


# Aide-mémoire Scilab

Christophe Saint-Jean

version du 22 mars 2013

Cette aide-mémoire Scilab est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale 3.0 France. 

## Généralités

### Conventions de cette aide-mémoire

$c^*$  sont des variables ou des expressions scalaires.

$v^*$  sont des variables ou des expressions vectorielles.

$M^*$ ,  $N$  et  $P$  représentent des valeurs ou des expressions matricielles.

$str^*$  sont des valeurs ou des expressions de type chaînes de caractères.

$cond^*$  sont des valeurs ou des expressions booléennes.

$cmd^*$  sont des valeurs ou des expressions booléennes.

$var^*$  sont des variables ou des expressions sans type spécifique.

## Aide

**help()** démarre le système d'aide

**help** (*str\_cmd*) recherche l'aide sur la fonction *cmd*  
--> help('inv')

**apropos**(*str\_kwd*) recherche dans l'aide le mot-clé *kwd* et ordonne par pertinence les résultats  
--> apropos('inv')

## Environnement

**who** liste les variables connues dans l'environnement.

**whos** liste de manière détaillée les variables connues dans l'environnement.

**clear**(*str\_var1*, *str\_var2*, ..., *str\_varN*) supprime les variables *var1*, *var2*, ..., *varN* de l'environnement.  
--> clear('M', 'n', 'i')

**clear()** supprime toutes les variables.

**load**(*str\_fichier*) charge les variables sauvegardées dans *fichier*.

**save**(*str\_fichier*) sauvegarde l'ensemble des variables dans *fichier*.

*c\_id* = **diary**(*str\_fichier*) ouvre le journal *id* dans le fichier et sauvegarde les commandes entrées dans la console.

**diary**(*c\_id*, 'close') sauve le journal *id*.

```
--> id = diary('TP1.txt')
id =
1.
--> // des commandes
--> diary(id, 'close')
```

**ls()** liste des fichiers du répertoire de travail.

**pwd()** affiche le répertoire courant.

**cd**(*str\_rep*) modifie le répertoire courant (-> *rep*).

**exec** (*str\_script*) exécute le fichier de commandes *script*.  
--> exec('TP1.sce')

**scinotes()** lance l'éditeur de texte intégré de Scilab.

## Constantes

**%i**, **%pi**, **%e** : imaginaire  $i$ ,  $\pi$ ,  $e$

**%eps** est un petit nombre réel représentable en machine.

**%t**, **%f** sont les deux valeurs booléennes *vrai*, *faux*

**%inf** représente  $+\infty$

**%nan** indique qu'une valeur n'est pas déterminée.

```
--> %inf/%inf
ans =
Nan
```

**isnan**( $M$ ), **isinf**( $M$ ) teste chaque élément de  $M$ .

```
--> M = [%nan 1; -2 %nan]; isnan(M)
ans =
T F
F T
```

## Autres fonctions

**disp**(*var1*, *var2*, ..., *varN*) affiche chacun des paramètres dans l'ordre *varN*, ..., *var2*, *var1*.

```
--> a=3; b=5; disp('Un petit texte :', a, a+b)
8.
3.
Un petit texte :
```

$c$  = **input**(*str\_msg*) permet de récupérer une valeur saisie par l'utilisateur.

```
--> h=input('Donnez une valeur pour h : ');
Donnez une valeur pour h : 6
--> disp(h)
6
```

## Vecteurs et matrices

### Définition

**[ ]** est la matrice vide.

**[ $c_1$ ,  $c_2$ , ...,  $c_N$ ]** ou **[ $c_1$   $c_2$  ...  $c_N$ ]** est un vecteur ligne.

```
--> [1 2 3]
ans =
1. 2. 3.
```

