

## Principales fonctions de Scilab

Eric Sonnendrücker, Université de Strasbourg

30 septembre 2009

Scilab est un logiciel libre de calcul numérique. Sources, binaires pour différentes plate-formes et documentation sont disponibles sur le site <http://www.scilab.org>

### Constantes prédéfinies

`%i` =  $\sqrt{-1}$ , `%pi` =  $\pi$ , `%e` =  $\exp(1)$ ,

`%eps` plus petit réel numérique  $\epsilon$  tel que  $1 + \epsilon \neq 1$  sur l'ordinateur,

`%t`, `%f` vrai et faux logique (p.ex, `2==2` donne `%t`),

`%inf` = infini,

`%nan` = NaN = not-a-number.

### Syntaxe

`variable = expression` : expression est évaluée, affectée à variable et affichée. Ex. `a=1+1`

`expression` : expression est évaluée, affectée à `ans` et affichée. Ex. `1+1`

`variable = expression; :` avec `:` à la fin expression et évalué à variable, mais pas affichée.

`// ceci est un commentaire` : commentaire

```
a = 1 + 2 + 3 + 4 ..
+ 5 + 6           : expression sur plusieurs lignes
```

### Opération arithmétiques sur les scalaires

Tous les scalaires numériques sont des réels double précision.

Addition, soustraction, multiplication : `a+b`, `a-b`, `a*b`

Division : `a/b` (résultat dans  $\mathbb{R}$ )

Exponentiation : `a**b` ou `a^b`

### Nombres complexes

`z=1+3*i` : définition d'un complexe

`real(z)`, `imag(z)`, `abs(z)` : partie réelle, partie imaginaire, module

### Fonctions

Fonctions courtes : utiliser le mot clef `deff`

`deff('f', 'disp(''hello'')')` : pas de variable d'entrée ou de sortie. `disp` affiche la valeur d'une variable

`deff('f(x)', 'disp(x**2+1)')` : 1 variable d'entrée

`deff('y=f(x)', 'y=x**2+1')` : 1 variable d'entrée, 1 variable de sortie

`deff('[s,d]=f(x,y)', 's=x+y,d=x-y')` : 2 variables d'entrée, 2 variables de sortie. Appel de cette fonction : `[ss,dd]=f(3,5)`

Fonctions longues : mots clefs `function`, `endfunction`

```
function [s,d,u]=f(x,y)
    s=x+y
    d=x-y
    u=sqrt(x**2+y**2)
endfunction
```

On peut utiliser un nombre quelconques de variables d'entrée et de sortie, même syntaxe que `deff` pour 0 ou 1 variable d'entrée ou de sortie.

### Boucles

Boucle de M à N (M,N entiers fixés) :

```
for i=M:N
    disp(i**2)
end
```

Boucle de M à N avec des pas de P (N,M,P entiers fixés) :

```
for i=M:P:N
    disp(i**2)
end
```

Boucle sur une liste de réels :

```
for i=[1, 3, 4.5, -7, 5, 2]
    disp(i**2)
end
```

Sur une ligne : remplacer les passage à la ligne par des virgules. Ex. `for i=1:6, disp(i**2),end`

### Instructions conditionnelles

*Opérateurs de comparaison :*

Egalité : `a==b`

Inégalité stricte : `a<b`, `a>b`

Inégalité large : `a<=b`, `a>=b`

Différence : `a~=b`

Logique a et b : `a & b`

Logique a ou b : `a | b`

Logique non a : `~a`

*Instructions conditionnelles :*

Sur une ligne :

`if x<0, y=-x, else, y=x, end` : avec affichage de y

`if x<0, y=-x; else, y=x; end` : sans affichage de y

Sur plusieurs ligne :

```
if x>=1
    instruction1;
elseif x<0
    instruction2;
else
    instruction3;
end
```

On omet ; après les instructions si on veut afficher le résultat.

