scilab enterprises

Scilab pour les vrais débutants

Ce document a été co-écrit par Scilab Enterprises et Christine Gomez, professeur de mathématiques au lycée Descartes à Antony, Hauts-de-Seine.

Table des matières

Introduction

À propos de ce document

L'objectif de ce document est de vous guider pas à pas dans la découverte des différentes fonctionnalités de base du logiciel Scilab pour un utilisateur n'ayant jamais utilisé un logiciel de calcul. Cette présentation se limite volontairement à l'essentiel pour permettre une prise en main facilitée de Scilab.

Les calculs, graphiques et illustrations sont réalisés avec Scilab 5.4.0. Vous pouvez donc reproduire toutes les commandes présentées à partir de cette version.

Installation de Scilab

Scilab est un logiciel de calcul numérique que chacun peut télécharger gratuitement. Disponible sous Windows, Linux et Mac OS X, Scilab est téléchargeable à l'adresse suivante : http://www.scilab.org/

Vous pouvez être averti des sorties de nouvelles versions du logiciel Scilab en vous inscrivant sur notre canal de notification à l'adresse suivante : http://lists.scilab.org/mailman/listinfo/release

Liste de diffusion et d'information

Pour faciliter l'échange entre les utilisateurs de Scilab, des listes de diffusion leur sont dédiées (liste en français, liste pour le monde de l'éducation, liste internationale en anglais). Le principe est simple : les personnes inscrites peuvent communiquer les unes avec les autres par courrier électronique (questions, réponses, partage de documents, retour d'expériences...).

Pour consulter les listes disponibles et s'inscrire, rendez-vous à l'adresse suivante : http://www.scilab.org/communities/user_zone/mailing_list

Ressources complémentaires

Si vous disposez d'une connexion à Internet, vous pouvez accéder au site Web de Scilab sur lequel vous trouverez une rubrique consacrée à l'utilisation de Scilab (http://www.scilab.org/support/documentation), avec des liens et des documents utiles pouvant être téléchargés et imprimés librement.

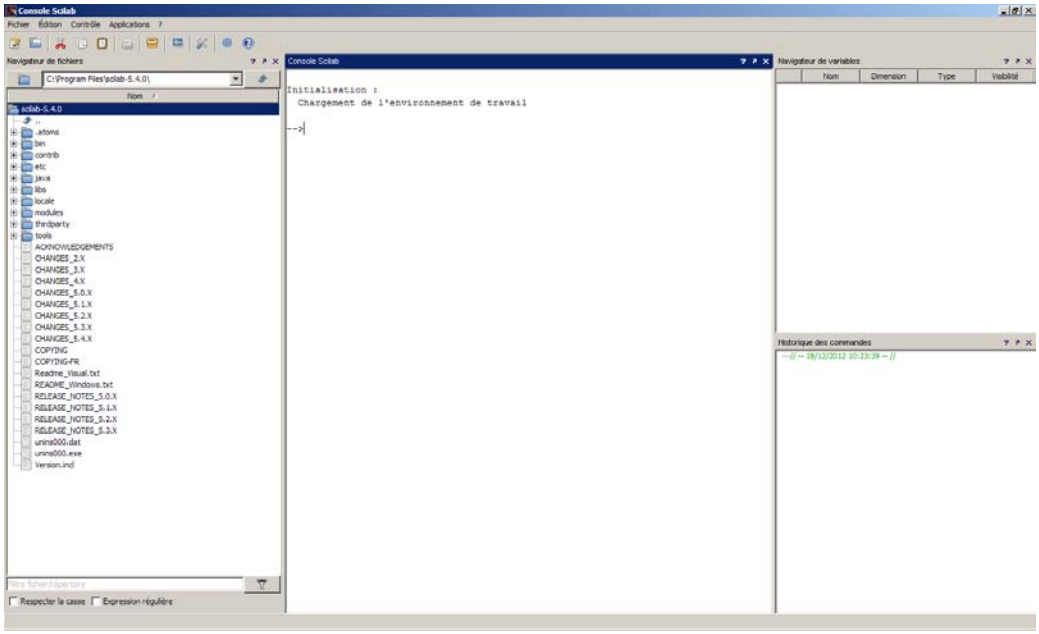
Chapitre 1 – Se familiariser à Scilab

L'espace de travail utile dans Scilab est constitué de plusieurs fenêtres :

- La console pour faire des calculs,
- L'éditeur pour écrire des programmes,
- Les fenêtres graphiques pour afficher des graphiques,
- L'aide.

L'environnement général et la console

Après avoir double-cliqué sur l'icône de Scilab pour lancer le logiciel, l'environnement par défaut de Scilab présente les fenêtres suivantes « dockées » - console, navigateurs de fichiers et de variables, historiques des commandes (voir « Gérer les fenêtres et personnaliser son espace de travail », page 11) :



Dans la console, après l'invite de commande « **-->** », il suffit de saisir une commande et d'appuyer sur la touche Entrée (Windows et Linux) ou Retour (Mac OS X) du clavier pour obtenir le résultat correspondant.

```
--> 57/4
 ans  =
    14.25
--> (2+9)^5
 ans  =
    161051.
```

À noter
Devant le résultat, **ans** s'affiche pour « answer » (« réponse » en anglais).

Il est possible de revenir en arrière à tout moment, avec les flèches du clavier ← ↑ → ↓ ou avec la souris, les touches gauche et droite permettant de modifier les instructions, et les touches haut et bas donnant la possibilité de revenir sur une commande précédemment exécutée.

Calculs numériques simples

Tous les calculs effectués par Scilab sont numériques. Scilab calcule avec des matrices (voir le chapitre 2, page 23).

Les opérations se notent « **+** » pour l'addition, « **–** » pour la soustraction, « ***** » pour la multiplication, « **/** » pour la division, « **^** » pour les exposants. La virgule des nombres décimaux est notée avec un point. Par exemple :

```
-->2+3.4
 ans  =
    5.4
```

Il est nécessaire de bien respecter la casse (majuscules et minuscules) pour que les calculs s'effectuent correctement. Par exemple, avec la commande **sqrt** qui permet de calculer la racine carrée :

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

alors que :

```
-->SQRT(9)
         !--error 4
Variable non définie: SQRT
```

Des nombres particuliers

%e et **%pi** représentent respectivement e et π :

```
--> %e
 %e  =
    2.7182818
```

```
--> %pi
 %pi  =
    3.1415927
```

%i représente la variable complexe **i** en entrée et s'affiche **i** en sortie :

```
--> 2+3*%i
 ans  =
    2. + 3.i
```

Pour ne pas afficher le résultat

En ajoutant un point-virgule « **;** » à la fin d'une ligne de commande, le calcul s'effectue mais le résultat ne s'affiche pas.

```
-->(1+sqrt(5))/2;
```

```
--> (1+sqrt(5))/2
 ans  =
    1.618034
```

Pour se rappeler le nom d'une fonction

Les noms des principales fonctions sont récapitulés au chapitre 3 de ce document (page 32). Par exemple :

```
--> exp(10)/factorial(10)
 ans  =
    0.0060699
```

À noter
Les fonctions disponibles sont également listées dans l'aide accessible en cliquant sur **? > Aide de Scilab** dans la barre des menus.

Il est possible d'utiliser la touche tabulation →| de son clavier pour compléter len nom d'une fonction ou d'une variable dont on a donné les premières lettres.

Par exemple, si l'on tape dans la console :

```
-->fact
```

et que l'on tape sur la touche tabulation, une petite fenêtre apparaît permettant de choisir entre toutes les fonctions et noms de variables commençant par **`fact`**, comme **`factorial`** et **`factor`**. Il suffit alors de double-cliquer sur la fonction souhaitée ou de la sélectionner avec la souris ou les touches ↑ ↓ et d'appuyer sur la touche Entrée (Windows et Linux) ou Retour (Mac OS X) pour l'insérer.

La barre de menus

Vous serez amené à utiliser tout particulièrement les menus listés ci-dessous.