**[ $c_1$ ;  $c_2$ ; ...;  $c_N$ ]** est un vecteur colonne.

```
--> [1; 2; 3]
ans =
1.
2.
3.
```

**[ $v_1$ ,  $v_2$ , ...,  $v_N$ ]** ou **[ $v_1$   $v_2$  ...  $v_N$ ]** est une matrice à  $n$  colonnes si les vecteurs colonnes  $v_1$ ,  $v_2$ , ...,  $v_N$  sont de même longueur.

```
--> [ 1;2;3] [4;5;6] ]
ans =
     1.     4.
     2.     5.
     3.     6.
```

**[ $v_1; v_2; \dots; v_n$ ]** est une matrice à  $n$  lignes si les vecteurs ligne  $v_1, v_2, \dots, v_n$  sont de même longueur.

```
--> [[1 2 3]; [4 5 6]]
ans =
     1.     2.     3.
     4.     5.     6.
```

**[ $M_1; M_2; \dots; M_n$ ]** est une matrice  $(l_1 + l_2 + \dots + l_n) \times c$  si les matrices  $M_1, M_2, \dots, M_n$  sont de taille  $l_i \times c$ .

```
--> M1 = [[1 2]; [3 4]], M2 = [5 6], [M1; M2]
M1 =
     1.     2.
     3.     4.
M2 =
     5.     6.
ans =
     1.     2.
     3.     4.
     5.     6.
```

**[ $M_1 M_2 \dots M_n$ ]** est une matrice  $l \times (c_1 + c_2 + \dots + c_n)$  si les matrices  $M_1, M_2, \dots, M_n$  sont de taille  $l \times c_i$ .

```
--> M1 = [[1 2]; [3 4]], M2 = [5;6], [M1 M2]
M1 =
     1.     2.
     3.     4.
M2 =
     5.
     6.
ans =
     1.     2.     5.
     3.     4.     6.
```

## Cas particuliers

**ones( $c\_lig, 1$ )** est le vecteur  $lig \times 1$  rempli de 1.

**ones( $1, c\_col$ )** est le vecteur  $1 \times col$  rempli de 1.

**ones( $c\_lig, c\_col$ )** est la matrice  $lig \times col$  remplie de 1.

**zeros( $c, 1$ ), zeros( $1, c$ ), zeros( $c\_lig, c\_col$ )** idem avec des 0.

**eye( $c, c$ )** est la matrice identité  $I_c$ .

**$c\_deb : c\_fin$**  produit la séquence des nombres vérifiant

$$[deb, fin] \cap [deb, deb + 1, deb + 2, \dots]$$

```
--> 1:4
ans =
     1.     2.     3.     4.
--> 1:-2
ans =
     []
--> 2.5:5
ans =
     2.5     3.5     4.5
```

**$c\_deb : c\_pas : c\_fin$**  produit la séquence des nombres vérifiant

$$[deb, fin] \cap [deb, deb + pas, deb + 2 * pas, \dots]$$

```
--> 1:1:4
ans =
     1.     2.     3.     4.
--> 4:-1:1
ans =
     4.     3.     2.     1.
--> 0:%pi/4:%pi
ans =
     0.     0.785     1.571     2.356     3.142
```

**linspace( $c\_deb, c\_fin, c\_n$ )** produit  $n$  nombres linéairement répartis dans l'intervalle  $[deb, fin]$ .

**logspace( $c\_deb, c\_fin, c\_n$ )** produit  $n$  nombres logarithmiquement répartis dans l'intervalle  $[deb, fin]$ .

```
--> linspace(2,4,3)
ans =
     2.     3.     4.
--> logspace(2,4,3)
ans =
    100.    1000.    10000.
--> 10^linspace(2,4,3)
ans =
    100.    1000.    10000.
```

**diag( $M$ )** est le vecteur des éléments diagonaux de  $M$ .

**diag( $v$ )** est une matrice carrée diagonale à partir de  $v$ .

**rand( $c\_lig, c\_col$ )** retourne une matrice de taille  $lig \times col$  dont les entrées sont choisies aléatoirement (uniformément dans  $[0, 1]$ ).

```
--> rand(2,3)
ans =
     0.728     0.547     0.740
     0.268     0.989     0.004
```

**rand( $M$ )** retourne une matrice de même taille que  $M$  dont les entrées sont choisies aléatoirement (uniformément dans  $[0, 1]$ ).

