Introduction

Référence de ce document : **http://www.juggling.ch/gisin/scilab/**

Ce qui suit **s'adresse aux personnes connaissant déjà** le logiciel **SciLab**.
Il existe de nombreuses initiations à SciLab, mais peu de résumés.

SciLab est un proche cousin de MatLab, "Mat" venant de "Matrices". C'est un environnement de **programmation scientifique**, basé sur le **traitement matriciel** et vectoriel.

Liens sur des **initiations et introductions à SciLab** :
https://www.scilab.org/fr/resources/documentation/tutorials
Divers liens du site officiel, pour des références.

https://www.scilab.org/fr/content/download/846/7841/file/livret_maths_2013.pdf
Bonne introduction du site Web officiel.

www.iecn.u-nancy.fr/~pincon/**scilab**/docA4.pdf
118 pages, c'est un livre, avec table des matières et exemples.

http://www.scilab.org/content/download/247/1702/file/introscilab.pdf
87 page, en anglais, une introduction officielle.

http://mabboux.pagesperso-orange.fr/informatique/**scilab**/tp_**scilab**.pdf
C'est une introduction avec un résumé, de 18 pages au total.

Pour des **livres** et des **liens**, c.f. mon site Web de référence : **http://www.juggling.ch/gisin/scilab/**

Liens sur des **résumés de SciLab** :
http://www.acsysteme.com/fr/memento-principales-fonctions-scilab
Résumé sur 12 pages, bien fait, dans un site Web, ce n'est pas un fichier .pdf.

http://www.proba.jussieu.fr/pageperso/roux/enseignements/1314/ifma/resume_scilab.pdf
Résumé sur 3 pages, utile.

Un site Web pour comparer plus de 560 langages de programmation :
http://rosettacode.org/wiki/Rosetta_Code
http://rosettacode.org/wiki/Scilab

Pour obtenir de l'aide, tapez help (F1) ou demo_gui depuis la console.
help gsort → donne de l'aide sur la fonction gsort

Résumé de commandes et fonctions SciLab,
rédigé par Bernard Gisin
Version, c.f. en-tête.

Raccourcis clavier de l'éditeur (Applications → SciNotes)

Ctrl+D	Commenter
Ctrl+Maj+D	Décommenter
Ctrl+I	Indenter proprement
F5	Enregistrer et exécuter le script de l'éditeur
Ctrl+E	Exécuter la sélection
Ctrl+Maj+O	Ouvrir la sélection dans l'éditeur
Ctrl+S	Sauver
Ctrl+G	Aller à la ligne ...
Ctrl+F	Rechercher dans un fichier
F1	Ouvrir l'aide en ligne / aide sur la sélection
help *linspace*	Donne de l'aide sur la fonction qui suit l'instruction "help"

Commandes de l'interpréteur SciLab

clc	Efface le contenu de la fenêtre de commandes
xdel(num)	Ferme la fenêtre graphique numéro num
clear	Efface toutes les variables.
pwd	Affiche le répertoire courant. pwd; rep=ans; pour que rep contienne le rép. courant
exec("fichier")	Exécution d'un programme SciLab
funcprot(0)	N'affiche pas de message lors de redéfinition de fonctions
help plot	pour de l'aide sur la fonction plot

Commentaires

// Ceci est un commentaire.
Sélectionnez plusieurs lignes et tapez ctrl+D pour les commenter.
shift+ctrl+D enlèvera les commentaires des lignes sélectionnées.
; Le ; en fin d'instruction est un inhibiteur d'affichage.

.. à la fin d'une ligne permet de la continuer sur la ligne suivante.

Variables et matrices

Les variables n'ont pas besoin d'être déclarée initialement.
Si on veut qu'une variable soit globale, il faut la déclarer comme suite dans les fonctions et dans la "base".
global variable_globale;
Les matrices sont stockées colonnes par colonne, a11 a21 .. aN1 a12 a22 .. aN2 a13 a23 .. aN3 .. aNM

string = "chaîne de caractères"