Applications

- L'historique des commandes permet de retrouver toutes les commandes des sessions précédentes et de la session courante.
- Le navigateur de variables permet de retrouver toutes les variables utilisées précédemment au cours de la même session.

Édition

Préférences (dans le menu **Scilab** sous Mac OS X) permet de régler et de personnaliser les couleurs, les polices et la taille des caractères dans la console et dans l'éditeur, ce qui est très utile quand on projette sur un écran.

Cliquez sur **Effacer la console** pour effacer tout le contenu de la console. Dans ce cas, l'historique est conservé et les calculs effectués lors de la session restent en mémoire. Vous pourrez toujours revenir sur une commande qui a été effacée en utilisant les flèches du clavier.

Contrôle

Pour interrompre un programme en cours d'exécution, on peut :

- Taper **`pause`** dans le programme ou cliquer sur **Contrôle > Interrompre** dans la barre de menus (Ctrl X sous Windows et Linux ou Commande X sous Mac OS X), si le programme est déjà lancé. Dans tous les cas, l'invite de commande « **`-->`** » se transformera en « **`-1->`** », puis en « **`-2->`** »…, si l'opération est répétée.
- Pour revenir au moment de l'interruption du programme, taper **`resume`** dans la console ou cliquer sur **Contrôle > Reprendre**.
- Pour arrêter définitivement un calcul sans possibilité de retour, taper **`abort`** dans la console ou cliquer sur **Contrôle > Abandonner** dans la barre de menus.

L'éditeur

Taper directement dans la console a deux inconvénients : l'enregistrement n'est pas possible, et si plusieurs lignes d'instructions ont été tapées, les modifications ne sont pas aisées. Pour enchaîner plusieurs instructions, l'éditeur est l'outil approprié.

Ouvrir l'éditeur

Pour ouvrir l'éditeur à partir de la console, cliquez sur la première icône  dans la barre d'outils ou sur **Applications > SciNotes** dans la barre de menus.

L'éditeur s'ouvre avec un fichier par défaut qui s'intitule « **`Sans titre 1`** ».

Écrire dans l'éditeur

On tape du texte dans l'éditeur comme dans n'importe quel traitement de texte.

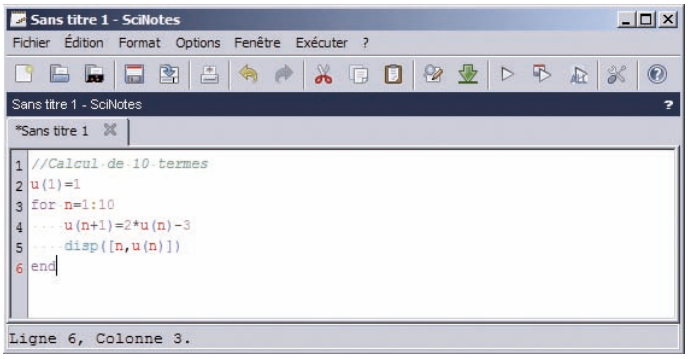
Dans l'éditeur de texte, l'apparition des parenthèses, ouvrantes et fermantes, et des commandes de fin de boucle, de fonction et de test est automatique. On peut cependant désactiver ces deux fonctionnalités dans le menu **Options > Complétion automatique**, en cliquant sur les deux entrées ci-dessous activées par défaut :

- **(,[,…**
- **if,function,…**

En principe, il faut aller à la ligne après chaque instruction, mais il est possible de taper plusieurs instructions sur une même ligne en les séparant par un point-virgule « **`;`** ».

Un décalage de début de ligne appelé indentation se fait automatiquement lorsqu'on commence une boucle ou un test.

Dans l'exemple suivant, on calcule 10 termes de la suite (u_n) définie par : $\begin{cases} u_1 = 1 \\ u_{n+1} = 2u_n - 3 \end{cases}$



À noter
- Pour écrire des commentaires qui ne seront pas pris en compte dans les calculs, les faire précéder de « **`//`** ».
- Pour changer la police, cliquez sur **Options > Préférences**.
- À l'écriture d'un programme, l'indentation est automatique. Si toutefois cela n'était pas le cas, cliquez sur **Format > Corriger l'indentation** pour la rétablir (Ctrl I sur Windows et Linux ou Commande I sur Mac OS X).

Enregistrer

Il est possible d'enregistrer tout fichier en cliquant sur **Fichier > Enregistrer sous**.

L'extension « .sce » à la fin du nom de fichier déclenchera automatiquement le lancement de Scilab à l'ouverture du fichier (excepté sous Linux et Mac OS X).

Copier dans la console, exécuter le programme

En cliquant sur Exécuter dans la barre de menus, trois options sont proposées :

- Exécuter « **…fichier sans écho** » (Ctrl Maj E sous Windows et Linux, Cmd Maj E sous Mac OS X) : le fichier est exécuté sans que le programme ne s'écrive dans la console (en ayant enregistré le fichier au préalable).
- Exécuter **« …fichier avec écho** » (Ctrl L sous Windows et Linux, Cmd L sous Mac OS X) : réécrit le fichier dans la console et l'exécute.
- Exécuter « **…jusqu'au curseur, avec écho** » (Ctrl E sous Windows et Linux, Cmd E sous Mac OS X) : réécrit la sélection choisie avec la souris dans la console et l'exécute ou exécute les données du fichier jusqu'à la position du curseur définie par l'utilisateur).

On peut aussi utiliser le copier/coller classique.

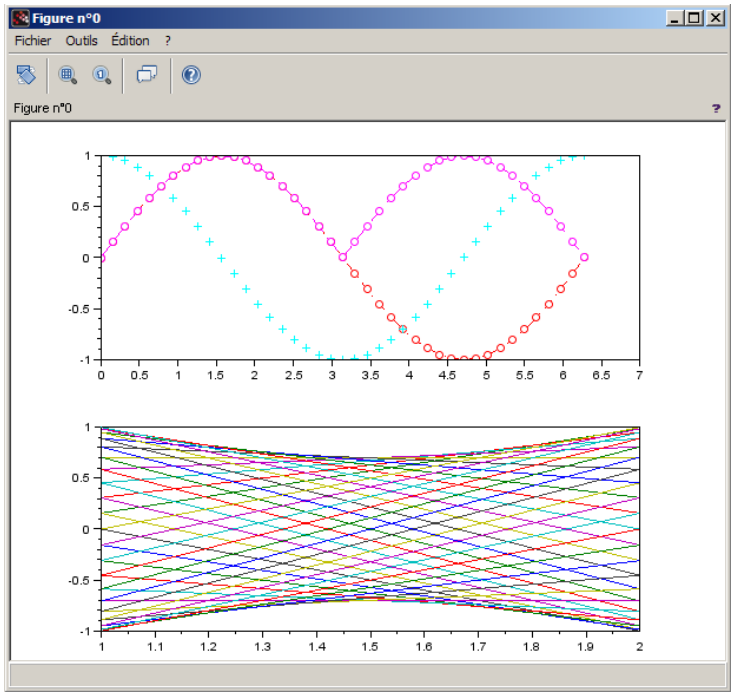
La fenêtre graphique

Ouvrir une fenêtre graphique

Une fenêtre graphique s'ouvre pour tout tracé. Il est possible de tracer des courbes, des surfaces et des nuages de points (voir le chapitre 2, page 18).

On obtient un exemple de courbe en tapant dans la console :

```
-->plot
```



À noter
- Pour effacer le tracé précédent, tapez **`clf`** (« clear figure » en anglais).
- Pour ouvrir une autre fenêtre graphique, tapez **`scf;`** (« set current figure » en anglais).
Si plusieurs fenêtres graphiques ont été ouvertes, on peut choisir celle dans laquelle on veut faire son tracé en tapant **`scf(n)`**; où n est le numéro de la fenêtre (indiqué en haut à gauche).