## Propriétés

**length( $v$ )** retourne le nombre d'éléments de  $v$ .

**length( $M$ )** retourne le nombre d'éléments de la matrice  $M$ .

**size( $M$ )** est le vecteur du nombre d'éléments par dimension de la matrice  $M$ .

**[ $c\_lig, c\_col$ ] = size( $M$ )** permet d'extraire directement le nombre de lignes  $lig$  et de colonnes  $col$  de la matrice  $M$ .

```
--> M = rand(3,2); length(M)
ans =
     6.
--> size(M)
ans =
     3.     2.
--> [l,c]=size(M)
c =
     2.
l =
     3.
```

## Indexation

$v(i)$ ,  $v(c_i)$ ,  $v(\$)$  désignent le premier, le  $i^{ieme}$  et le dernier élément du vecteur  $v$ .

$v(1:c_i)$  est le vecteur des  $i$  premiers éléments du vecteur  $v$ .

$v(\$-c_i:\$)$  est le vecteur des  $i+1$  derniers éléments du vecteur  $v$ .

$v(v\_ind)$  est un vecteur d'éléments extraits de  $v$  à partir du vecteur  $ind$  de leurs indices.

```
-->v=[2 4 -1 4 1];
-->[v(1) v(3) v($)]
ans =
    2.    -1.    1.
-->v(1:3)
ans =
    2.    4.    -1.
-->v($-2:$)
ans =
    -1.    4.    1.
-->v([1 4 3 2 2])
ans =
    2.    4.    -1.    4.    4.
```

$v(v\_TF)$  est un vecteur d'éléments extraits de  $v$  à partir du vecteur booléen  $TF$ .

```
-->v=[2 4 -1 4 1];
-->v > 1
ans =
    T    T    F    T    F
-->v(v>1)
ans =
    2.    4.    4.
```

$M(i,j)$  désigne l'élément de  $M$  à ligne  $i$ , colonne  $j$ .

$M(i,:)$  désigne la  $i^{ieme}$  ligne de la matrice  $M$ .

$M(:,j)$  désigne la  $j^{ieme}$  colonne de la matrice  $M$ .

```
-->M = [1 0 4;-2 1 2]
M =
    1.    0.    4.
   -2.    1.    2.
-->M(1,:)
ans =
```

```
    1.    0.    4.
-->M(:,2)
ans =
    0.
    1.
```

$M(i)$  est le vecteur colonne des éléments de  $M$  (indice linéaire).

$M(c_i)$  est le  $i^{ieme}$  élément en indice linéaire de  $M$ .

```
-->M = [1 0;-2 1]; M(:)
ans =
    1.
   -2.
    0.
    1.
-->M(3)
ans =
    0.
```

$M(v_I,v_J)$  est une sous-matrice de  $M$  à partir des vecteurs  $I$  et  $J$  de leurs indices ligne et colonne.

```
-->M = round(12*rand(3,3)-3)
M =
    0.    0.   -1.
   -1.   -2.    5.
    6.    5.    7.
-->M(1:2,1:2)
M =
    0.    0.
   -1.   -2.
-->M([1,3],[2 3 2])
ans =
    0.   -1.    0.
    5.    7.    5.
```

## Transformations

**flipdim**( $v,i$ ) inverse l'ordre des éléments du vecteur  $v$ .

**flipdim**( $M,c\_dim$ ) retourne la matrice  $M$  selon la dimension  $dim$ .

**matrix**( $v,c\_lig,c\_col$ ) retourne une matrice de taille  $lig \times col$  à partir des valeurs de  $v$ .

**matrix**( $M,c\_lig,c\_col$ ) ou **matrix**( $M,[c\_lig,c\_col]$ ) retourne une matrice  $lig \times col$  à partir des valeurs de  $M$ .

```
-->v=1:6;M = matrix(v,2,3)
ans =
    1.    3.    5.
    2.    4.    6.
-->matrix(M,3,2)
ans =
    1.    4.
    2.    5.
    3.    6.
```