### Vecteurs

Création de vecteurs :

`v = [1, 2, 3]` : vecteur ligne

`v = [1; 2; 3]` : vecteur colonne

`v = [v, 1]` : ajoute un élément à un vecteur existant

`v = linspace(a,b,N)` vecteur allant de *a* à *b* avec *N* termes uniformément répartis.

Manipulation de vecteurs :

`w = v(1:3)` : extrait un sous-vecteur (indices 1 à 3)

`v(1:3) = [0,0,0]` : modifie une partie d'un vecteur

`v(5:$)` : \$ représente le dernier indice

`v(5:length(v))` : identique à `v(5:$)`

### Création de Matrices

`m = [1, 2, 3; 4,5,6]` : matrice définie ligne par ligne

`m = ones(M,N)` : matrice de dimension (M,N) contenant uniquement des 1

`m = zeros(M,N)` : matrice de dimension (M,N) contenant uniquement des 0

`m = eye(M,N)` : matrice de dimension (M,N) contenant des 1 sur la diagonale principale et des 0 ailleurs

`m = diag(v)` : matrice contenant un vecteur v connu sur la diagonale et des 0 ailleurs. Ex. `diag(ones(4,1))`

`m = diag(v,i)` : matrice contenant un vecteur v connu sur la ième diagonale supérieure et des 0 ailleurs

`m = diag(v,-i)` : matrice contenant un vecteur v connu sur la ième diagonale inférieure et des 0 ailleurs

---

## Manipulation de matrices

`M = A(1:3,2:4)` : extrait une sous-matrice de la matrice A  
`M = A(:,2:4)` : extrait les colonnes 2 à 4  
`M = A(2:4,:)` : extrait les lignes 2 à 4  
`x = A(2,4)` : accède à un élément de A  
`A(1:3,3:4)=[1,2;3,4;5,6]` : modifie une sous-matrice de A. *Attention aux dimensions.*  
`m = m(:)` : transforme la matrice m en vecteur colonne en ordonnant colonne par colonne.

---

## Opérations sur les matrices

`A'` : adjointe de A (transposée du complexe conjugué)  
`A.'` : transposée de A. Pour les matrices réelles `A'=A.'`  
`A+B`, `A*B`, `A^2` : addition, multiplication, exponentiation matricielle.  
`A.*B`, `A./B`, `A.^2` : multiplication, division, exponentiation terme à terme de deux matrice de mêmes dimensions.  
`sin(A)`, `log(A)`, `exp(A)` : toutes les fonctions usuelles s'appliquent terme à terme aux vecteurs et aux matrices.  
`min(A)`, `max(A)` : minimum, maximum des éléments de la matrice  
`sum(A)`, `sum(A,1)`, `sum(A,2)` : somme des éléments de A, des lignes de A, des colonnes de A.  
`prod(A)`, `cumsum(A)`, `cumprod(A)` : produit, somme cumulative, produit cumulatif. S'applique aussi ligne par ligne ou colonne par colonne.  
`[nl,nc]=size(A)` : dimensions de A

---

## Algèbre linéaire

`x=A\b` :  $x$  est la solution du système linéaire  $Ax = b$ .  
`det(A)` : déterminant de A  
`inv(A)` : inverse de A  
`spec(A)` : valeurs propres de A. Permet aussi d'obtenir les vecteurs propres.

---

## Opérations booléennes sur une matrice

`find(A>0.5)` : tableau contenant les indices des éléments de A où la condition est vraie.  
`A(find(x==1))=3` : met les éléments de A correspondant aux indices où x vaut 1 à 3.

---

## Matrices creuses

Beaucoup de matrices en analyse numérique contiennent une grande proportion de zéros on gagne de l'espace mémoire et du temps de calcul en ne les stockant pas. Scilab propose dans ce cas un stockage contenant l'indice de ligne, l'indice de colonne et la valeur des termes non nuls.