Modifier un tracé

La loupe  permet de faire un zoom. Pour effectuer un zoom en deux dimensions, cliquez sur l'icône et avec la souris créez un rectangle qui constituera la nouvelle vue agrandie. Pour effectuer un zoom en trois dimensions, cliquez sur l'icône et créez le parallélépipède qui constituera la nouvelle vue agrandie. Il est également possible de zoomer en utilisant la mollette de la souris. Pour revenir à l'écran initial, cliquez sur l'autre loupe  .

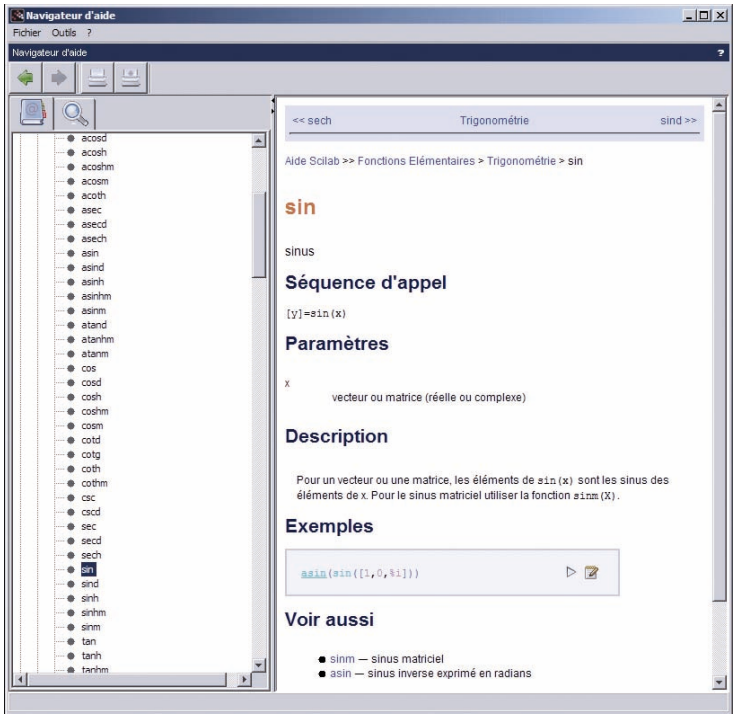
L'icône  permet de faire tourner la figure (particulièrement utile en 3D) avec des actions de clic droit qui seront guidées par des messages en bas de la fenêtre.

Pour des modifications plus précises, cliquez sur **Édition > Propriétés de la figure** ou **Propriétés des axes** et laissez-vous guider (cette option n'est pas encore disponible sous Mac OS X).

L'aide en ligne

Pour accéder à l'aide en ligne, cliquez sur **? > Aide Scilab** dans la barre de menus, ou tapez dans la console :

```
-->help
```



À noter
Une partie de l'aide est disponible en français, le reste est en anglais.
Des exemples d'utilisation peuvent être exécutés dans Scilab et édités dans SciNotes en utilisant les boutons associés dans le cadre de l'exemple.

Pour obtenir de l'aide sur des fonctions, tapez dans la console **`help`** et le nom de la fonction souhaitée. Par exemple :

```
-->help sin
```

affichera l'aide de la fonction **`sin`** (sinus).

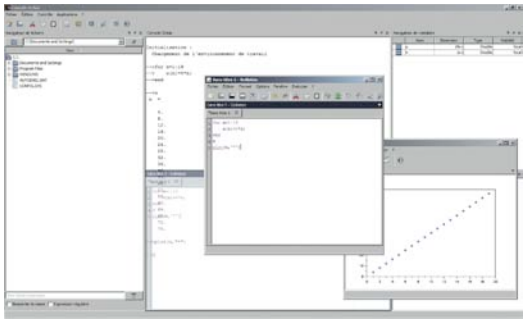
Gérer les fenêtres et personnaliser son espace de travail

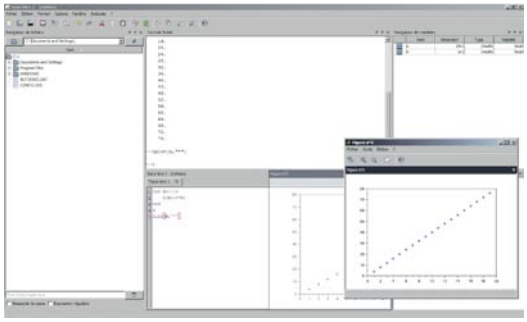
Comme pour l'environnement par défaut de Scilab, rassemblant les fenêtres de la console, les navigateurs de fichiers et de variables et l'historique des commandes, toutes les autres fenêtres de Scilab peuvent être repositionnées dans une seule et même fenêtre. Par exemple, l'utilisateur peut choisir de placer l'éditeur dans l'environnement par défaut de Scilab.

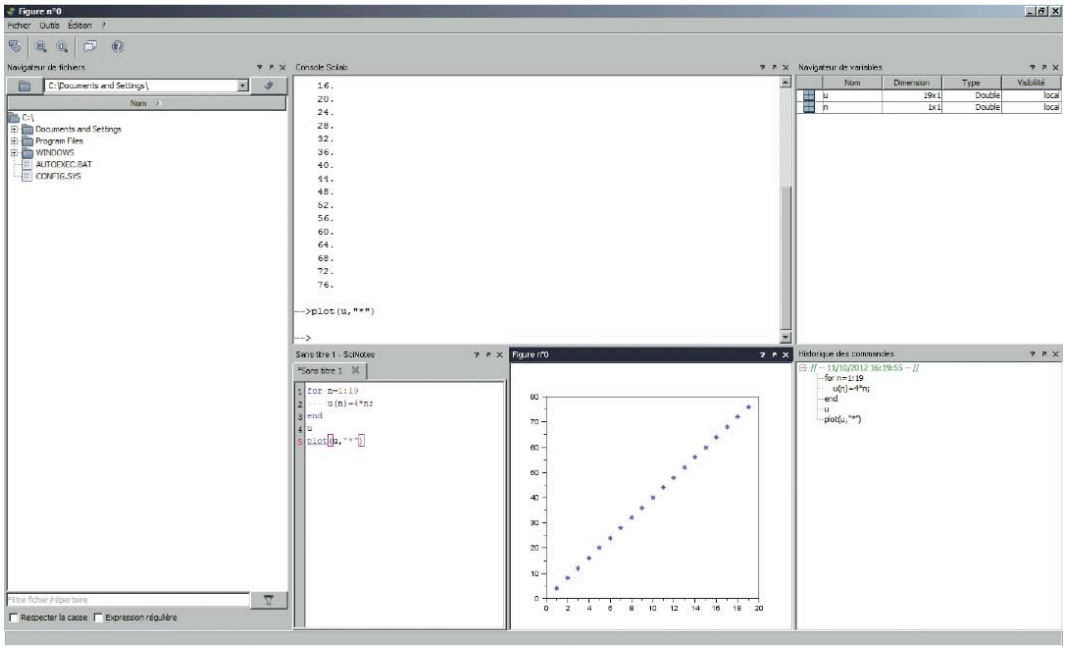
Pour placer une fenêtre dans une autre, on repère d'abord la barre horizontale bleue sous Windows, ou noire sous Mac OS X et Linux, située en haut de la fenêtre sous la barre d'outils et contenant un point d'interrogation à droite.

- Sous Windows et Linux, cliquez sur cette barre avec le bouton gauche de la souris, et, en maintenant ce bouton enfoncé, déplacez la flèche de la souris dans la fenêtre souhaitée.
- Sous Mac OS X, cliquez sur cette barre et, en maintenant le clic sur la souris, déplacez-la dans la fenêtre souhaitée.

Un rectangle apparaît indiquant le positionnement futur de la fenêtre. Lorsque la position est celle que vous souhaitez, relâchez le bouton de la souris. Pour annuler et faire ressortir la fenêtre, cliquez sur la petite flèche à droite de la même barre.