**repmat**( $M,c\_lig,c\_col$ ) ou **repmat**( $M,[c\_lig,c\_col]$ ) retourne une matrice  $(lig * n) \times (col * m)$  par recopie de la matrice  $M$  de taille  $n \times m$ .

```
-->M = [1 2; 3 4]
M =
    1.    2.
    3.    4.
-->repmat(M,2,3)
ans =
    1.    2.    1.    2.    1.    2.
    3.    4.    3.    4.    3.    4.
    1.    2.    1.    2.    1.    2.
    3.    4.    3.    4.    3.    4.
```

## Autres fonctions utiles

**find**( $v$ ) retourne les indices de valeurs de  $v$  différentes de 0.

**find**( $M$ ) retourne les indices linéaires de valeurs de  $M$  différentes de 0.

$[v_I,v_J] = \mathbf{find}(M)$  retourne les indices ligne  $I$  et colonne  $J$  des valeurs de  $M$  différentes de 0.

```
-->v = [2, 0, 1, 0, -1, 3]
v =
    2.    0.    1.    0.   -1.    3
-->find(v)
ans =
    1.    3.    5.    6.
-->find(matrix(v,2,3))
ans =
```

```

1. 3. 5. 6.
-->[I,J] = find(matrix(v,2,3))
J =
1. 2. 3. 3.
I =
1. 1. 1. 2.

```

**find**(*v\_b*), **find**(*M\_b*), [*v\_I*,*v\_f*] = **find**(*M\_b*) idem pour les vecteurs et matrices booléennes avec les valeurs %t.

**and**(*v\_b*) retourne %t ssi tous les éléments de *b* sont %t (quantificateur universel  $\forall$ ).

**and**(*v\_b*) retourne %t ssi tous les éléments de *b* sont différents de 0.

**or**(*v\_b*) retourne %t ssi au moins un élément de *b* est %t (quantificateur existentiel  $\exists$ ).

**or**(*v\_b*) retourne %t ssi au moins un élément de *b* est différent de 0.

```

-->v = [2, 0, 1, 0, -1, 3]
v =
2. 0. 1. 0. -1. 3.
-->v>0
ans =
T F T F F T
-->[and(v>0) or(v<0)]
ans =
F T

```

## Math

### Calculs élémentaires

**sin,cos,tan,asin,acos,atan,atan2,log,log10,exp**

diverses fonctions mathématiques

```

-->format(6); v = linspace(0,0.5,6)*%pi
v =
0. 0.314 0.628 0.942 1.257 1.571
-->sin(v)
ans =
0. 0.309 0.588 0.809 0.951 1.

```

**min**(*v*) retourne le minimum d'un vecteur *v*.

[*c\_min*,*c\_i*] = **min**(*v*) idem avec *i* l'indice du minimum (le 1<sup>er</sup>)

```

-->v = [2 3 1 2 1]; [m,i] = min(v)
i =
3.
m =
1.

```

**min**(*M*) retourne le minimum d'une matrice *M*.

**min**(*M*, 'c') retourne le vecteur (colonne) des minima par ligne d'une matrice *M*.

[*v\_min*,*v\_I*] = **min**(*M*, 'c') idem avec *I*(j) l'indice du minimum pour la ligne *j*.

**min**(*M*, 'r') retourne le vecteur (ligne) des minima par colonne d'une matrice *M*.

[*v\_min*,*v\_I*] = **min**(*M*, 'r') idem avec *I*(j) l'indice du minimum pour la colonne *j*.

**max**(*v*), ..., **max**(*M*, 'r') idem pour le maximum.

**sum**(*v*) donne la somme du vecteur *v*.

**sum**(*M*) donne la somme des éléments de la matrice *M*.

**sum**(*M*, 'r') ou **sum**(*M*, i) retourne le vecteur ligne des sommes par colonne.

**sum**(*M*, 'c') ou **sum**(*M*, 2) retourne le vecteur colonne des sommes par ligne.

**prod**(*v*), ..., **prod**(*M*, 'c') idem pour le produit.

**cumsum**(*v*) retourne le vecteur *w* tel que

$$w(j) = \sum_{i=1}^j v(i)$$

```

-->v = 1:5; w = cumsum(v)
w =
1. 3. 6. 10. 15.

