**Université Aix Marseille II** **Maîtrise MMI**

# INITIATION A MATLAB

Anne PETRENKO (COM, Marseille)

**Edition Avril 2004**

# INITIATION A MATLAB

**Avant-propos** Ce fascicule s'inspire de la présentation de Matlab rédigée par Hoang Le-Huy, Professeur au Département de génie électrique et informatique de l'Université Laval, Québec, au Canada, dans son cours polycopié destiné à ses étudiants (Edition Septembre 98). Je le remercie de m'avoir autorisé à utiliser son travail à des fins d'enseignement au COM, Université Aix-Marseille.

Anne Petrenko

**PLAN**



## Généralités

MATLAB est un système interactif de programmation scientifique, pour le calcul numérique et la visualisation graphique, basé sur la représentation matricielle des données. Le nom dérive de cette représentation : MATLAB = MATrix LABoratory.

Matlab permet de faire des calculs mathématiques 'semi-formels'. Il connaît un grand nombre d'opérations ou de fonctions mathématiques : fonctions usuelles, calcul matriciel, fonctions plus spécifiques du signal, etc.

MATLAB étant disponible sur différentes plateformes matérielles, on accède au logiciel en lançant l'exécutable **matlab** suivant la procédure habituelle de l'environnement concerné. On se retrouve alors dans un environnement dit *fenêtre de commande* dans lequel on peut écrire des commandes ou exécuter des

fonctions. La sortie du logiciel s'effectue en tapant **quit** ou **exit**. Sur les systèmes qui permettent le multi-fenêtrage, on peut travailler sous MATLAB dans une fenêtre et conserver son éditeur de texte dans une autre afin de modifier simplement les fonctions ou 'programmes matlab' à mettre au point.

# 1. INTRODUCTION

Matlab est un logiciel de calcul matriciel à syntaxe simple. Avec ses fonctions spécialisées, Matlab peut aussi être considéré comme un langage de programmation adapté pour les problèmes scientifiques.

Il est largement utilisé par la communauté océanographique dans le monde. Vous pourrez trouver de larges 'boites à outils' de programmes océanographiques déjà écrits sur des sites WEB divers répertoriés au site de Woods Hole suivant:

http://uop.whoi.edu/whit/mug.html

Matlab fonctionne dans plusieurs environnements tels que Linux, X-Windows, Windows, Macintosh.

* Avant la première utilisation de matlab:

Chaque utilisateur doit créer son répertoire de travail (ex., optique), où il pourra enregistrer ses fichiers.

* On lance l'exécutable **matlab** :

Cliquez sur l' icône de matlab ou tapez dans la console de LINUX la commande :

**matlab &**

[note : n'oubliez pas le symbole "&" sinon vous perdrez la main dans la console LINUX. Si vous l'oubliez, appuyez sur **ctrl Z** puis tapez **bg** dans le terminal LINUX]

L'interface Matlab se compose d'une fenêtre principale divisée en 3 sous fenêtres :

(i) En haut à gauche, il y a une fenêtre contenant deux onglets : **Launch Pad** et **Workspace**. L'onglet Launch Pad est visible par défaut. Le Launch Pad ne nous concerne pas ; il s'agit d'une interface pour obtenir de l'information et des démos sur des composantes de Matlab. Nous ne l'utiliserons pas. Le Workspace permet de gérer les variables utilisées dans Matlab.

(ii) En bas à gauche, il y a une fenêtre contenant deux onglets : **Command History** et **Current Directory**. L'onglet Command History est visible par défaut; il indique les dernières commandes effectuées. Le Current Directory gère l'emplacement des fichiers. Celui-ci sera utile pour le travail avec les fichiers de données et les programmes matlab ("m-files").

(iii) Sur la droite, il y a une grande fenêtre : **Command Window**. La Command Window est la fenêtre d'interaction avec Matlab. Quand on a le symbole **>>** à l'écran[1], on peut écrire des commandes ou exécuter des fonctions.