Chapitre 2 - Programmer

Dans les exemples donnés dans ce document, toute ligne précédée de « **-->** » est une commande, les autres lignes sont les retours (résultats de calcul, messages d'erreur…). Il ne faut pas écrire « **-->** » dans l'éditeur. Nous l'avons introduit uniquement pour bien différencier les lignes de commande des retours de calculs, l'affichage se faisant ainsi dans la console après un copier/coller. Présentées dans un tableau (sans « **-->** » et sans retour de calcul), les commandes sont donc représentées telles qu'elles devront être tapées dans l'éditeur.

Variables, affectation et affichage

Les variables

Scilab n'est pas un logiciel de calcul formel. Il calcule uniquement avec des nombres. Tous les calculs sont en réalité faits avec des matrices, mais cela peut passer inaperçu. Bien que la notion de matrice ne soit pas connue, on utilise les vecteurs et les suites de nombres qui sont, en fait, des matrices $1 \times n$ ou $n \times 1$, de même qu'un nombre est une matrice de dimension 1×1.

Les variables n'ont pas besoin d'être déclarées à l'avance, mais toute variable doit avoir une valeur. Par exemple, demander la valeur de la variable **a** sans lui avoir donné de valeur auparavant, produit une erreur :

```
-->a
  !--error 4
Variable non définie : a
```

Si l'on affecte une valeur à la variable **a**, il n'y a plus d'erreur :

```
--> a=%pi/4
 a  =
    0.7853982
--> a
 a  =
    0.7853982
```

On peut utiliser n'importe quel nom de variable qui n'est pas déjà défini par le système :

```
--> Pisur2=%pi/2
 Pisur2  =
    1.5707963
```

À noter
Tout comme les fonctions Scilab, un nom de variable ne doit pas comporter d'accents ou de caractères spéciaux.

Si l'on n'affecte pas le résultat d'un calcul à une variable, la valeur sera affectée automatiquement à la variable appelée **ans** :

```
-->3*(4-2)
 ans  =
    6.


-->ans
 ans  =
    6.
```

Les fonctions

Les fonctions sont le moyen le plus simple et le plus naturel pour faire des calculs à partir de variables et obtenir des résultats à partir de celles-ci.

La définition d'une fonction commence par **function** et finit par **endfunction**. Par exemple, pour transformer des euros (e) en dollars (d) avec un taux de change (t), définissons la fonction **dollars**. Les variables sont **e** et **t** et l'image est **d**.

```
-->function d=dollars(e,t); d=e*t; endfunction


-->dollars(200,1.4)
 ans  =
    280.
```

Le plus souvent, on utilise des fonctions numériques à une variable. Par exemple, deux fonctions **f** et **g** sont définies à l'aide des commandes ci-dessous :

```
-->function y=f(x); y=36/(8+exp(-x)); endfunction


-->function y=g(x); y=4*x/9+4; endfunction
```

À noter
Les variables **x** et **y** sont des variables muettes, leurs noms pouvant être réutilisés dans la définition d'autres fonctions ou dans Scilab.

Les fonctions ayant été définies, elles peuvent être utilisées pour calculer des valeurs :

```
--> f(10)
 ans  =
    4.4999745
--> g(12.5)
 ans  =
    9.5555556
```

Demander l'affectation d'une valeur

L'affectation se fait de façon simple avec le symbole « **=** ».

L'affichage

Écrire

Taper le nom d'une variable affiche sa valeur, sauf avec « **;** » en fin de commande.

Les crochets

Les crochets permettent de définir des matrices (voir la page 23). Comme énoncé précédemment, le calcul matriciel est à la base des calculs effectués dans Scilab. Un espace ou une virgule permet de passer d'une colonne à la suivante et un point-virgule, d'une ligne à l'autre.

Pour définir un vecteur colonne et obtenir un affichage en colonne :

```
-->v=[3;-2;5]
 v  =
    3.
  - 2.
    5.
```

Pour définir un vecteur ligne et obtenir un affichage en ligne :

```
-->v=[3,-2,5]
 v  =
    3.  - 2.    5.
```

À noter
Il est aussi possible de taper cette commande sous la forme : **v=[3 -2 5]**

Pour définir une matrice et afficher un tableau :

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

À noter
Il est aussi possible de taper cette commande sous la forme :
m=[1,2,3;4,5,6;7,8,9]

La fonction disp

La fonction **disp** est toujours suivie de parenthèses.

Avec le vecteur **v** précédent :

```
-->v(2)
 ans  =
  - 2.


-->disp(v(2))
  - 2.
```

Pour afficher une chaîne de caractères (en général une phrase), on la met entre guillemets :

```
-->disp("Bob a gagné")
  Bob a gagné
```

Pour mélanger des mots et des valeurs, utilisez la commande **string** qui transforme les valeurs en caractères, et « **+** » entre les différentes parties :

```
-->d=500;


-->disp("Bob a gagné "+string(d)+" dollars")
 Bob a gagné 500 dollars
```

Si la phrase contient une apostrophe, il est nécessaire de la doubler dans la chaîne de caractères pour qu'elle s'affiche correctement :

```
-->afficher("C''est juste")
 C'est juste
```

Les boucles

L'incrémentation

L'opérateur « **:** » permet de définir des vecteurs de nombres dont les coordonnées sont en suite arithmétique. On donne « la première valeur : le pas : la dernière valeur » (pas forcément atteinte). Si le pas n'est pas mentionné, sa valeur est 1 par défaut.

Par exemple, pour définir un vecteur ligne d'entiers qui s'incrémentent de 1 entre 3 et 10 :

```
-->3:10
 ans  =
    3.    4.    5.    6.    7.    8.    9.    10.
```

ou qui s'incrémentent de 2 entre 1 et 10 :

```
-->1:2:10
 ans  =
    1.    3.    5.    7.    9.
```

ou qui se décrémentent de 4 entre 20 et 2 :

```
-->u=20:-4:2
 u  =
    20.    16.    12.    8.    4.
```

For

La structure de boucle la plus simple pour un nombre connu d'itérations s'écrit avec **`for … end`** qui signifie « Pour … fin de pour ».

Exemple : Calcul de 20 termes d'une suite définie par récurrence par : $\begin{cases} u_1 = 4 \\ u_{n+1} = u_n + 2n + 3 \end{cases}$

Algorithme	Éditeur Scilab
Mettre 4 dans u(1)	`u(1)=4;`
Pour n allant de 1 à 20	`for n=1:20`
u(n+1) prend la valeur u(n)+2n+3	`u(n+1)= u(n)+2*n+3;`
Afficher n et u(n)	`disp([n,u(n)])`
Fin de pour	`end`

While

Si l'on veut que la boucle s'arrête lorsqu'un objectif donné est atteint, on utilisera la forme **`while … end`** qui signifie « Tant que … fin de tant que ».

Exemple : J'ai replanté en 2005 un sapin de Noël qui mesurait 1,20 m. Il grandit de 30 cm par an. J'ai décidé de le couper quand il dépasserait 7 m. En quelle année vais-je couper mon sapin ?

Algorithme	Éditeur Scilab

Mettre 1,2 dans h (h = hauteur du sapin)
Mettre 2005 dans a (a = année)
Tant que h<7
h prend la valeur h+0,3 (mon sapin grandit)
a prend la valeur a+1 (une année se passe)
Fin de tant que
Afficher a (la dernière année)
	`h=1.2;`
	`a=2005;`
	`while h<7`
	`h=h+0.3;`
	`a=a+1;`
	`end`
	`disp("Je couperai mon..`
	`sapin en "+string(a))`

À noter
Quand une commande est trop longue pour être écrite sur une seule ligne, l'éditeur va automatiquement à la ligne. On peut aussi mettre « **`..`** » (deux points) avant d'aller à la ligne.

Les tests

Les opérateurs de comparaison

Comparer des nombres ou vérifier si une affirmation est vraie ou fausse sont des tests utiles. Voici les commandes correspondantes :

Égal	Différent	Inférieur	Supérieur	Inférieur ou égal	Supérieur ou égal
==	**<>**	**<**	**>**	**<=**	**>=**
Vrai	Faux	Et	Ou	Non	
%T	**%F**	**&**	**|**	**~**	

À noter
Attention à la précision. Les calculs faits étant approchés, le test « **==** »donne parfois des réponses fausses (voir les problèmes de précision, page 30).