```

**cumprod**(*v*) retourne le vecteur *w* tel que

$$w(j) = \prod_{i=1}^j v(i)$$

```

-->v = 1:5; w = cumprod(v)
w =
1. 2. 6. 24. 120.

```

**diff**(*v*) retourne le vecteur des différences successives de *v*.

$$w(j) = v(j+1) - v(j)$$

```

-->v = 1:5; w = diff(v)
w =
1. 1. 1. 1.

```

**mean**(*v*) ou **mean**(*M*) retourne la moyenne des éléments d'un vecteur ou d'une matrice.

**median**(*v*) ou **median**(*M*) est l'élément médian d'un vecteur ou d'une matrice.

**unique**(*v*) extrait les éléments de *v* sans doublons (pas en place).

**union**(*v\_A*,*v\_B*), **intersect**(*v\_A*,*v\_B*)

**setdiff**(*v\_A*,*v\_B*), **setequal**(*v\_A*,*v\_B*) sont les opérations ensemblistes.

**gsort**(*v*) trie les éléments de *v* par ordre décroissant.

**gsort**(*v*, 'g', 'i') trie les éléments de *v* par ordre croissant.

## Calcul matriciel

*M'* est la matrice adjointe de *M*.

*M.*' est la transposée de *M*.

*M* + *N* et *M* - *N* sont la somme et la soustraction de *M* et *N*.

*P* = *M* \* *N* est le produit matriciel de *M* par *N*.

$$P_{i,j} = \sum_k M_{i,k} N_{k,j}$$

*P* = *M* .\* *N* est le produit élément par élément de *M* par *N*.

$$P_{i,j} = M_{i,j} N_{i,j}$$

*P* = *M* ./ *N* est la division élément par élément de *M* par *N*.

*P* = *M* ^ *c* est la puissance *c*<sup>ieme</sup> de *M*.

$$P = \underbrace{M * M * \dots * M}_c \text{ fois}$$

$P = M.^c$  est la puissance  $c^{ieme}$  élément par élément de  $M$ .

$$P_{i,j} = M_{i,j}^c$$

$P = c * M$  est le produit de chaque élément de  $M$  par  $c$ .

$$P_{i,j} = c M_{i,j}$$

$\sim M$  est la négation logique d'une matrice booléenne.

$M == N$  est le test d'égalité termes à termes.

$M <> N$  est le test d'inégalité termes à termes.

$M \& N$  correspond au "ET logique" entre deux matrices booléennes.

$M | N$  correspond au "OU logique" entre deux matrices booléennes.

$[v_{sol}, M_{Ker}] = \text{linsolve}(M, v)$  est la méthode générique de résolution d'un système linéaire d'équations

$$Mx + v = 0$$

Le vecteur  $sol$  est une solution particulière.

La matrice  $Ker$  est de base orthonormée du s.e.v des solutions.

**det**( $M$ ) est le déterminant de la matrice  $M$ .

**inv**( $M$ ) est l'inverse de la matrice inversible  $M$ .

**rank**( $M$ ) donne le rang de la matrice  $M$ .

**cond**( $M$ ) est le rapport entre la plus grande et la plus petite valeur propre de  $M$  (conditionnement).

**spec**( $M$ ) retourne l'ensemble des valeurs propres de  $M$ .

$[v_{valp}, v_{vecp}] = \text{spec}(M)$  retourne l'ensemble des valeurs propres et vecteurs propres de  $M$ .

```
-->A=diag([1,2,3]);X=rand(3,3);M=inv(X)*A*X;
-->spec(M)
valp =
    3.0000    1.0000    2.0000
-->[valpd,vecp] = spec(M)
vecp =
    3.0000    0.0000    0.0000
    0.0000    1.0000    0.0000
    0.0000    0.0000    2.0000
```

```
valpd =
    -0.694    0.550    0.151
    0.006    0.022    0.571
    0.720   -0.835   -0.807
```

**norm**( $M$ ) , **norm**( $M,1$ ), **norm**( $M,'fro'$ ) sont des exemples de norme matricielle :  $\|M\|_2, \|M\|_1, \|M\|_\infty$ .