`sparse(A)` : transforme une matrice pleine en matrice creuse  
`full(A)` : transforme une matrice creuse en matrice pleine  
`speye(nl,nc)` : matrice identité creuse à nl ligne et nc colonnes  
`spones(A)` : matrice creuse avec des 1 aux indices stockés de la matrice creuse A  
`spzeros(A)`, `spzeros(nl,nc)` : matrice creuse avec des 0  
`spget(A)` : retourne les indices stockés de la matrice creuse A

---

## Différentiation et intégration numérique

`z=inttrap(y)` : intégrale du vecteur y avec la formule des trapèzes. Par défaut avec un pas de un. Les abscisses peuvent être données.  
`y=diff(x)` : différences divisées de y

---

## Solveurs numériques

`fsolve` : calcule une racine d'un système d'équations non linéaires  
`ode` : résolution numérique d'équation différentielle

---

## Tracé de courbes

scilab trace des courbes définies par des valeurs numériques, i.e. des tableaux de points.  
`x=linspace(-%pi,%pi,100)` ; : tableau de 100 points entre  $-\pi$  et  $\pi$   
`plot(sin(x))` : trace sinus en les points de x en fonction de l'indice de x  
`plot(x,sin(x))` : trace sinus en fonction de x  
`plot(x,sin(x),'g-',x,cos(x),'r:')` : trace sin en trait continu vert et cos en trait pointillé rouge. On peut aussi superposer plus de courbes.  
*Si on n'efface pas la fenêtre (avec `clf` p.ex.) les graphes successifs sont automatiquement superposés.*  
*Couleurs :* 'b' : bleu, 'g' : vert, 'r' : rouge, 'c' : cyan, 'm' : magenta, 'y' : jaune, 'k' : noir, 'w' : blanc.

*Styles de ligne :* '-' : continu (défaut), '--' : tirets, '-.' : points-tirets, ':' : pointillée, '.' : un point à chaque valeur de x, 'o' : un cercle à chaque valeur de x, 'x' : une croix à chaque valeur de x... et d'autres !

`plot2d(x,[sin(x),cos(x)])` : similaire à `plot` avec syntaxe différente. Ici tracé de deux courbes aux points.  
`legends(['leg1','leg2'],[1,2])` : ajoute une légende  
`xbasc`, ou `clf` : efface la fenêtre graphique courante  
`figure(N)` : passe à la fenêtre N en la créant si nécessaire.  
`xtitle('titre du graphe')` : ajoute un titre sur la figure  
`subplot(325)` : crée des sous-figures (3 lignes et 2 colonnes) dans un graphique et rend active la 5ème.

---

## Graphiques 2D

`x=linspace(-3,3,100)`; `[X,Y]=meshgrid(x)`; : Création d'un maillage 2D dans le cas où les abscisses et les ordonnées sont identiques  
`x=linspace(0,4,100)`; `y=linspace(-3,3,100)`;  
`[X,Y]=meshgrid(x,y)`; : Création d'un maillage 2D pour des abscisses et ordonnées différentes  
`plot3d(x,y,exp(-X.^2-Y.^2))` : graphe d'une fonction de x,y en hauteur.  
`plot3d1(x,y,exp(-X.^2-Y.^2))` : graphe d'une fonction de x,y en hauteur et en couleur.  
`grayplot(x,y,exp(-X.^2-Y.^2))` : isovaleurs couleur d'une fonction de x,y  
`xset('colormap',cmap)` : permet de changer la carte de couleurs, `cmap` doit prendre les valeurs `jetcolormap(N)`, `hotcolormap(N)`, `graycolormap(N)` où N est le nombre de couleurs définies (p.ex 50).  
`contour(x,y,exp(-X.^2-Y.^2),10)` : graphe d'une fonction de x,y avec 10 lignes de niveaux.

---

## Animations

En fixant certains paramètres graphiques, on peut faire une animation qui ne clignote pas à l'aide d'une simple boucle. Exemple :

```
x=linspace(-%pi,%pi,100);
xset('pixmap',1)
for n=1:100
    clf
    plot(x,sin(x-0.1*n))
    xset('wshow')
end
```