a = 123.67	"=" est l'opérateur d'assignation
[12, 34, 3.2]	Définition d'un vecteur ligne
[1 ; 2 ; 3]	Définition d'un vecteur colonne
[11, 12, 13; 21, 22, 23]	Définition d'une matrice. Écrite ligne par ligne
[]	La matrice vide, de dimension 0 x 0
[1:10]	Vecteur ligne des entiers de 1 à 10
1:0.1:2	Vecteur ligne de 1 à 2 par pas de 0.1
zeros(3,4)	Matrice nulle de 3 lignes et 4 colonnes
ones(3,4)	Matrice de "1" de trois lignes et quatre colonnes
eye(4,4)	Matrice identité de taille quatre
diag([1 2 3 4])	Matrice diagonale de taille quatre, avec les élements 1, 2, 3 et 4 sur la diagonale
diag([1 2 3 4],2)	Matrice de taille 4+2, avec les éléments 1, 2, 3, 4 sur la 2e sur-diagonale, sinon nulle
diag([1 2 3 4],-1)	Matrice de taille 4+1, avec les éléments 1, 2, 3, 4 sur la 1è sous-diagonale, sinon nulle
linspace(1,3,10)	Vecteur ligne de 10 composantes allant de 1 à 3
logspace(1,3,7)	Vecteur ligne de 7 composantes allant de 10^1 à 10^3 espacés logarithmique
real(z)	Partie réelle du nombre complexe z
imag(z)	Partie imaginaire du nombre complexe z
abs(z)	Module du nombre complexe z
conj(z)	Conjugué de z. conj(a + b*%i) == a - b*%i
complex(a, b)	Nombre complexe a + b*%i

Constantes spéciales

ans	Dernier résultat d'un calcul
%pi, %e	Constante π et e
%i	Le nombre imaginaire i qui satisfait donc $i^2 = -1$
%nan	Not a Number, résultat d'un calcul indéfini
%inf	Infini
%eps	Précision de la machine. 1 + %eps == 1 est faux. 2 + %eps == 2 est vrai
%t, %f	Vrai, Faux, résultat d'une opération logique

Opérateurs arithmétiques et matriciels + fonctions

+ - * /	Opérateurs standards Partie entière de la division, c.f. page 5
modulo(a, b)	Donne le reste de la division de a par b
a^b	a puissance b
A * B	Multiplication matricielle
A .* B	Multiplication terme par terme des deux matrices, de même dimension
A .^ B	Mise à la puissance terme par terme des deux matrices, de même dimension
A^2	Carré de la matrice A
A.^2	Matrice dont les coefficients sont les carrés de ceux de A
A^(-1)	Matrice inverse de A (on la calcule rarement)
A'	Matrice transposée de A, les lignes deviennent les colonnes
A\b	A est une matrice, b un vecteur, donne la solution de A * x = b
exp(A)	Matrice dont les coefficients sont les exponentielles de ceux de A
f(A)	Matrice dont les coefficients sont les images par f des coefficients de A
expm(A)	Exponentielle matricielle de A
A(2, 3)	Coefficient de la 2^e ligne et 3^e colonne de la matrice A
A(2,$)	Dernier coefficient de la 2^e ligne de A
A($-1, 3)	Avant dernier coefficient de la 3^e colonne de A
A(:, 4)	Vecteur colonne formé de la 4^e colonne de la matrice A
A(4, :)	Vecteur ligne formé de la 4^e ligne de la matrice A
[A, B]	Matrice formée de A suivit de B, de nombre de colonnes=celui de A + celui de B
[A; B]	Matrice formée de A suivit de B, de nombre de lignes = celui de A + celui de B
A([1,3],[2,4])	Matrice 2 x 2 formée des intersections des lignes 1 et 3 et des colonnes 2 et 4.
zeros(3,4)	Matrice nulle de 3 lignes et 4 colonnes
ones(3,4)	Matrice de ”1” de trois lignes et quatre colonnes
eye(4,4)	Matrice identité de taille quatre
diag([1 2 3 4])	Matrice diagonale de taille quatre, avec les élements 1, 2, 3 et 4 sur la diagonale
diag([1 2 3 4], 2)	Matrice de taille 4+2, avec les éléments 1, 2, 3, 4 sur la 2^e sur-diagonale, sinon nulle
diag([1 2 3 4], -1)	Matrice de taille 4+1, avec les éléments 1, 2, 3, 4 sur la 1^è sous-diagonale, sinon nulle
linspace(1,3,10)	Vecteur ligne de 10 composantes allant de 1 à 3
logspace(1,3,7)	Vecteur ligne de 7 composantes allant de 10^1 à 10^3 espacés logarithmique
rand()	Retourne un nombre aléatoire entre 0 et 1
rand(3, 5)	Matrice de dimension 3 x 5 de nombres aléatoires entre 0 et 1
rand(mat)	Matrice de même dimension que mat, de nombres aléatoires entre 0 et 1
rand("uniform")	Les prochains nombres seront tirés uniformément
rand("normal")	Les prochains nombres seront tirés selon la loi Normale(0, 1)
rand("info")	Retourne le mode "uniform" ou "normal" des nombres tirés
rand("seed", 77)	Initialise à 77 la suite de nombre aléatoire. 77 = seed
rand("seed")	Retourne le "seed" qui a initialisé la suite aléatoire