$[M_U, M_D, M_V] = \text{svd}(M)$  retourne la décomposition en valeurs singulières de  $M$ .

$$M = U * D * {}^tV$$

$[M_U] = \text{chol}(M)$  calcule la décomposition de Cholesky pour une matrice symétrique définie positive  $M$ .

$$M = {}^tU * U$$

## Polynômes

$[p] = \text{poly}(M, 'x')$  définit  $p$  comme le polynôme caractéristique de  $M$ .

$$p(x) = \det(M - xI)$$

$[p] = \text{poly}(o, 'x')$  définit le polynôme  $p(x) = x$ .

$[p] = \text{poly}(v_{coeff}, 'x', 'coeff')$  définit le polynôme  $p$  à partir de ses coefficients.

$[p] = \text{poly}(v_{rac}, 'x', 'roots')$  définit le polynôme  $p$  à partir de ses racines.

$+, *, /$  opérations arithmétiques entre polynômes.

**horner**( $p, v$ ) évalue le polynôme pour toutes les valeurs de  $v$ .

**roots**( $p$ ) recherche les racines de  $p$ .

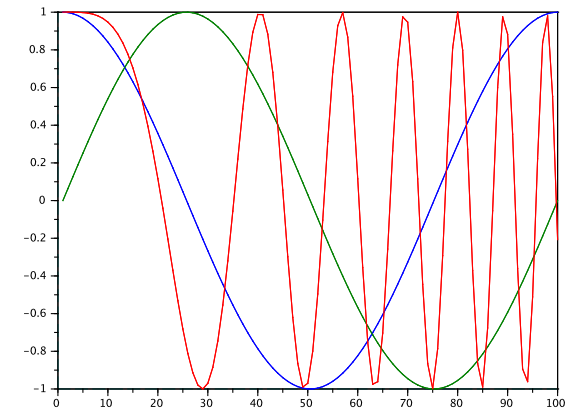
$[v_{pol}] = \text{polfact}(p)$  factorise le polynôme suivant ses racines.

## Graphiques

**plot**( $v_y$ ) trace la courbe reliant les points  $\{(t, y(t))\}$ .

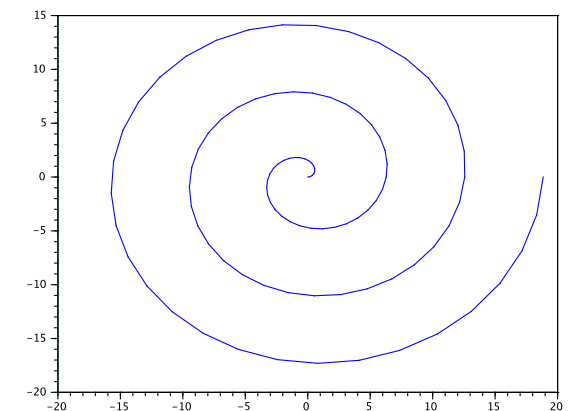
**plot**( $M$ ) applique la fonction de tracé précédente à chacune des colonnes de  $M$  (une couleur par colonne).

```
-->x = linspace(0,2*pi,100);
-->plot([cos(x.) sin(x.) cos(x.^2).']
```