[note : Le symbole [>>] indique à l'utilisateur où il faut rentrer la commande. On ne peut pas "revenir en arrière", c'est-à-dire, il ne faut pas essayer de placer le curseur sur une ligne au-dessus du dernier [>>]. Pour taper une autre commande on le fait à la suite]

Il existe deux modes de fonctionnement:

1) mode interactif: Matlab exécute les instructions au fur et à mesure qu'elles sont données par l'usager

2) mode exécutif: Matlab exécute ligne par ligne un programme matlab (d'extension .m). Un 'programme Matlab' (ou 'm-file' en anglais) est une suite d'instructions matlab écrites dans un éditeur de texte et sauvées dans un fichier avec une extension .m

* A chaque début de session :

L'utilisateur indique à Matlab que le répertoire **optique** défini précédemment est le répertoire de travail de la session,

soit en tapant la commande: `>> cd chemin d'accès au répertoire` **optique**

soit en allant dans l'onglet **Current Directory** et en parcourant les dossiers jusqu'à arriver au dossier **optique**.

## 2. Commandes simples

Chaque ligne est exécutée immédiatement après la touche Entrée. Les commandes de base d'Unix peuvent être utilisées:

**pwd** pour afficher le chemin d'accès au répertoire de travail actuel

**dir** ou **ls** pour obtenir le contenu de ce répertoire

**cd** pour changer de position dans l'arborescence

>> cd /home/ENS/licence/stu/nom; pour aller dans votre répertoire

[ note: Si la commande unix n'est pas reconnue, elle marchera avec un ! devant]

### *2.1 Opérations mathématiques*

## Nombres, opérations arithmétiques et variables

[1] Si ce symbole ne revient pas à la fin d'une instruction; c'est qu'il y a un problème et matlab donne des indications sur l'origine et le moyen de le résoudre

NOMBRES

Les nombres réels peuvent être écrits sous différents formats:

5 1.0237 0.5245E-12 12.78e6 0.001234 -235.087

Les nombres complexes peuvent être écrits sous forme cartésienne ou polaire:

Forme cartésienne: 0.5 + i*2.7 -1.2 + j*0.789 2.5 + 9.7i

Forme polaire: 1.25*exp(j*0.246)

OPÉRATIONS ARITHMÉTIQUES

+ Addition

- Soustraction

* Multiplication

/ Division

^ Puissance

VARIABLES

On attribue une valeur numérique à une variable en tapant directement son expression. Le symbole d'affectation de valeur à une variable est le caractère **=**.

```
>> a=1.25
a =
    1.2500
>> b=1+2
b =
     3
>> c=a*b
c =
    3.7500
```

Voici quelques variables prédéfinies :

- **ans** :résultat de la dernière évaluation
- **pi** : 3.1416…
- **inf** : Infini (1/0)
- **NaN** : "Not a Number" (0/0)
- **i**, **j** : représentent tous les deux le nombre imaginaire unité ($\sqrt{-1}$)

## Vecteurs et matrices

Les vecteurs et les matrices peuvent être construits directement suivant la syntaxe suivante :

- ils sont délimités par des crochets,

- les éléments sont entrés ligne par ligne,
- les éléments appartenant à la même ligne sont séparés par des espaces (ou par des virgules),
- les différentes lignes doivent **comporter le même nombre d'éléments** et sont séparées par des points-virgule.

VECTEURS

On peut définir un vecteur x en donnant la liste de ses éléments:

```
>> x=[0.5       1.2        -3.75      5.82       -0.735]
x =
     0.5000     1.2000     -3.7500    5.8200     -0.7350
```

On a ainsi défini un (1,5) vecteur, *i.e.* un vecteur ligne contenant 5 éléments.

Exemple d'un (5,1) vecteur colonne :

```
>> y=[0.5;1.2;-3.75;5.82;-0.735]
y =
    0.5000
    1.2000
   -3.7500
    5.8200
   -0.7350
```

On peut aussi définir un vecteur en donnant la suite qui forme le vecteur :

```
>> z= 2:0.6:5                    Note: le pas est de 0.6
z =
     2.0000     2.6000     3.2000     3.8000     4.4000     5.0000
```

```
Note z= 2:5          par défaut crée une suite avec un pas unitaire
```

```
>> t=1:5.6
t =
     1     2     3     4     5
        (puisque 5+1 est strictement supérieur à 5.6)
```

ou en utilisant une fonction qui génère un vecteur :

```
>> x =linspace(0.5,2,4)
x =
     0.5000    1.0000    1.5000    2.
```

Note linspace($v_i$,$v_f$,n) crée une liste de n éléments uniformément répartis entre $v_i$ et $v_f$. Ce qui est équivalent à $v_i : \frac{vf - vi}{n-1} : v_f$ .

