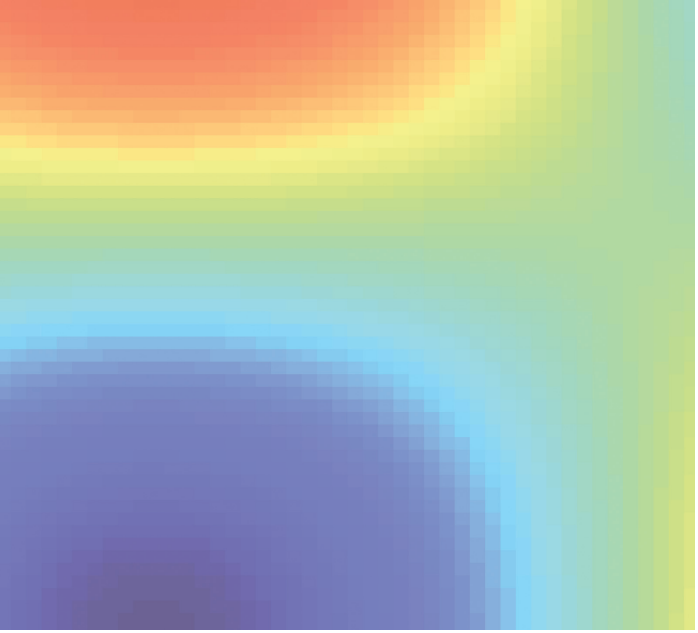
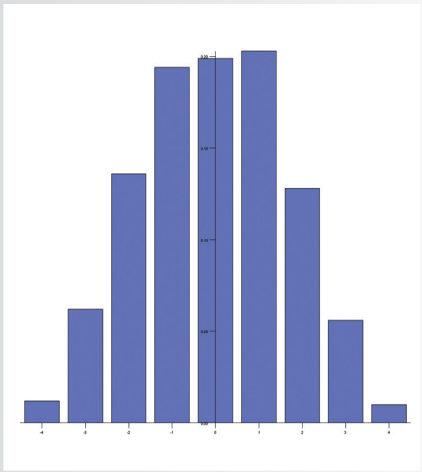
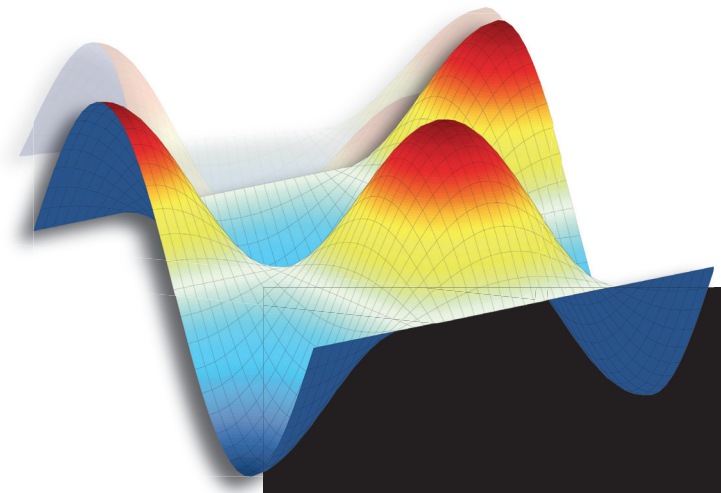


```
for n=1:1000
  s(n)=sum(tirage_entier(4,-1,1));
end
for i=1:9
  Y(i)=frequence(i-5,s);
end
clf; bar([-4:4],Y)
```





Scilab

Scilab pour les Lycées

Scilab pour les Lycées

Réalisé avec le soutien de l'INRIA Paris - Rocquencourt, ce livret a été co-écrit par le Consortium Scilab et Christine Gomez, professeur de mathématiques au lycée Descartes à Antony. Le Consortium Scilab remercie Christine Gomez et toutes les personnes qui ont relu le livret, en particulier Jocelyne Martin, professeur au lycée Talma à Brunoy.

© 2010 Consortium Scilab - Digiteo. Tous droits réservés.