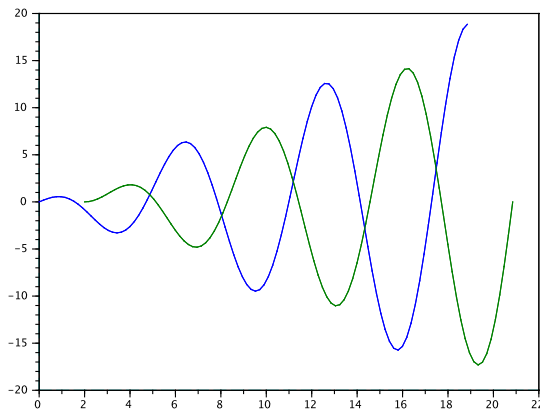
**plot**( $v_x, v_y$ ) trace la courbe reliant les points  $\{(x(t), y(t))\}$

```
-->t = linspace(0,6*pi,100);
-->plot(t.*cos(t).t.*sin(t))
```



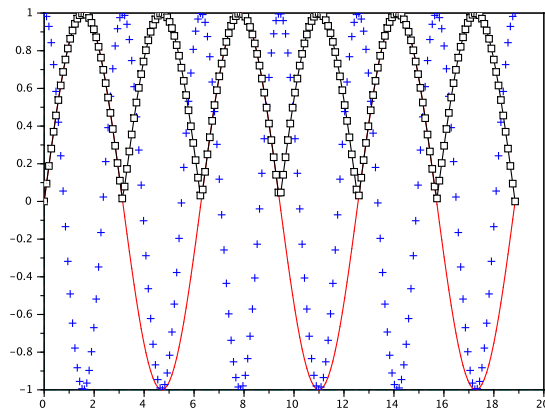
**plot**( $M_x, M_y$ ) applique la fonction de tracé précédente successivement à chacune des colonnes des deux matrices.

```
-->t = linspace(0,6*pi,100)';
-->plot([t t+2],[t.*cos(t) t.*sin(t)])
```



**plot**( $v_1x, v_1x, str_1, \dots, v_nx, v_nx, str_n$ ) applique la fonction de tracé **plot**( $v_x, v_y$ ) avec le style décrit dans la chaîne de caractères (voir

```
-->t=linspace(0,6*pi,200);
-->plot(t,sin(t),'r-',t+eps*cos(2*t),'b+', ...
-->      t,abs(sin(t)),'ks-')
```



**paramfplot2d**( $f, v_x, v_t$ ) produit une animation de la courbe paramétrée  $x \mapsto f(x; t)$ .

**Matplot**( $M$ ) affiche une matrice sous la forme d'une image.

**scf**( $c\_num$ ) positionne le prochain tracé sur la figure  $num$ .

**gcf**() retourne le numéro de la figure active.

**clf**() efface la figure courante.

**clf**( $c\_num$ ) efface la figure  $num$ .

**drawlater**() retarde l'affichage d'un graphique tout en permettant de le préparer en mémoire.

**drawnow**() affiche le graphique préparé en mémoire.

**xarrows**(...), **xrect**(...) sont des primitives de tracé d'objets graphiques.

**plot3d**(...) est la fonction générique de tracé 3d.

**genfact3d** permet de générer les facettes d'une surface pour un dessin en 3d.

**surf**( $M\_Z$ ) dessine une surface en 3d à partir des valeurs de la matrice  $Z$

**surf**( $M\_X, M\_Y, M\_Z$ ) idem en spécifiant en  $X$  et  $Y$  les coordonnées de la grille sur laquelle on connaît  $Z$ .

```
t=linspace(0,6*pi,100)';
plot3d(t.*cos(t),t.*sin(t),t)
t=[0:0.3:2*pi]';
z=sin(t)*cos(t');
[xx,yy,zz]=genfact3d(t,t,z);
plot3d(xx,yy,zz)
```

## Programmation

### Fonctions et paramètres

```
function [var_o1,...,var_om]=nom_fonct(var_i1,...,var_in)
    cmd_1
    ...
    cmd_n;
endfunction
```

où  $i_1, \dots, i_n$  et  $o_1, \dots, o_m$  désignent respectivement les paramètres d'entrée et les sorties de la fonction *nom\_fonc*. *Convention* : le code de la fonction *nom\_fonc* se trouve dans le fichier *nom\_fonc.sci*.

