de copie, modification, sous-Licence ou distribution du Document est sans objet et vous prive automatiquement des droits que cette Licence vous accorde. En revanche, les personnes qui ont reçu de votre part des copies ou les droits sur le document sous couvert de cette Licence ne voient pas leurs droits révoqués tant qu'elles en respectent les principes.

10 - RÉVISIONS FUTURES DE CETTE LICENCE

La Free Software Foundation peut publier de temps en temps de nouvelles versions révisées de cette Licence. Ces nouvelles versions seront semblables à la présente version dans l'esprit, mais pourront différer sur des points particuliers en fonction de nouvelles questions ou nouveaux problèmes. Voyez http ://www.gnu.org/copyleft/ pour plus de détails.

Chaque version de cette Licence est dotée d'un numéro de version distinct. Si un Document spécifie un numéro de version particulier de cette Licence, et porte la mention « ou toute autre version ultérieure », vous pouvez choisir de suivre les termes de la version spécifiée ou ceux de n'importe quelle version ultérieure publiée par la Free Software Foundation. Si aucun numéro de version n'est spécifié, vous pouvez choisir n'importe quelle version officielle publiée par la Free Sofware Foundation.