Introduction

À propos de ce livret 5
Installation de Scilab 5
Liste de diffusion et d'information 6
Ressources complémentaires 6

1- Se familiariser à Scilab

La console 7
L'éditeur 11
La fenêtre graphique 12
L'aide en ligne 14
Gérer les fenêtres et personnaliser son espace de travail 14

2- Programmer avec Scilab

Variables, affectation et affichage 17
Les boucles 22
Les tests 24
Les tracés en 2 et 3 dimensions 26
L'arithmétique 34
Compléments sur les matrices et les vecteurs 34
Problèmes de précision 41

3- Exemples d'utilisation de Scilab au lycée

Variables, affectation, affichage 45
Boucles 48
Tests 54
Tracés de courbes 57
Simulations 60
Arithmétique 65
Divers 71

4- Fonctions Scilab utiles au lycée

Analyse 77
Arithmétique 77
Probabilités et statistiques 78
Ensembles 79
Graphique 80
Utilitaires 80

Introduction

À PROPOS DE CE LIVRET

L'objectif de ce livret est de vous guider pas à pas dans la découverte des différentes fonctionnalités de base du logiciel Scilab dans le cadre d'une utilisation en classes de lycées. Cette présentation se limite volontairement à l'essentiel pour permettre une prise en main facilitée de Scilab.

Les calculs, graphiques et illustrations du livret sont réalisés avec Scilab 5.2 enrichi du module lycée. Vous pouvez donc réaliser toutes les commandes présentées à partir de la version 5.2 de Scilab.

INSTALLATION DE SCILAB

Scilab est un logiciel de calcul numérique que chacun peut télécharger gratuitement. La version utile au lycée est la version de base du logiciel à laquelle le « Module Lycée » est ajouté. Ce module contient des fonctions spécifiques à l'enseignement des mathématiques au lycée.

Disponible sous Windows, Linux et Mac OS, Scilab est téléchargeable à l'adresse suivante : http://www.scilab.org/download

Une fois Scilab téléchargé et installé, il faut ajouter le module complémentaire appelé « Module Lycée ».
Pour cela, il suffit :

- d'être connecté à Internet,
- de lancer Scilab,
- dans la barre de menus, de cliquer sur **Applications > Gestionnaire de modules - ATOMS.**

La fenêtre ci-contre apparaît :

Cliquez sur **Module Lycée**, puis sur le bouton **Installer**.
Fermez Scilab, relancez-le, le module est installé.
Cette opération ne se fait qu'une fois, désormais le « Module Lycée » se chargera automatiquement au démarrage de Scilab sur votre poste de travail.

Si vous ne disposez pas de connexion Internet sur l'ordinateur sur lequel vous souhaitez installer Scilab et le module lycée, connectez-vous depuis un poste connecté à Internet à l'adresse suivante pour connaître la marche à suivre : http://www.scilab.org/lycee/index_lycee.php?page=download

Vous pouvez être averti des sorties de nouvelles versions de ce module en vous inscrivant sur une liste de diffusion (voir ci-après). La mise à jour est réalisée en suivant les mêmes opérations et en cliquant cette fois sur le bouton « Mettre à jour ».

Liste de diffusion et d'information

Pour faciliter échange et discussion entre les utilisateurs de Scilab pour les lycées, le Consortium Scilab a mis en place une liste de diffusion dédiée. Le principe est simple. Les personnes inscrites peuvent communiquer les unes avec les autres par courrier électronique (questions, réponses, propositions, retour d'expériences...). Il suffit d'envoyer son message à l'adresse de la liste de diffusion : lycee@lists.scilab.org, pour que celui-ci soit redistribué automatiquement à tous les inscrits de la liste. Pour vous inscrire, envoyez un courrier électronique (il n'est pas utile de mettre un sujet ou un texte dans celui-ci) à cette adresse : lycee-subscribe@lists.scilab.org. Vous recevrez une confirmation de votre inscription.

À noter
Les inscrits à la liste reçoivent également les dernières actualités consacrées à Scilab pour les lycées.

Ressources complémentaires

Si vous disposez d'une connexion Internet, vous pouvez accéder au site Web de Scilab sur lequel vous trouverez une rubrique entière consacrée à l'utilisation de Scilab au lycée (http://www.scilab.org/lycee), avec des actualités, des liens et des documents utiles, dont le présent livret, des exercices et des corrigés d'épreuves pratiques, pouvant être téléchargés et imprimés librement.

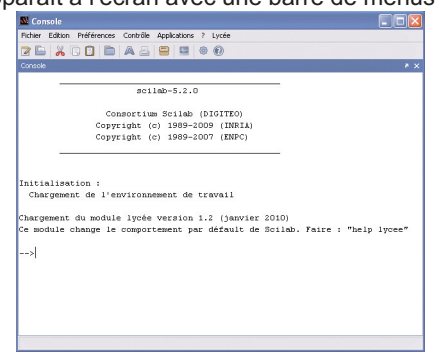
Chapitre 1
Se familiariser à Scilab

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

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

LA CONSOLE

Après avoir double-cliqué sur l'icône de Scilab pour lancer le logiciel, une fenêtre appelée console apparaît à l'écran avec une barre de menus et une barre d'outils :



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) 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 34).