*Extrait du fichier "mafonction.sci"*

```
function [x2]=mafonction(x)
    x2 = x**2;
endfunction
```

*Puis dans la console*

```
-->mafonction(3)
ans =
    9.
```

**deff**(*str\_specif*, *str\_code*) permet de définir une fonction (courte) en ligne sans passer par un fichier .sci. La chaîne de caractères *str\_specif* contient la spécification de la fonction, la chaîne de caractères *str\_code* en contient le code.

```
-->deff('[x2]=mafonction(x)','x2 = x**2');
-->mafonction(3)
ans =
    9.
```

**deff**(*str\_specif*, *str\_code*, 'p') rajoute l'option de profilage à la fonction définie en ligne.

## Structures de contrôle

Branchement conditionnel **if**

```
if cond then
    cmd_1; ...; cmd_n;
elseif cond_i then
    cmd_i1; ...; cmd_in;
else
    cmd_e1; ...; cmd_en;
end
```

Répétitive **for**

```
for var=v do
    cmd_1; ...; cmd_n;
end
```

**for**  $var=M$  **do**

```
    cmd_1; ...; cmd_n;
end
```

Répétitive **while**

```
while cond do
    cmd_1; ...; cmd_n;
end

v = 1;
if v==0,
    for i=1:3,
        disp(i);
    end
else
    i=3;
    while i<=3,
        disp(i);
        i = i+1;
    end,
end
```

## Autres structures de données

**struct**(*str\_chp1*, *var1*, ..., *str\_chpn*, *varn*) retourne un enregistrement composé d'un ensemble de couples champ / valeur.

```
-->M = rand(26,2);M = M.' * M;
-->S = struct('Mat',M,'Det',det(M))
S =
    Mat: [2x2 constant]
    Det: 21.51
-->S.Mat
ans =
    7.54    7.885
```

```
7.885    11.1
-->S.det= 6;

list(var1, var2, ..., varn) est une liste composée des éléments passé en paramètres.
-->L = list(1:5,rand(2,2),[%f %f %t])
L =
    L(1)
    1.    2.    3.    4.    5.
    L(2)
    0.560    0.728
    0.125    0.268
    L(3)
    F F T
-->for l=L,
-->disp(sum(l))
-->end
    15.
    1.68
    1.
-->L(4) = 'chaîne';
```

## Flot d'exécution, Débogage

**break** interrompt la répétitive courante.

[*var\_o1*,...,*var\_om*] = **return**(*var1*,...,*varm*) provoque la sortie inconditionnelle de la fonction en cours d'interprétation.

**pause** interrompt l'exécution en l'attente de l'appui sur le clavier.

**xpause**(*c\_tps*) interrompt l'exécution pendant *tps* millisecondes.

**setbpt**(*str\_func*), **setbpt**(*str\_func,c\_lig*) ajoute un point d'arrêt à l'entrée de la fonction *nunc* ou à un certain numéro *lig* de ligne.

**delbpt**(), **delbpt**(*str\_func*), **setbpt**(*str\_func,c\_lig*) supprime toute ou partie des points d'arrêt existants.

**dispbpt**() liste les points d'arrêt existants.

**abort** interrompt l'interprétation courante.

**quit** provoque la sortie de Scilab.

**tic**() démarre le chronomètre.

**toc**() retourne le temps écoulé depuis le dernier **tic**.

**showprofile**(*var\_fun*) décore un code des informations de profilage après exécution de la fonction *fun*.

**plotprofile**(*var\_fun*) affiche un graphique de profilage après exécution de la fonction *fun*.

```
deff('benchchol(n)', ['for i=1:n'
                        '    M = rand(i,i)'
                        '    chol(M.''*M)'
                        'end'],'p')

-->benchchol(500);
-->showprofile(benchchol)
function []=fun(n) |1 |0 |0|
    for i = 1:n, |500|0 |0|
        M = rand(i, i) |500|0.96|4|
        chol((M.') * M) |500|9.24|6|
    end, |1 |0 |0|
endfunction |1 |0 |0|
```