x ; V ; A peuvent être un nombre, un vecteur ou une matrice !

Quelques fonctions standards

round(4.7)	→ 5. Arrondi à l'entier le plus proche. round(–4.7) = – 5
floor(4.7)	→ 4. Plus grand entier plus petit ou égale à 4.7. floor(–4.7) = – 5
ceil(4.7)	→ 5. Plus petit entier plus grand ou égale à 4.7. ceil(–4.7) = – 4
int(4.7)	Arrondi au prochain entier vers 0, à éviter
sqrt(x)	Racine carrée de x
abs(x)	Valeur absolue de x. Module de x, si x est un nombre complexe.
sign(x)	1 si x > 0, –1 si x < 0, 0 si x = 0
log(x)	Logarithme naturel (Néperien) de x
log10(x)	Logarithme en base 10 de x
exp(x)	Exponentiel de x, = e^x
10^x	10^x
modulo(n, m)	Reste de la division de n par m.
size(A) size(A, 1)	Vecteur ligne indiquant les dimensions de A. sA=size(A); sA(1) = nbr de lignes Nombre de lignes de A ; nombre de colonnes de A
ndims(A)	Nombre de dimensions d'un tableau A. =1 pour un vecteur, =2 pour une matrice
length(V)	Nombre d'éléments d'un vecteur ou d'une matrice
gsort(V, "g", "i")	Trie les valeurs de V par ordre croissant "i", c.f. "help gsort" pour les paramètres
sum(V)	Somme des nombres de V
cumsum(V)	Vecteur de somme cumulative des sommes de V
mean(V)	Moyenne des valeurs = sum(V) / length(V)
deff('z=moy(x,y)', 'z=(x+y)/2')	Permet de définir une fonction à partir de sa description dans deux chaînes de caractères. Utile en cours d'exécution d'un programme.
find(V > 0.4)	Retourne un vecteur d'indices de V satisfaisant la condition V > 0.4
color('red')	Retourne la couleur rouge. red, green, blue, cyan, magenta, yellow,
color(r,g,b)	Retourne une couleur, r, g, b entier entre 0 et 255.
rgb2name(r,g,b)	Retourne le nom de la couleur définie par r, g, b entre 0 et 255
V>1 sum(V>1)	Retourne un vecteur de boolean indiquant si V(i) > 1 ou non. Somme des composantes de V qui sont >1.
feval(matrice, fct)	Évalue la fonction ftc sur chaque composante de la matrice.

Fonctions trigonométriques

sin(x) cos(x)	Le sinus de x Le cosinus de x
tan(x) cotg(x)	La tangente de x La cotangente de x
asin(theta)	L'arcsinus de theta
acos(theta)	L'arccosinus de theta
atan(theta)	L'arctangente de theta
sinh, cosh, ...	L'équivalent pour les fonctions hyperboliques

Opérateurs logiques

Le résultat d'une opération logique est une soit %T (vrai) soit %F (faux).
Tout nombre non nul est considéré comme vrai, 0 est considéré comme faux.
0 & 7 → %F 3 & 7 → %T

a == b	Test si a égale b. Retourne %T si vrai, sinon retourne %F
a <> b	Test si a est différent de b. Retourne %F si a == b, sinon retourne %T
a ~= b	Comme a <> b
~	Opérateur de négation. ~(a == b) est identique à a ~= b
a > b a >= b	Retourne %T si a > b Retourne %T si a >= b
a < b a <= b	Retourne %T si a < b Retourne %T si a <= b
t1 & t2	ET logique. Vrai si et seulement si t1 et t2 sont vrais
t1 | t2	OU logique. Faux si et seulement si t1 et t2 sont faux
isempty(A)	Vrai si le vecteur A, la matrice A ou la liste A est vide
isnan(x)	Vrai si x est un résultat donnant Nan
isinf(x)	Vrai si x est un résultat donnant Inf
isnum(str)	Vrai si un string représente un nombre
exists("*fichier*")	Test si le *fichier* existe. Utile pour exec("*fichier*") s'il existe.