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  =
    2.718281828459
```

```
-->%pi
 %pi  =
    3.1415926535898
```

`%i` représente le `i` des complexes 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 de la ligne de commande, le calcul s'effectue mais le résultat ne s'affiche pas.

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

```
-->(1+sqrt(5))/2
 ans  =
    1.6180339887499
```

Pour se rappeler le nom d'une fonction
Les noms des principales fonctions sont récapitulés au chapitre 4 de ce livret (page 77).

Par exemple :

```
-->exp(10)/factorielle(10)
 ans  =
    0.0060699034928
```

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

Par exemple, si l'on tape dans la fenêtre de commande :

```
-->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 `factorielle` et `factorise`. Il suffit alors de double-cliquer sur la fonction souhaitée ou de la sélectionner avec la souris et d'appuyer sur la touche **Entrée** (Windows et Linux) ou **Retour** (Mac OS) pour l'insérer.

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

La barre de menus

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

Préférences

Dans la barre de menus, **Préférences** permet de régler et de personnaliser les couleurs, les polices et la taille des caractères dans la console, ce qui est très utile quand on projette sur un écran devant une classe.

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 sur Windows et Linux ou Commande X sur Mac OS), si le programme est déjà lancé. Dans tous les cas, la flèche « `-->` » 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.

- Pour arrêter définitivement un calcul sans possibilité de retour :
  Taper `abort` dans la console.
  Ou cliquer sur **Contrôle > Abandonner** dans la barre de menus.

Lycée

À partir du menu Lycée, deux applications vous sont proposées pour illustrer votre cours en classe :

- **Ajustement affine** par la méthode des moindres carrés,
- **Calcul d'aire** pour l'encadrement de l'aire du domaine compris entre une courbe et l'axe des abscisses par la méthode des rectangles.

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 > Editeur** dans la barre de menus.

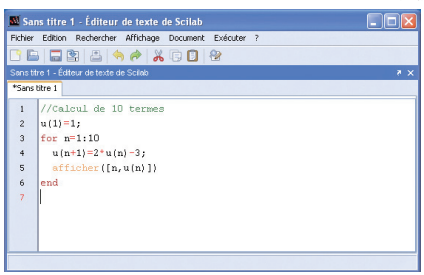
L'éditeur s'ouvre avec un fichier par défaut s'intitulant **`Sans titre 1`**.

Écrire dans l'éditeur

On tape du texte dans l'éditeur comme dans n'importe quel traitement de texte. 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 alors 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 « `//` ».

À noter
- Pour changer la police, cliquez sur **Affichage > Définir la police…**
- À l'écriture d'un programme, l'indentation est automatique. Si toutefois cela n'était pas le cas, cliquez sur **Edition > Corriger l'indentation** pour la rétablir (Ctrl I sur Windows et Linux ou Commande I sur Mac OS).

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 (sauf sous Linux).

Copier dans la console, exécuter le programme

Le plus simple est de sélectionner avec la souris la partie de code souhaitée, puis de faire un clic droit et de choisir **Exécuter la sélection dans Scilab** (sous Mac OS, appuyez sur la touche Commande et cliquez avec la souris pour accéder à ce même menu contextuel). Le morceau de code est alors copié, puis exécuté dans la console.

Une autre solution est possible. Cliquez sur **Exécuter** dans la barre de menus, trois options sont proposées :

- **Charger dans Scilab** (Ctrl L sur Windows et Linux ou Commande L sur Mac OS) : recopie tout le code dans la console (comme un copier / coller), et l'exécute.
- **Evaluer la sélection :** après avoir sélectionné un morceau de code, il est recopié, puis exécuté dans la console.
- **Execute le fichier dans Scilab** (Ctrl E sur Windows et Linux ou Commande E sur Mac OS) : le fichier contenant tout le code est exécuté mais n'est pas recopié dans la console. Seuls les résultats seront visibles si leur affichage a été demandé. **Pour cette option, le fichier doit d'abord avoir été enregistré.**

On peut aussi utiliser le copier / coller classique.