Lorsque l'on veut comparer deux vecteurs (ou deux matrices), les tests « **==** » et « **<>** » comparent terme à terme. Par exemple :

```
-->X=[1,2,5]; Y=[5,3,5];


-->X==Y
 ans  =
  F F T
```

Pour tester si deux vecteurs sont égaux, on utilise **isequal**, et s'ils sont différents, **~isequal** :

```
-->isequal(X,Y)
 ans  =
  F


-->~isequal(X,Y)
 ans  =
  T
```

If…then

Les structures classiques sont les suivantes :

- **if … then … else … end** (« Si…alors…sinon…fin de si »),
- **if … then … elseif … then … else … end** (« Si…alors…ou si…alors … ou … fin de si »).

if … then doivent être écrits sur la même ligne, de même que **elseif … then**.

Exemple : Alice lance trois dés.

- Si elle obtient trois 6, elle gagne 20 €,
- Si elle obtient trois résultats identiques différents de 6, elle gagne 10 €,
- Si elle obtient deux résultats identiques, elle gagne 5 €,
- Sinon, elle ne gagne rien.

On peut simuler un lancer et calculez le gain d'Alice, en utilisant les fonctions :

- **`grand`** (voir page 22),
- **`unique(D)`** qui ne garde qu'une fois les valeurs qui apparaissent plusieurs fois dans **`D`**,
- **`length(unique(D))`** qui donne la taille du vecteur ainsi obtenu, donc 1 si les trois termes sont égaux, 2 si deux termes sont égaux.

Algorithme	Éditeur Scilab

Mettre les trois valeurs dans D
Si Alice obtient trois 6, alors
  Alice gagne 20 euros
Sinon, si elle obtient 3 dés identiques, alors
  Alice gagne 10 euros
Sinon, si elle obtient 2 dés identiques, alors
  Alice gagne 5 euros
Sinon
  Alice ne gagne rien
Fin de si
Afficher le gain d'Alice
	`D=grand(1,3,"uin",1,6);`
	`if D==[6,6,6] then`
	`  G=20;`
	`elseif length(unique(D))==1 then`
	`  G=10;`
	`elseif length (unique(D))==2 then`
	`  G=5;`
	`else`
	`  G=0;`
	`end`
	`disp("Alice gagne "+..`
	`string(G)+ " euros")`

Les tracés en 2 et 3 dimensions

Les tracés dans le plan se font avec la commande **`plot`**. On peut choisir la couleur et l'aspect en mettant les indications de couleur et de style de points entre guillemets :

- Les couleurs

`"b"` = bleu (par défaut), **`"k"`** = noir, **`"r"`** = rouge, **`"g"`** = vert, **`"c"`** = cyan, **`"m"`** = magenta, **`"y"`** = jaune, **`"w"`** = blanc.

- Les styles de points

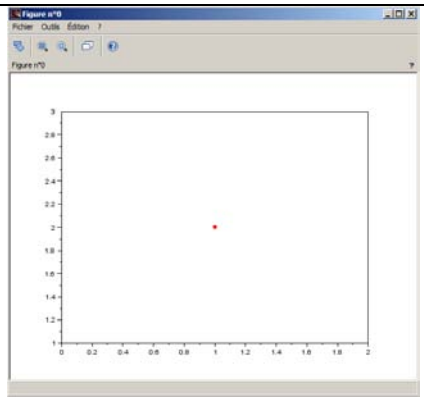
Reliés (par défaut), ou **`"."`**, **`"+"`**, **`"o"`**, **`"x"`**, **`"*"`**.

D'autres options sont disponibles avec : **`"s"`**, **`"d"`**, **`"v"`**, **`"<"`**, et **`">"`**.

Tracés de base

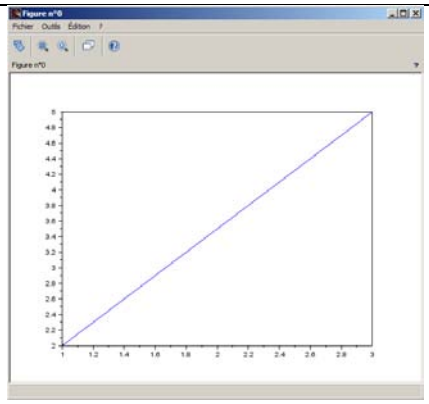
Pour tracer un point

Tracer le point A(1 ; 2) avec un point rouge.

Éditeur Scilab	Fenêtre graphique
`plot(1,2,".r")`	

Pour tracer un segment

Tracer le segment [AB] en bleu (par défaut) avec A(1 ; 2) et B(3 ; 5).

Éditeur Scilab	Fenêtre graphique
`plot([1,3],[2,5])`	

Tracés de courbes planes définies par des fonctions y=f(x)

Pour une fonction $x \to f(x)$ on donne d'abord les valeurs de x avec la commande **`linspace`** en écrivant : **`x=linspace(a,b,n);`** où **a** est la plus petite valeur de la variable x, **b** est la plus grande valeur de x, et **n** est le nombre de valeurs qui seront calculées entre **a** et **b**.

Ne pas oublier le « **`;`** » sinon les n valeurs de x s'afficheront.

Par exemple, soient deux fonctions f et g définies sur [-2 ; 5] par :

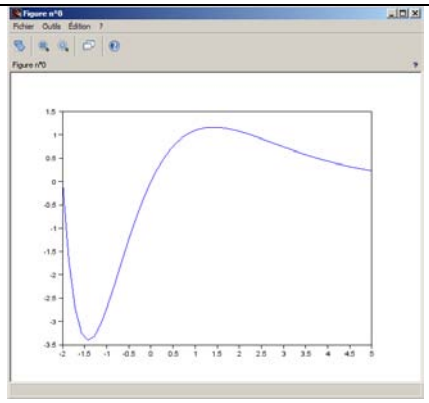
$f(x) = (x^2 + 2x)e^{-x}$, et $g(x) = \sin\left(\frac{x}{2}\right)$

Ci-dessous avec ce programme, on obtient le tracé de la courbe de f, en bleu par défaut.

Éditeur Scilab

```
function y=f(x)
  y=(x^2+2*x)*exp(-x)
endfunction
x=linspace(-2,5,50);
plot(x,f)
```

Fenêtre graphique

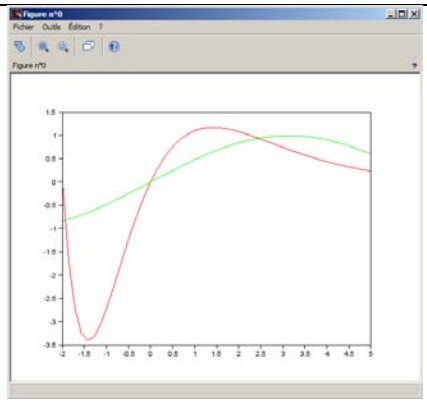


En ajoutant le programme ci-dessous, on obtient le tracé des deux courbes, celle de **f** en rouge et celle de **g** en vert. Le tracé précédent a été effacé grâce à la commande **clf** (« clear figure » en anglais).

Éditeur Scilab

```
function y=g(x)
  y=sin(x/2)
endfunction
x=linspace(-2,5,50);
clf
plot(x,f,"r",x,g,"g")
```

Fenêtre graphique



Tracés de nuages de points

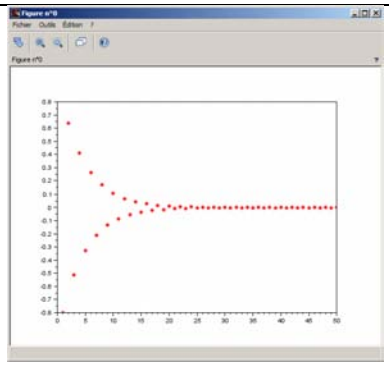
Termes d'une suite

Le cas le plus courant est celui où l'on veut tracer les points $M(n,u(n))$ après avoir calculé les coordonnées $u(n)$ d'un vecteur u. On écrit alors **plot(u,"*r")** en spécifiant la forme et la couleur des points du nuage entre guillemets. On a choisi ici des étoiles de couleur rouge qui ne sont pas reliées. Par défaut, les points sont bleus et reliés.