Strings, chaîne de caractères

strS = "un string" strS = 'un string'	Assignation d'une chaîne de caractères Deux écritures possibles
length(strS)	Longueur d'un string
str1 + str2	String formé de str1 suivit de str2. Concaténation de deux strings
strcat(["ab", "cd"])	Concaténation des strings d'un vecteur ou d'une matrice
string(x)	Converti les éléments de x en string
strplit("abcd")	Retourne un vecteur contenant les 4 caractères a, b, c et d
strsubst("aba", "a", "cd")	Retourne "cdbcd", donc substitue "a" par "cd"
ascii("abcd")	Retourne un vecteur contenant les codes ascii (ou utf-8) des caractères.
isnum("44.5")	Vrai si le string représente un nombre.
evel("44.5")	Convertit le string "44.5" en nombre.
eval("sin(%pi/2)")	Évalue le string, comme si c'était une commande scilab. c.f. aussi evstr(Z)

Syntaxe de programmation, test if et boucles

..	.. à la fin d'une ligne indique que la suite continue sur la ligne suivante.
function y = f(x) y = x.*x; endfunction	Définition d'une fonction d'une variable. x et y peuvent être des matrices.

function [y, z] = f(x1, x2)
  y = cos(x1);
  z = sin(x2);
endfunction
	Définition d'une fonction de plusieurs variables.
	Elles peuvent être des matrices.
	Le résultat est un vecteur lignes, les composantes pouvant être des matrices.

if (a == b) then
  // cas où a et b sont égaux
elseif (a > b) // *facultatif*
  // cas où a > b
else // *facultatif*
  // autres cas
end
	Test if.
	elseif est facultatif, il peut y en avoir plusieurs
	else est facultatif

switch *expression*
  case *valeur_1*
  // instructions
  case (*valeur_2*, *valeur3*) // *facultatif*
  // instructions
  otherwise
  // instructions
end
	L'expression retourne un nombre.
	On teste différentes valeurs possibles.

a = 0;
for i=1:10
  a = a + i^2;
end;
	Boucle for
	la variable i varie de 1 à 10 par pas de 1.

a = 0;
for i=1:0.1:3
  a = a + i^2;
end;
	Boucle for
	la variable i varie de 1 à 3 par pas de 0.1.

for i=V; disp(i); end;	Boucle avec i prenant toutes les valeurs du vecteur ligne V

a = 0; i = 1;
while (a < 5)
  a = a + 1 / i;
  i = i + 1;
end;
	Boucle "tant que"

break	Dans une boucle, termine la boucle (sort de la boucle)
continue	Dans une boucle, interrompt l'exécution d'une itération et passe à la suivante
return	Sort d'une fonction et retourne à l'appelant

for i=1:10
  disp(i);
  if (i == 5)
  mprintf("i=%2d", i)
  pause;
  end;
  end;
	pause; → Met le programme en pause.
	Très utile pour le **debugging**
	resume continue l'exécution
	abort arrête l'exécution

halt()	Arrête l'exécution du programme

Affichage

disp(A)	Affiche la matrice A
format(20)	Indique que les nombres affichés avec disp(...) prennent 20 caractères.
lines(nombre)	Indique combien de lignes sont affichées avant de faire une pause
funcprot(0)	N'affiche pas de message lors de redéfinition de fonctions