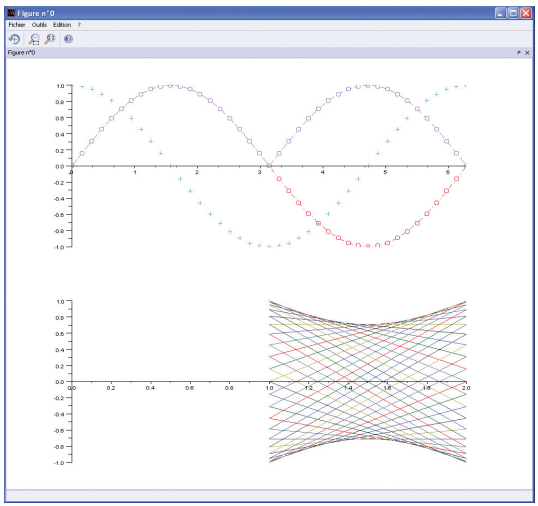
La fenêtre graphique

Ouvrir une fenêtre graphique

La fenêtre graphique s'ouvre pour tout tracé.
Il est possible de tracer des courbes, des surfaces, des nuages de points, des histogrammes (voir le chapitre 2, page 26).

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

```
-->plot
```



À noter
- Pour effacer un tracé précédent, tapez **`clf`** (« clear figure » en anglais).
- Pour ouvrir une autre fenêtre graphique, tapez **`scf;`** (« select 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 petite loupe permet de faire un zoom. Cliquez sur l'outil et avec la souris créez un rectangle qui constituera la nouvelle vue agrandie. Pour revenir à l'écran initial, cliquez sur l'autre loupe .
L'icône permet de faire tourner la figure (particulièrement utile en 3D).

Pour faire apparaître un quadrillage, tapez dans la console **`quadrillage`**.

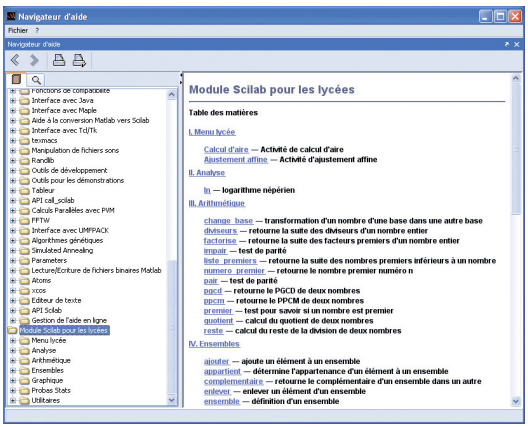
Pour des modifications plus précises, cliquez sur **Edition > Propriétés de la figure** ou **Propriétés des axes** et laissez-vous guider.

À noter
Cette dernière option n'est pas encore disponible sur Mac OS.

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 en utilisant le copier/coller entre la fenêtre d'aide et la console Scilab.
L'aide sur les fonctions du module lycée (en bas de la liste) est entièrement disponible en français.

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 sur la fonction sin (sinus).

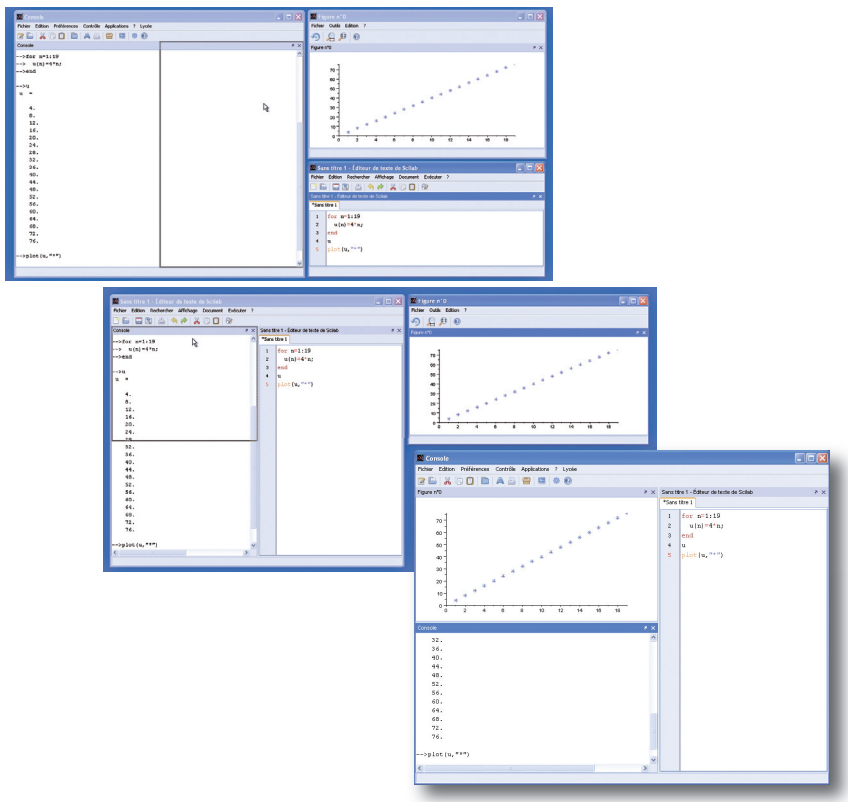
GÉRER LES FENÊTRES ET PERSONNALISER SON ESPACE DE TRAVAIL