Éditeur Scilab

```
for n=1:50
  u(n)=(-0.8)^n;
end
clf; plot(u,"*r")
```

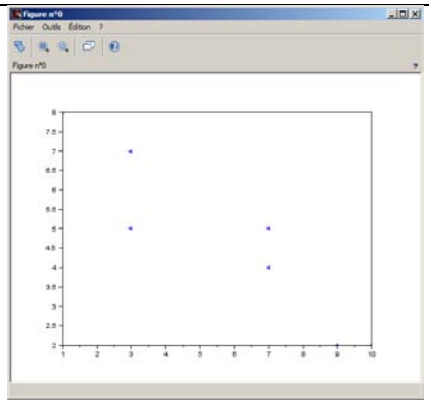
Fenêtre graphique



Statistiques doubles

Les nuages statistiques sont donnés sous la forme de deux vecteurs : appelons les X et Y, on écrira alors **`plot(X,Y,"<")`** pour tracer le nuage des points $M(X_i;Y_i)$ avec des triangles bleus.

Éditeur Scilab	Fenêtre graphique

`X=[1,3,3,7,7,9,10];`
`Y=[8,7,5,5,4,2,2];`
`clf; plot(X,Y,"<")`
	

Tracés en trois dimensions

Scilab permet de tracer des surfaces et des courbes dans l'espace avec un grand nombre d'options pour le traitement des faces cachées, la couleur des faces, les points de vue, etc. Nous ne donnerons ici que deux exemples.

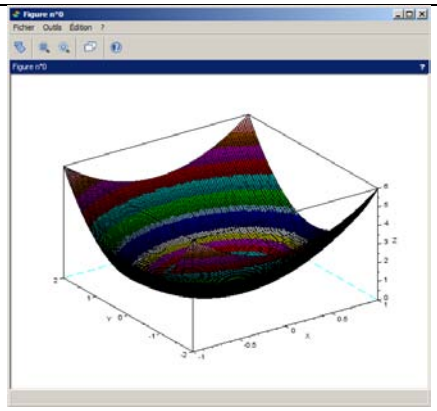
La fonction **`surf`** permet de tracer une surface. Cette fonction prend trois variables d'entrée, **`x`**, **`y`** et **`z`**. **`x`** et **`y`** sont des vecteurs de taille respective m et n correspondant à des points des axes (Ox) et (Oy). **`z`** est une matrice de dimension $n \times m$ dont l'élément z_{ij} est la cote du point de la surface d'abscisse x_i et d'ordonnée y_j.

Pour tracer la surface définie par une fonction du type $z = f(x,y)$, il faut donc :

- Définir la fonction f
- Calculer **`z=feval(x,y,f)'`**
  `feval(x,y,f)` retourne la matrice $m \times n$ dont l'élément ij est $f(x_i,y_j)$ que l'on va transposer en utilisant l'apostrophe « ' »
- Appliquer **`surf(x,y,z).`**

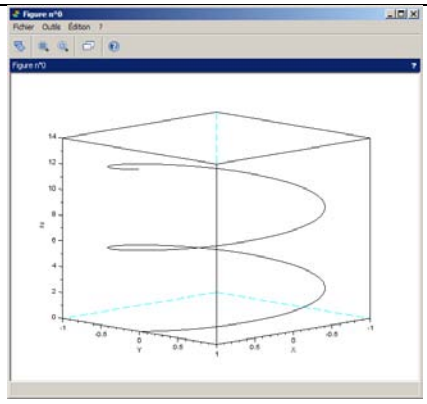
Le tracé de la surface $z = 2x^2 + y^2$ (paraboloïde elliptique) :

Éditeur Scilab	Fenêtre graphique

`function z=f(x,y)`
`  z=2*x^2+y^2;`
`endfunction`
`x=linspace(-1,1,100);`
`y=linspace(-2,2,200);`
`z=feval(x,y,f)';`
`clf`
`surf(x,y,z)`
	

La fonction **param3d** permet de tracer une courbe dans l'espace. **param3d** prend trois arguments, **x**, **y** et **z**, qui sont des vecteurs de même taille correspondant aux points (x_i, y_i, z_i) de la courbe.

Le tracé de l'hélice définie par $(x = \cos(t), y = \sin(t), z = t)$:

Éditeur Scilab	Fenêtre graphique
`t=linspace(0,4*%pi,100);` `param3d(cos(t),sin(t),t)`	

Simulations et statistiques

De nombreuses fonctions existent pour effectuer des simulations de façon rapide et performante.

Tirages aléatoires avec ordre et remise

- **grand(1,p,"uin",m,n)** retourne un vecteur de p tirages entiers aléatoires pris entre m et n avec p entier positif, m et n entiers et $m \leq n$.

```
-->t= grand(1,4,"uin",1,6)
 t  =
    3.    1.    3.    6.
```

- **grand(1,p,"unf",a,b)** retourne un vecteur de p tirages réels aléatoires pris entre a et b avec p entier positif, a et b réels et $a \leq b$.

```
-->tr= grand(1,2,"unf",-1,1)
 tr  =
  - 0.7460264    0.9377355
```

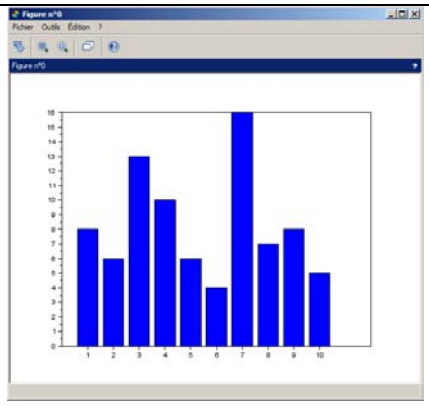
Statistiques

Toutes les fonctions statistiques habituelles sont listées à la page 32.

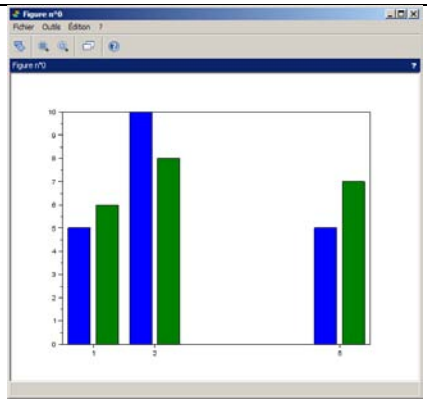
Gardez particulièrement en mémoire :

- La fonction **`bar(x,n,couleur)`** qui trace des diagrammes en barre :

Éditeur Scilab	Fenêtre graphique

`x=[1:10];`
`n=[8,6,13,10,6,4,16,7,8,5];`
`clf; bar(x,n)`
	

- Pour un diagramme en barres représentant deux séries côte à côte : soit la série de valeurs X, et les deux séries d'effectifs n1 et n2. Pour le tracé, n1 et n2 doivent être des vecteurs colonne, c'est pourquoi dans l'exemple ci-dessous, on prend les transposées :

Éditeur Scilab	Fenêtre graphique
`X=[1,2,5];n1=[5,10,5];n2=[6,8,7];` `bar(X,[n1',n2'])`	

L'argument optionnel **`couleur`** définit la couleur comme dans la fonction **`plot`**.

Compléments sur les matrices et les vecteurs

Accéder aux éléments

Les crochets permettent de définir une matrice. Un espace ou une virgule permet de passer d'une colonne à la suivante et un point-virgule, d'une ligne à l'autre.