mprintf("*format", données*)
mprintf("i=%6d v=%12.8f", i, v);
mprintf('a string: %s\n', 'Scilab');
mprintf('an integer: %d\n', 10);
mprintf('an integer: %4d\n', 10);
mprintf('a left justified integer: %-4d\n', 10);
mprintf('an integer converted to float: %#fd\n',10);
mprintf('an integer with a sign: %+4d\n', 10);
mprintf('an integer with a sign: %+4d\n', -10);
mprintf('an integer padded with zeros: %04d\n', 10);
mprintf('an unsigned integer: %u\n', 10);
mprintf('an unsigned integer: %4u\n', -10);
mprintf('an integer converted to hexadecimal: %x\n', 10);
mprintf('a float: %d\n', %pi);
mprintf('a float: %3.2d\n', %pi);
mprintf('a float (exponential form): %3.2e\n', %pi);
mprintf('a float (exponential form): %3.2g\n', %pi);
mprintf('a character: %c\n', 'a');
mprintf('a character: %c\n', 'aaa');
	Affichage formaté de variables.
	\n indique un retour à la ligne
	\t indique un tabulateur
	
	%6d pour des nombres entiers
	%12.8f pour des nombres à virgule
	%s pour les strings
	%e pour la notation exponentielle
	%E pour la notation ingénieur

Gestions du temps et des dates

tic()	Démarre un chronomètre
toc()	Retourne le temps en secondes depuis le départ du chronomètre
getdate()	Retourne un vecteur contenant la date et l'heure, c.f. help getdate
etime(d2, d1)	Retourne le temps entre deux dates.
xpause(*temps en microsecondes*)	Attente du temps demandé.

Graphisme

clf() clf(*num*)	Efface la fenêtre graphique courante Efface la fenêtre graphique numéro *num*
scf(3)	"Set Current graphic Figure" Crée une nouvelle fenêtre graphique, n° 3. La suite est dessinée dans cette fenêtre graphique.
x=0:0.1:7; y=sin(x); **plot**(x, y);	Trace le graphique de y en fonction de x. c.f. plus d'options ci-dessous.
plot2d(..) bar(V) pie(V) histplot(...) champ(...) contour(...) surf(...)	Alternative à plot, avec plus de paramètres. C.f. help plot2d Trace un diagramme en barre. C.f. help bar Camenbert. C.f. help pie Histogramme. C.f. help histplot Champ de vecteurs. C.f. help champ Lignes de niveau. C.f. help contour Trace une surface en 3D
mygcf = gcf()	"Get Curent Figure" Retourne une structure sur la figure courante
myscf = scf()	"Set Curent Figure" Crée une nouvelle fenêtre graphique et retourne dans myscf une structure de données sur cette fenêtre graphique.
mygdf = gdf()	"Get Default Figure" Retourne une structure sur la figure par défaut
mygdf.children	Autre structure des axes de la figure
mygda = gda() mygda.thickness = 4;	"Get Default Axes" Retourne une structure de données concernant les axes du graphique imposera que la courbe soit plus épaisse. Taper gda() pour une liste des paramètres.
sdf()	"Set Default Figure"
gca()	"Get Default Axes"

plot(x, y, 'Color', 'red', 'LineStyle', '-.', 'Marker', 'diam'); c.f. GlobalProperty
LineStyle : '-' trait plein ; '--' traitillé ; ':' pointillé ; '-.' trait - point ; 'none' pas de ligne.
'Marker', '.' ; 'o' ; 'x' ; '+' ; '*' ; 's' ; 'd' ; 'v' ; '^' ; '<' ; '>' ; 'p'
'MarkSize', *un nombre qui donne la taille de la marque en points*
'Color', 'r' ; '' ; 'g' ; 'b' ; 'y' ; 'm' ; 'c' ; 'w' ; 'k' (black) ;
'Color', [r, g, b] où r, g, b sont entre 0 et 1 pour red green blue
Pour des **annotations**, c.f.
legend ; xlabel ; ylabel ; zlabel ; title
Pour des **opérations diverses**, c.f.
colorbar → affiche l'échelle des couleurs
zoom_rect → zoom sur un rectangle
un_zoom → restaure le zoom par défaut
xstring → dessine une chaîne de caractères
xinfo → affiche une chaîne dans la barre d'état

Voir : help Graphics Entities (help graphics puis cliquer sur "graphics entities")
fplot3d1(x, y, f) → Trace une surface définie par une fonction.
deff('z=myf(x,y)', 'z=x^2 - y^2'); x=-3:0.2:3; y=-2:0.2:2; fplot3d1(x, y, myf);