À partir de la version 5.2 de Scilab, plusieurs fenêtres peuvent être repositionnées dans une seule et même fenêtre de travail. Par exemple, l'éditeur et une fenêtre graphique peuvent être placés dans la même fenêtre que la console.

Pour placer une fenêtre dans une autre, on repère d'abord la barre horizontale bleue sur Windows et noire sur Mac OS et Linux située en haut de la fenêtre sous la barre d'outils, contenant une petite flèche à droite avant la croix de fermeture.

- Sur 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.
- Sur Mac OS, 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 avec Scilab

Dans les exemples donnés dans ce livret, 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. Dans la plupart des classes de lycées, la notion de matrice n'est pas connue, mais 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  =
    0.7853981633974
```

```
-->a
 a  =
    0.7853981633974
```

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

```
-->Pisur2=%pi/2
 Pisur2  =
    1.5707963267949
```

Si l'on n'affecte pas la valeur 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 transformer des variables en d'autres variables.

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 au lycée des fonctions numériques à une variable réelle. 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
```

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

```
-->f(10)
 ans  =
    4.4999744626844
-->g(12.5)
 ans  =
    9.5555555555556
```

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.
- La commande **return** permet de sortir d'une fonction (voir l'exemple 26, page 66).

Demander l'affectation d'une valeur

L'affectation se fait de façon simple avec le symbole « = ».
Pour demander avec une phrase la valeur à attribuer à une variable, on utilise **input**, en mettant la phrase entre des guillemets droits « "" ».

Reprenons l'exemple du calcul des dollars à partir d'euros :

Éditeur Scilab	Console Scilab

`e=input("Somme en euros : ");`
`t=input("Taux de change : ");`
`d=e*t;`
`afficher("Somme en dollars : "+string(d))`
	`Somme en euros : 200`
	`Taux de change : 1.4`
	
	`Somme en dollars : 280`

Avec **input**, le programme attend une valeur.

Pour exécuter correctement le programme, enregistrez-le puis cliquez sur **Exécuter > Execute le fichier dans Scilab** dans la barre de menus. Renseignez alors dans la console, les valeurs au fur et à mesure qu'elles vous sont demandées (dans l'exemple, on a tapé `200`, entrée, puis `1.4`, entrée).

Si vous cliquez sur **Exécuter > Charger dans Scilab** ou sur **Exécuter > Evaluer la sélection**, ou lors d'un copier/coller, vous obtiendrez une erreur car le programme étant recopié, la valeur attendue pour `e` sera `t`, qui n'est pas un nombre.

L'affichage

Écrire

Taper le nom d'une variable affiche sa valeur, sauf avec « `;` » en fin de commande. Quand il y a beaucoup de valeurs à afficher, Scilab en donne quelques unes et demande s'il doit continuer : tapez `n` pour non ou n'importe quelle autre touche pour oui. Tant que l'utilisateur ne répond pas, Scilab attend et ne peut faire aucun autre calcul.

Les crochets

Les crochets permettent de définir des matrices (voir la page 34).
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 afficher
`afficher` est le nom francisé de la fonction **`disp`** utilisée dans Scilab sans le module lycée. Elle est toujours suivie de parenthèses.

Avec le vecteur v précédent :

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

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

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

```
-->afficher("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;

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

Si la phrase contient une apostrophe, il faut la doubler dans la chaîne de caractères :

```
-->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, par défaut, 1.

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 : $u_1 = 4$ et $u_{n+1} = u_n + 2n + 3$

Algorithme	Éditeur Scilab

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

À noter
Cette suite a été posée au bac en donnant u_0=1, mais pour définir le vecteur u, nous devons commencer à $u(1)$, première coordonnée de u.