Remarque:

**Lors qu'on ajoute un «;» à la fin d'une instruction, elle est exécutée mais le résultat n'est pas affiché:**
**(à ne pas oublier dans les programmes ou lors de la manipulation de longs fichiers de données)**

```
>> a=[1 2 3 4 5];
>> b=-2.5;
>> c=b*a;
```

Lors qu'il n'y a pas de «;» à la fin d'une instruction, elle est exécutée et le résultat est affiché:

```
>> a=[1 2 3 4 5]
a =
     1     2     3     4     5
>> b=-2.5
b =
     -2.5000
>> c=b*a
c =
     -2.5000    -5.0000    -7.5000    -10.0000   -12.5000
```

Des vecteurs définis au préalable peuvent être utilisés pour générer d'autres vecteurs, par exemple:

```
>> a=[1 2 3];
>> b=[4 5 6];
>> c=[a b]
c =
   1   2   3   4   5   6
```

Si l'on veut déterminer la longueur d'un vecteur, on utilisera la commande "**length**", cela donne :

```
>> ll=length(c)
ll =
   6
```

MATRICES

On définit aussi une matrice A en donnant ses éléments:

```
>> A=[0.5 2.7 3.9 4;4.5 0.85 -1.23 3;-5.12 2.47 9.03 2]
A =
    0.5000    2.7000    3.9000    4.0000
    4.5000    0.8500   -1.2300    3.0000
   -5.1200    2.4700    9.0300    2.0000
```

Pour en connaître la dimension, on utilise la commande "**size**" :

```
>> [l c]=size(A)
l =
   3
c =
   4
```

Certaines matrices sont prédéfinies dans matlab. Ainsi "**zeros(m,n)**" crée une (m,n) matrice de zéros; "**ones(m,n)**" crée une (m,n) matrice de uns et "**eye(n)**" crée une (n,n) matrice identité. Par exemple :

```
>> B=eye(4)
B =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

EMPLOI DES INDICES

Les éléments d'un vecteur ou d'une matrice peuvent être adressés en utilisant les indices sous la forme suivante:

t(10) élément no. 10 du vecteur t

A(2,9) élément se trouvant à ligne 2, colonne 9 de la matrice A

B(:,7) la colonne 7 de la matrice B

C(3,:) la ligne 3 de la matrice B

OPÉRATIONS MATRICIELLES

Les opérations matricielles exécutées par MATLAB sont illustrées dans le tableau suivant:

B = A' La matrice B est égale à la matrice A transposée

E = inv(A) La matrice E est égale à la matrice A inversée

C = A + B Addition

D = A - B Soustraction

Z = X*Y Multiplication

X = B/A Division (Équivalent à B*inv(A)°

[Note: X = A\B Équivalent à inv(A)*B ]

*Note: Attention aux dimensions des multiplications et divisions de matrice*

Etant donné que les scientifiques - et entre autres, les océanographes - utilisent très souvent des opérations entre deux colonnes d'un tableau de données, une opération spéciale a été adaptée:

OPÉRATION «ÉLÉMENT PAR ÉLÉMENT»

Les opérations «élément par élément» des vecteurs et des matrices sont effectuées en ajoutant un point (.) avant les opérations `* / ^'`

Exemple 1:

```
>> A=[1 2 3 4 5];
>> B=[6 7 8 9 10];
Essayez de faire C = A*B; et justifiez ce qui se passe;
puis essayez la multiplication 'élément par élément'
>> C=A.*B
C =
     6    14    24    36    50
>> D=A./B
D =
     0.1667     0.2857     0.3750     0.4444     0.5000
```

## fonctions

EXPRESSIONS MATHÉMATIQUES

On écrit les expressions mathématiques de la façon habituelle:

```
z = 5*exp(-0.4*x).*sin(7.5*y);
```

FONCTIONS MATHÉMATIQUES

Les fonctions mathématiques de base sont données dans le tableau suivant:

| | | | | |
|---|---|---|---|---|
| min<br>valeur minimale | max<br>valeur maximale | mean<br>valeur moyenne | std<br>écart type | cov<br>covariance |
| sum<br>somme | abs<br>valeur absolue; module (nb complexe) | angle<br>argument (nb. complexe) | real<br>partie réelle | imag<br>partie imaginaire |
| conj<br>conjuguée (nb. complexe) | round<br>arrondir | fix<br>arrondir (vers zéro) | floor<br>arrondir (vers -∞) | ceil<br>arrondir (vers ∞) |
| sqrt<br>racine carrée | rem<br>reste | exp<br>exponentielle | log<br>logarithme base e | log10<br>logarithme base 10 |

Les fonctions trigonométriques sont données dans le tableau suivant:

| | | | | | | |
|---|---|---|---|---|---|---|
| sin | cos | tan | asin | acos | atan | atan2 |
| sinh | cosh | tanh | asinh | acosh | atanh | |

La plupart de ces fonctions sont appliquées élément par élément pour les calculs matriciels.

**FONCTION "HELP"**

Pour obtenir de l'aide sur un sujet, une instruction ou une fonction, on tape help suivi par le sujet, l'instruction ou la fonction désirée.

Exemple 2:

```
>> help mean
 MEAN   Average or mean value.
    For vectors, MEAN(X) is the mean value of the elements in X. For
    matrices, MEAN(X) is a row vector containing the mean value of
    each column.  For N-D arrays, MEAN(X) is the mean value of the
```

```
elements along the first non-singleton dimension of X.

MEAN(X,DIM) takes the mean along the dimension DIM of X.
Example: If X = [0 1 2
                 3 4 5]
then mean(X,1) is [1.5 2.5 3.5] and mean(X,2) is [1
                                                   4]
See also MEDIAN, STD, MIN, MAX, COV.
```

On peut aussi obtenir de l'aide en cliquant sur Help en haut de la fenêtre.

**ESPACE DE TRAVAIL (Workspace)**

Les variables sont définies au fur et à mesure que l'on donne leurs noms et leurs valeurs numériques ou leurs expressions mathématiques.

Les variables ainsi définies sont stockées dans l'espace de travail et peuvent être utilisées dans les calculs subséquents.

**INFORMATION SUR L'ESPACE DE TRAVAIL**

Pour obtenir une liste des variables dans l'espace de travail, on utilise l'onglet Workspace ou sinon on utilise l' instruction suivante à partir de la Command window:

>> whos Affichage détaillé des variables dans l'espace de travail.

ATTENTION: si vous quittez Matlab, vous perdez toutes les variables définies

sauf si:

**ENREGISTRER LES VARIABLES DE L'ESPACE DE TRAVAIL DANS UN FICHIER**

Pour enregistrer les variables de l'espace de travail dans un fichier, on utilise les instructions suivantes:

**>> save** Enregistrer toutes les variables dans un fichier matlab.mat. Dans une session ultérieure, taper >> **load** pour ramener l'espace de travail enregistré.

Si vous voulez enregistrer dans un fichier de nom choisi:

**>>save** fichier1.mat x y z A X

Enregistre les variables x, y, z, A, X dans le fichier fichier1.mat.

Dans une session ultérieure, taper **>> load fichier1.mat** pour ramener les variables x, y, z, A, X dans l'espace de travail.

**Note: Si vous n'êtes pas dans un répertoire dans lequel vous avez le droit d'écrire, Matlab refusera d'exécuter save**

Pour effacer toutes les variables de l'espace de travail, on utilise la commande : >> **clear**

**CHARGEMENT DE FICHIERS DE DONNEES**

1. Si des variables ont été sauvées en format matlab dans le fichier fichier1.mat. Dans la session actuelle ou dans une session ultérieure, vous pouvez recharger ces données en tapant:

**>> load fichier1.mat** (si vous etes dans les répertoire contenant fichier1.mat)

Les données sont récupérées avec le nom des variables qui avaient été sauvées.

2. Si des données sont sous forme de matrices régulières dans un fichier (ex station1.dat ou file500.txt), vous pouvez charger ces données en tapant:

**>> load file500.txt**

la matrice de données est alors disponible dans l'espace matlab sous le nom file500 (pas d'extension).

3. Si des données sont dans un fichier excel (ex sarrefl_TD.xls'), vous pouvez charger ces données en tapant:

**>> [data,texte]=xlsread('sarrefl_TD.xls')**

la matrice de données est alors disponible dans l'espace matlab sous le nom data (pas d'extension). Si il y avait du texte en en-tête des données, il est dans texte.

## *2.2 Graphiques*

### 2.2 a) Graphiques 2D

TRAÇAGE DE COURBES

Par exemple, on utilise l'instruction **plot** pour tracer un graphique 2D:

| | |
|---|---|
| plot(x,y) | Tracer le vecteur y en fonction du vecteur x |
| plot(t,x,t,y,t,z) | Tracer x(t), y(t) et z(t) sur le même graphique |
| plot(t,z,'r--') | Tracer z(t) en trait pointillé rouge |

FORMAT DE GRAPHIQUE

On peut aussi choisir le format du graphique:

| | |
|---|---|
| plot(x,y) | Tracer y(x) avec échelles linéaires |
| semilogx(f,A) | Tracer A(f) avec échelle log(f) |
| semilogy(w,B) | Tracer B(w) avec échelle log(B) |
| polar(theta,r) | Tracer r(theta) en coordonnées polaires |

bar(x,y) Tracer y(x) sous forme des barres

grid Ajouter une grille

Exemple 3:

```
>> t=0:0.01e-3:0.06;
>> y=10*exp(-60*t).*cos(120*pi*t);
>> z=10*exp(-60*t).*sin(120*pi*t);
>> a=10*exp(-60*t);
>> plot(t,y,'r',t,z,'g'),grid
>> hold on
```

Remarque cette commande "hold on" permet de tracer plusieurs graphes sur la même figure; sinon une deuxième commande plot (ou autre) effacera le graphe précédent pour effectuer la dernière commande

```
>> plot(t,a,'b--')
>> plot(t,-a,'b--')
>> title('Fonctions sinusoidales amorties')
>> xlabel('Temps , s'),ylabel('Tension , V')
```

Si vous voulez tracer un autre graphe, vous pouvez faire:

```
>> hold off
>> plot(y,z),grid
>> axis equal
>> xlabel('y'),ylabel('z')
```

hold off recommencera un nouveau graphe à la prochaine commande graphique (en restant dans la même fenêtre graphique),

**ou**

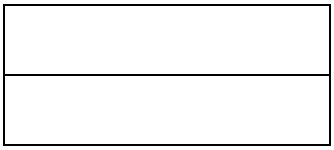
```
>> figure(2)          ouvrira une deuxième fenêtre graphique, sans
                      détruire la première
>> plot(y,z),grid
>> axis equal
>> xlabel('y'),ylabel('z')
```

### 2.2 b) Divers sur les graphes

GRAPHIQUE MULTIPLE

On peut tracer plusieurs graphiques dans la même fenêtre en utilisant l'instruction **subplot** pour diviser la fenêtre en plusieurs parties.

- Diviser la fenêtre en deux parties (2 x 1)




Exemple 3b:

```
>> t=0:0.01e-3:0.06;
>> y=10*exp(-60*t).*cos(120*pi*t);
>> z=10*exp(-60*t).*sin(120*pi*t);
```

```
>> subplot(211),plot(t,y,'r',t,z,'g'),grid
>> a=10*exp(-60*t);
>> hold on
>> plot(t,a,'b--')
>> plot(t,-a,'b--')
>> title('Fonctions sinusoidales amorties')
>> xlabel('Temps , s'),ylabel('Tension , V')

>> subplot(212),plot(y,z),grid
>> axis equal
>> xlabel('y'),ylabel('z')
```

Attention faites un **clf** (efface la figure) et **clear** (efface les données de la fenêtre de travail matlab) avant de passer a l'exercice suivant

Exemple 4:

```
>> w=logspace(0,3,1000);
>> s=j*w;
>> H=225./(s.*s+3*s+225);
>> AdB=20*log10(abs(H));
>> phase=angle(H)*(180/pi);
>> subplot(2,1,1),semilogx(w,AdB),grid
>> xlabel('w , rad/s'),ylabel('Amplitude , dB')
>> subplot(2,1,2),semilogx(w,phase),grid
>> xlabel('w , rad/s'),ylabel('Phase , degre')
```

- Diviser la fenêtre en deux parties (1 x 2)

| | | |
|---|---|---|
| subplot(1,2,1) | | subplot(1,2,2) |

- Diviser la fenêtre en quatre parties (2 x 2)

| | | |
|---|---|---|
| subplot(2,2,1) | | subplot(2,2,2) |
| subplot(2,2,3) | | subplot(2,2,4) |

etc...

AJOUT DU TEXTE AU GRAPHIQUE

| | |
|---|---|
| title('Titre du graphique') | Donner un titre au graphique |
| xlabel('Temps') | Étiquette de l'axe x |
| ylabel('Tension') | Étiquette de l'axe y |
| text(x,y,'Valeur absolue') | Ajouter du texte au graphique à la position x,y |
| gtext('Valeur absolue') | Ajouter du texte au graphique avec la souris |

MANIPULATION DE GRAPHIQUES

| | |
|---|---|
| axis([-1 5 -10 10]) | Choix des échelles x = (-1,5) et y = (-10,10) |

| | |
|---|---|
| hold on | Garder le graphique sur l'écran (pour tracer plusieurs courbes sur le même graphique) |

IMPRESSION ET ENREGISTREMENT DE GRAPHIQUES

| | |
|---|---|
| print -dps | Imprimer le graphique en PostScript |
| print -dpsc | Imprimer le graphique en PostScript Couleur |
| print -dps dessin.ps | Enregistrer le graphique en PostScript dans le fichier dessin.ps |

COMMUNICATION AVEC L'USAGER

On peut afficher un message, une valeur à l'écran avec l'instruction disp:

| | |
|---|---|
| disp('Ceci est un test') | Afficher "Ceci est un test" sur l'écran |

On peut entrer une valeur avec l'instruction input:

| | |
|---|---|
| x = input('Valeur de x = ') | Afficher sur l'écran "Valeur de x = " et attendre qu'un nombre soit tapé sur le clavier |

# 3. Programmation avec Matlab

## *3.1 FICHIERS M (ou mode executil)*

Les fichiers M sont des fichiers ASCII contenant des suites d'instructions MATLAB dont le nom a comme extension **.m**. Par exemple «test1.m». Dans la fenêtre Commande, si l'on tape >> test1, les instructions contenues dans le fichier test1.m seront exécutées par matlab une par une. On peut créer des fichiers M à l'aide de «Text Editor» ou dans la fenêtre matlab en choisissant dans le menu "file" : new M-file.

Exemple d'un fichier M:

```
% Ceci est un exemple de fichier M
% Les lignes "commentaires" commencent par "%"
for i=1:10
for j=1:4
x=0.005*i;
y=30+j;
z(i,j)=10*exp(-y*x)*cos(120*pi*x);
end
end
```

- **Conseil: à chaque fois que vous créez un programme, commencez par des commentaires avec:**

  % nom du fichier

% date de création

% utilité et notes sur le fichier

### *3.2 Opérateurs relationnels et logiques*

| **opérateurs relationnels** | | **opérateurs logiques** | |
|---|---|---|---|
| < | inférieur à | & | et |
| > | supérieur à | \| | ou |
| <= | inférieur ou égal à | ~ | non |
| >= | supérieur ou égal à | | |
| == | égal à | | |
| ~= | différent de | | |

### *3.3 BOUCLE FOR*

On peut créer une boucle en utilisant for ... end. On peut aussi réaliser des boucles FOR imbriquées.

Exemple 5:

Boucle FOR simple:

```
for i=1:100,
wt = 24*i*0.01;
x(i)=12.5*cos(wt+pi/6);
end
```

Deux boucles FOR:

```
for i=1:5
for j=1:20
amp=i*1.2;
wt=j*0.05;
v(i,j)=amp*sin(wt);
end
end
```

### *3.4 BOUCLE WHILE*

On peut créer une boucle en utilisant while ... end.

Exemple 6:

```
>>n=1;
>>while 2^n<100
   n=n+1;
  end
>>disp(n-1)
   6
```

### *3.5 INSTRUCTION IF ... ELSEIF ... ELSE*

L'instruction IF ... ELSEIF ... ELSE permet de choisir plusieurs options.

Exemple 7:

```
n=input('Donner un nombre positif ');
if rem(n,3)==0
disp('Ce nombre est divisible par 3')
elseif rem(n,5)==0
disp('Ce nombre est divisible par 5')
else
disp('Ce nombre n''est pas divisible par 3 ou par 5')
end
```