Animation

paramfplot2d(f, x, theta, flag, rect)	Animation en 2D, c.f. help paramfplot2d
comet(x, y, p, "colors", color(r, g, b))	Animation en 2D d'un ou plusieurs points, c.f. help comet r, g, b entiers entre 0 et 255.
xclick()	Lit les coordonnées de la souris lors d'un clique dans la fenêtre graphique. Permet aussi de lire une touche du clavier. c.f. help xclick
xgetmouse() a = []; while (a(1) > 0.1); a=xgetmouse(); disp(a); end; a(1)=1;	Retourne les coordonnées de la souris
drawlater()	Indique de ne pas dessiner les graphiques
drawnow()	Indique de dessiner ce qui était en attente du 'drawlater()'

Gestion de fichiers

exec("fichier")	Exécution d'un programme SciLab

Explications concernant les fenêtres graphique

Se référer au programme d'exemples : **Plot_param.sce**
Se référer à : **help Graphics Entities**
Pour changer les couleurs d'éléments d'un graphique, on utilise une table de couleurs qui se nomme '**color_map**'.
Une fenêtre graphique s'appelle une **figure**.
gcf() retourne une structure sur la fenêtre graphique par active (Get Current Figure)
scf(n°) crée si nécessaire et définit la fenêtre graphique active,

```
win5=scf(5);  win5b=gcf();  win5==win5b  retourne %T.
```

www=winsid() => retourne un vecteur des fenêtres ouverts.

```
www=winsid(); for ii=www; xdel(ii); end;   // ferme toutes les fenêtres.
```

Show_window([n°]); // est censé mettre au premier plan une fenêtre. Ne fonctionne pas !

```
mygda=gda();  mygda.auto_clear='on';  // efface automatiquement le graphique
xdel(num);
```

Ferme la fenêtre graphique numéro num

J'ai passé plus de trois heures à chercher comment récupérer le chemin complet du fichier en cours d'exécution.
get_absolute_file_path(…) ; est une piste, mais il faut lui transmettre le nom du fichier en cours d'exécution.
sciargs(); // Est la solution en théorie, mais ne fonctionne pas sous linux.
dispfiles(); // affiche ce qu'on veut. On peut l'utiliser, mais c'est compliqué.

```
//disp(get_absolute_file_path("test01.sce")); // Retourne le répertoire du fichier si c'est celui en cours d'exécution.
//disp(getversion());
//disp(getos());
//disp(SCI);
//disp(SCIHOME);
//disp(TMPDIR);
//disp(get_absolute_file_path("./"));
//disp(getdebuginfo());

//args =sciargs(); // En théorie, retourne le nom du fichier exécuté, ne marche pas en pratique sous linux !?!
//disp(size(args)); disp(args(1));

//dispfiles(); // Afficher les informations sur les fichiers ouvert.
        // 1 = le fichier en cours d'exécution.  6 = la sortie standard.  5=l'enstée, 0=sortie d'erreurs

// Permet de lire le fichier en cours d'exécution.  1 = le descripteur du fichier en cours d'exécution.
//disp(mgetstr(100,1));  // Lit 100 caractères du fichier en cours d'exécution.
//disp(mgetl(1, 9)); // lit 9 lignes du fichier en cours d'exécution.

--------------------------------------------------------------------------
// Enregistre tout ce qui sort dans la console, dans un fichier.
[id, filename] = diary("text0.txt");
dispfiles(); // la 4ème ligne contiendra le chemin complet du fichier en cours d'exécution
diary([], "close"); // Fin d'enregistrement dans le fichier.

id2 = mopen("text0.txt", "rt"); // Récupère les informations dans le fichier, qui contient le chemin complet du fichier en cours d'exécution
mgetl(id2, 3); // Saute les 3 premières lignes.
strPath = mgetl(id2, 1); // Lit la 4ème ligne, qui contient le chemin complet du fichier en cours d'exécution.
mclose(id2);

nnn = regexp(strPath, "/\//"); // Cherche les occurences de "/" dans la chaîne de caractères. La dernière occurence sera utilisée
strPath = part(strPath, [2:nnn($)]); // Saute le 1er caractères qui est un "|" et va jusqu'au dernier "\".
disp(strPath); // Contient le nom du répertoire contenant le fichier en cours d'exécution !
```