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

À noter
Il est aussi possible de taper cette commande sous la forme :
`m=[1,2,3;4,5,6]`

Les parenthèses permettent d'accéder aux éléments ou de les modifier.

```
-->m(2,3)
 ans  =
    6.


-->m(2,3)=23
 m  =
    1.    2.    3.
    4.    5.    23.
```

L'opérateur « : » sert à désigner toutes les lignes ou toutes les colonnes d'une matrice.

Pour avoir la deuxième ligne de la matrice **m**, tapez :

```
-->m(2,:)
 ans  =
    4.    5.    23.
```

et la troisième colonne :

```
-->m(:,3)
 ans  =
    3.
    23.
```

Pour obtenir la transposée d'une matrice ou d'un vecteur, on utilise l'apostrophe « ' » :

```
-->m'
 ans  =
    1.    4.
    2.    5.
    3.    23.
```

Opérations

Les opérations « ***** », « **/** » sont des opérations matricielles. Pour faire des opérations élément par élément, on fera précéder le signe opératoire d'un point : « **.*** », « **./** ».

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

```
-->B=[1;1;2]
 B  =
    1.
    1.
    2.
```

```
-->A*B
 ans  =
    9.
    21.
```

Multiplication matricielle

```
-->A*A
    !--error 10
Multiplication incohérente.
```

Les dimensions ne sont pas bonnes

```
-->A.*A
 ans  =
    1.     4.     9.
    16.    25.    36.
```

Multiplication élément par élément

```
-->2*(A+2)
 ans  =
    6.     8.     10.
    12.    14.    16.
```

L'opération se fait sur chaque élément car 2 est un nombre

```
-->A/A
 ans  =
    1.            1.518D-16
    3.795D-15     1.
```

Donne la matrice X telle que X*A = A
La réponse exacte est :
1. 0
0 1.

Pour des raisons de précision de calcul, le résultat peut être légèrement différent suivant votre version de Scilab et votre système d'exploitation (voir les précisions de calcul, page 29)

```
-->A./A
 ans  =
    1.     1.     1.
    1.     1.     1.
```

Donne la matrice divisée élément par élément

Dans le cas des vecteurs :

```
-->C=1:4
 C  =
    1.     2.     3.     4.
```

```
-->C*C
    !--error 10
Multiplication incohérente.
```

Les dimensions ne sont pas bonnes

`-->C.*C`
` ans  =`
`    1.    4.    9.    16.`
	Il est aussi possible d'écrire **`C^2`** car, pour les vecteurs, l'écriture avec un exposant se traduit par une opération élément par élément. Ce n'est pas le cas pour les matrices

`-->1/C`
` ans  =`
`    0.0333333`
`    0.0666667`
`    0.1`
`    0.1333333`
	Dans ce cas spécifique aux vecteurs, on trouve le vecteur X tel que C*X = 1

`-->(1)./C`
` ans  =`
`    1.    0.5    0.3333333`
`0.25`
	Inverse élément par élément
	Comme précédemment, **`C^(-1)`** aurait été possible. Les parenthèses autour de 1 sont nécessaires pour que le point ne soit pas considéré comme une virgule faisant partie du nombre 1. On peut aussi écrire **`1 ./C`** avec un espace entre **`1`** et « **`.`** »

Résolutions de système

Pour résoudre le système linéaire AX = Y, où A est une matrice carrée, utilisez l'anti-slash « \ »

X = A \ Y.

Attention, l'opération Y / A donnera (à condition que les dimensions soient bonnes) un autre résultat, soit la matrice Z telle que Z A = Y.

Pour résoudre le système : $\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\end{pmatrix} \times X = \begin{pmatrix}1\\1\end{pmatrix}$

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


-->Y=[1;1];


-->X=A\Y
 X  =
  - 0.5
    0.
    0.5


-->A*X
 ans  =
    1.
    1.
```

Quelques fonctions utiles

Trier

La fonction **gsort** permet d'ordonner par ordre croissant ou décroissant les éléments d'un vecteur.

```
-->v=[2,6,9,6,-4,0,2]
 v  =
    2.    6.    9.    6.  - 4.    0.    2.


--> gsort(v,"g","i")
 ans  =
  - 4.    0.    2.    2.    6.    6.    9.


--> gsort(v)
 ans  =
    9.    6.    6.    2.    2.    0.  - 4.
```

Taille

La fonction **length** retourne le nombre de coordonnées dans le cas d'un vecteur. La fonction **size** retourne les dimensions (lignes, colonnes) dans le cas d'une matrice.

```
-->U=[1:10]
 U  =
    1.    2.    3.    4.    5.    6.    7.    8.    9.    10.


-->length(U)
 ans  =
    10.


-->m=[1 2 3;4 5 6];


-->size(U)
 ans  =
    2.    3.
```

Somme et produit

Les fonctions **sum** et **prod** calculent respectivement la somme et le produit des éléments de leur argument.

```
-->U=[1:10];


-->sum(U)
 ans  =

    55.


-->prod(U)
 ans  =

    3628800.
```

Unique

La fonction **unique** ne garde qu'une fois les éléments dans un vecteur (même si ceux-ci sont répétés plusieurs fois) et les ordonne par ordre croissant. Elle peut être très utile pour faire des tests.

```
-->v=[2,6,9,6,-4,0,2]
 v  =

    2.    6.    9.    6.  - 4.    0.    2.


-->unique(v)
 ans  =

  - 4.    0.    2.    6.    9.
```

Trouver

La fonction **find** permet de rechercher des éléments dans un vecteur ou une matrice et retourne un vecteur contenant les indices correspondants.

Pour trouver tous les éléments du vecteur *w* plus petits que 5 :

```
-->w=[1,5,3,8,14,7,3,2,12,6]; find(w<5)
 ans  =

    1.    3.    7.    8.
```

Le vecteur résultat (1,3,7,8) nous indique que les éléments w_1, w_3, w_7 et w_8 sont plus petits que 5.

Pour trouver tous les éléments du vecteur w égaux à 3 :

```
-->w=[1,5,3,8,14,7,3,2,12,6]; find(w==3)
 ans  =
    3.    7.
```

Le vecteur résultat (3,7) indique que les éléments w_3 et w_7 sont égaux à 3.

Problèmes de précision

Pour le calcul

Les nombres ont une valeur absolue comprise entre environ $2{,}2\times10^{-308}$ et $1{,}8\times10^{+308}$.

Le nombre **`%eps`** égal à **`2.220446049D-16`** donne la plus petite précision relative que l'on puisse espérer dans le calcul, soit environ 16 chiffres.

Exemple 1 : Calcul de sin(π)

```
-->sin(%pi)
 ans  =
1.225D-16
```

La valeur de $sin(\pi)$ ci-dessus n'est pas 0, mais on la considère comme nulle. En effet, par rapport à la valeur maximale de la fonction sinus (soit 1), elle est égale à 0 avec une erreur inférieure à **`%eps`**.

Exemple 2 : Testons si le triangle de côtés $\sqrt{3}$, 1 et 2 est rectangle :

```
-->a=sqrt(3)
 a  =
    1.7320508
```

```
-->b=1
 b  =
    1.
```

```
-->c=2
 c  =
    2.
```

```
-->a^2+b^2==c^2
 ans  =
  F
```

Le programme répond faux car la valeur de a^2+b^2 est approchée

```
-->abs(a^2+b^2-c^2)<%eps
 ans  =
  F
```

Le programme répond faux car la précision demandée est absolue

```
-->abs(a^2+b^2-c^2)/c^2<%eps
 ans  =
  T
```

Le programme répond vrai car la précision demandée est relative

Pour l'affichage

Par défaut, les résultats sont affichés avec 10 caractères, comprenant le point décimal et le signe. La fonction **`format`** permet d'afficher plus de chiffres. Pour avoir 20 chiffres, vous taperez alors **`format(20)`**.

Reprenons $a = \sqrt{3}$:

```
-->a^2
 ans  =
    3.
```

Ici, il y a 7 décimales, on ne voit pas la précision

```
-->format(20)

-->a^2
 ans  =
    2.99999999999999956
```

Ici, il y a 17 décimales, on voit la précision

Résolution d'équations différentielles

Nous montrerons ici comment on peut trouver les solutions d'un système explicite d'équations différentielles. Le principe consiste à se ramener à des équations différentielles d'ordre 1 :

$\begin{cases} y'(t) = f\big(t, y(t)\big) \\ y(t_0) = y_0 \end{cases}$ où t_0 et t représentent le temps et y_0 et $y(t)$ sont des vecteurs,

puis, à appliquer la fonction **`ode`** : y=ode(y0,t0,t,f), avec :

- **`y0`** : condition initiale, vecteur de dimension n,
- **`t0`** : instant initial,
- **`t`** : vecteur de dimension T des instants où l'on veut avoir la solution. Ce vecteur doit commencer par **`t0`**,
- **`f`** : fonction définissant le système sous la forme :

```
function yprim=f(t,y)
    yprim(1)=...
    yprim(2)=...
    ....
    yprim(n)=...
endfunction
```

La solution y est une matrice de dimension $n \times T$: $\begin{pmatrix} y_1(1) & y_1(2) & \vdots & y_1(T) \\ y_2(1) & y_2(2) & \vdots & y_2(T) \\ \vdots & \vdots & & \vdots \\ y_n(1) & y_n(2) & \dots & y_n(T) \end{pmatrix}$

Exemple : Résoudre l'équation différentielle $\begin{cases} y'' = -4y \\ y(0) = 3, y'(0) = 0 \end{cases}$

On ramène cette équation d'ordre 2 à un système de deux équations d'ordre 1 en posant :

$Y = \begin{pmatrix} Y(1) \\ Y(2) \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix}, Yprim = \begin{pmatrix} Yprim(1) \\ Yprim(2) \end{pmatrix} = \begin{pmatrix} y' \\ y'' \end{pmatrix}$ et $\begin{matrix} Yprim(1) = (2) \\ Yprim(2) = -4 \times Y(1) \end{matrix}$

Commentaire	Éditeur Scilab

On définit la fonction qui aux deux variables t et y (qui est un vecteur) fait correspondre le vecteur y'	`function yprim=f(t,y)`
	`  yprim(1)=y(2);`
	`  yprim(2)=-4*y(1) ;`
	`endfunction`

On précise les valeurs de t pour le graphique (le logiciel choisit lui-même les valeurs de t pour son calcul interne de solution).	`t0=0; tmax=5;` `t=t0:0.05:tmax;`
On précise les conditions initiales	`y0=3; yprim0=0;`
On applique ode	`y=ode([y0;yprim0],t0,t,f);`
On trace la courbe intégrale de y en fonction de t	`clf; plot(t,y(1,:))`

Chapitre 3 – Fonctions Scilab utiles

Pour l'analyse

- **`sqrt(x)`** retourne la racine carrée de x pour x réel positif ou nul, et la racine complexe de partie réelle positive sinon.
- **`log(x)`** retourne le logarithme népérien de x avec x nombre réel ou complexe.
- **`exp(x)`** retourne l'exponentielle de x avec x nombre réel ou complexe.
- **`abs(x)`** retourne la valeur absolue du réel x (ou le module si x est complexe).
- **`int(x)`** retourne la troncature du réel x (entier avant la virgule).
- **`floor(x)`** retourne la partie entière du réel x (entier n tel que $n \leq x < n+1$).
- **`ceil(x)`** pour x réel retourne l'entier n tel que $n-1 < x \leq n$.

Pour les probabilités et statistiques

- **`factorial(n)`** retourne la factorielle de n avec n entier positif ou nul.
- **`grand(1,p,"uin",m,n)`** retourne un vecteur de p tirages entiers pris entre m et n avec p entier positif, m et n entiers et $m \leq n$.
- **`grand(1,p,"unf",a,b)`** retourne un vecteur de p tirages réels pris entre a et b avec p entier positif, a et b réels et $a \leq b$.
- **`sum(n)`** retourne la somme des valeurs du vecteur n (sert à calculer un effectif total).
- **`cumsum(n)`** retourne le vecteur des valeurs cumulées croissantes du vecteur n (sert à calculer les effectifs cumulés croissants).
- **`length(v)`** retourne le nombre de coordonnées du vecteur v.
- **`gsort(v)`** retourne trié le vecteur de nombres ou de chaînes de caractères v dans l'ordre décroissant.
- **`gsort(v,"g","i")`** retourne trié le vecteur de nombres ou de chaînes de caractères v dans l'ordre croissant.
- **`mean(v)`** retourne la moyenne du vecteur de nombres v.
- **`stdev(v)`** retourne l'écart type du vecteur de nombres v.
- **`bar(v,n,couleur)`** trace le diagramme en barre avec v en abscisse, n en ordonnée, v et n étant des vecteurs lignes de même dimension. Par défaut, **`bar(n)`** trace le diagramme en barres de n en bleu avec 1,2,3… en abscisses.
- **`bar(v,[n1',n2'])`** trace un diagramme en barre double avec v en abscisse, n1 en ordonnée en bleu et n2 en ordonnée en vert, avec v, n1 et n2 vecteurs lignes de même dimension.
- **`rand(n,p)`** avec n et p entiers positifs, retourne une matrice $n \times p$ de nombres pris aléatoirement entre 0 et 1.
- **`rand()`** retourne un nombre réel pris aléatoirement entre 0 et 1.
- **`floor(x)`** retourne la partie entière du nombre réel x. En particulier, si p est un réel compris entre 0 et 1, **`floor(rand()+p)`** vaudra 1 avec une probabilité p et 0 avec une probabilité $1-p$.

Pour afficher et tracer

- **`clf`** signifie « clear figure » et efface la figure présente sur la fenêtre graphique.
- **`plot`** permet de tracer des courbes ou des nuages de points en dimension 2.
- **`linspace(a,b,n)`**, avec a et b réels et n entier, définit un vecteur de n valeurs régulièrement espacées entre a et b.
- **`scf`** permet d'ouvrir ou de sélectionner d'autres fenêtres graphiques.
- **`surf`** permet de tracer des surfaces en dimension 3.
- **`bar(X,Y)`** où X et Y sont des vecteurs, dessine le diagramme en bâtons de la série des valeurs de X ayant pour effectifs les valeurs de Y.
- **`plot(X,Y,"*")`** trace le nuage des points de coordonnées (X(i),Y(i)) sous forme d'étoiles. On peut préciser la couleur.
- **`plot(Y,"+")`** trace le nuage des points de coordonnées (i,Y(i)) sous forme de croix.
- **`disp("Phrase")`** affiche ce qui est écrit entre les guillemets.
- **`disp(A)`** où A est une matrice de nombres affiche le tableau des valeurs de A.
- **`disp("Phrase"+string(x))`** affiche la phrase et la valeur du nombre x.
- **`xclick`** retourne les coordonnées du point cliqué sur la fenêtre graphique.

Utilitaires

- **`unique(v)`** retourne le vecteur v avec une seule occurrence de ses éléments dupliqués.
- **`sum(v)`** retourne la somme de tous les éléments du vecteur ou de la matrice v.
- **`prod(v)`** retourne le produit de tous les éléments du vecteur ou de la matrice v.
- **`find(<test sur v>)`** retourne les indices des éléments du vecteur v vérifiant le test.
- **`disp(x,y,…)`** affiche les valeurs de ses arguments dans la console.
- **`string(x)`** transforme le nombre x en chaîne de caractères.
- **`format(n)`** où n est un entier supérieur ou égal à 2 fixe l'affichage à n caractères, y compris le signe et la virgule.
- **`zeros(n,p)`** définit une matrice $n \times p$ ne contenant que des 0.
- **`feval(x,y,f)`** où x et y sont des vecteurs de tailles respectives m et n, définit la matrice $m \times n$ dont l'élément (i,j) est $f(x(i),y(j))$.
- **`help`** fonction ouvre le navigateur d'aide à la page de la fonction.
- **`tic`** déclenche un chronomètre.
- **`toc`** arrête le chronomètre.