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`
	`afficher("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 « . . » 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 41).

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 dés identiques différents de 6, elle gagne 10 €,
- si elle obtient deux dés identiques, elle gagne 5 €,
- sinon, elle ne gagne rien.

Simulez un lancer et calculez le gain d'Alice, en utilisant les fonctions :

- **`tirage_entier`** (voir la page 30),
- **`unique(D)`** qui ne garde qu'une fois les valeurs qui apparaissent plusieurs fois dans **`D`**,
- **`taille(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=tirage_entier(3,1,6);`
	`if D==[6,6,6] then`
	`  G=20;`
	`elseif taille(unique(D))==1 then`
	`  G=10;`
	`elseif taille(unique(D))==2 then`
	`  G=5;`
	`else`
	`  G=0;`
	`end`
	`afficher("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 point 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 :**

relié (par défaut), ou `"."`, `"+"`, `"o"`, `"x"`, `"*"`
D'autres options sont disponibles avec : `"s"`, `"d"`, `"v"`, `"<"`, et `">"`.

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

Pour une fonction $x \to f(x)$ définie sur un intervalle de $\mathbb{R}$, donnez avec la commande **`linspace`** les valeurs de x, 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`** 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 :

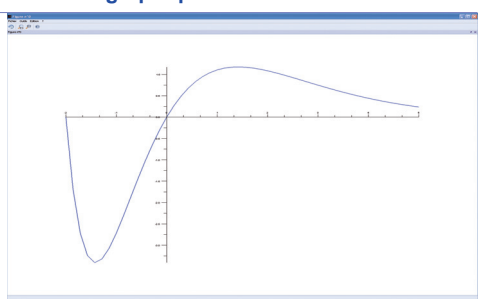
$$f(x) = (x^2 + 2x)e^{-x}, \text{ et } \quad g(x) = \sin\left(\frac{x}{2}\right)$$

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

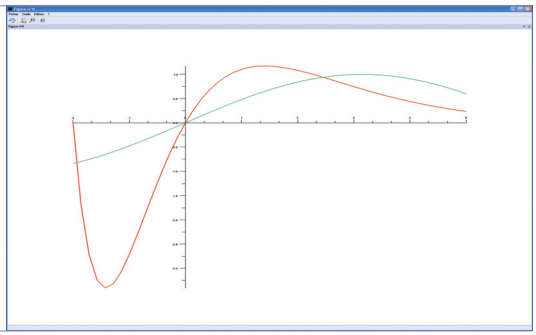
Éditeur Scilab

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

Fenêtre graphique



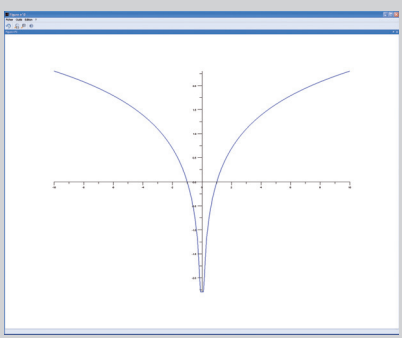
Éditeur Scilab	Fenêtre graphique

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

En ajoutant le programme ci-dessus, 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).

À noter

Les arguments de la fonction **plot** sont toujours des nombres réels. Si l'on donne des nombres complexes comme arguments, la fonction **plot** utilise leur partie réelle sans donner de message d'erreur.
Par exemple, si l'on trace la courbe de la fonction **ln** entre -10 et 10 en utilisant la commande **x=linspace(-10,10,100); plot(x,ln(x))**, la fonction **plot** tracera entre -10 et 10 la partie réelle du logarithme. On aura ainsi $\ln(3)$ comme image de -3 puisque $\ln(-3) = \ln(3e^{i\pi}) = \ln(3) + i\pi$.



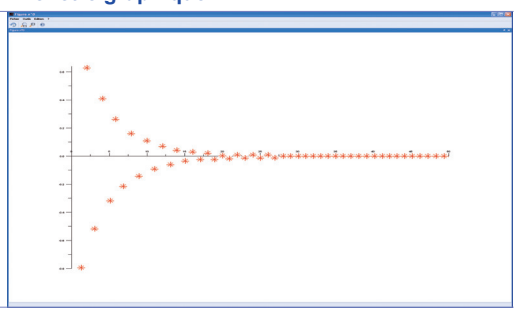
Tracés de nuages de points

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. Par exemple :

Éditeur Scilab	Fenêtre graphique

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

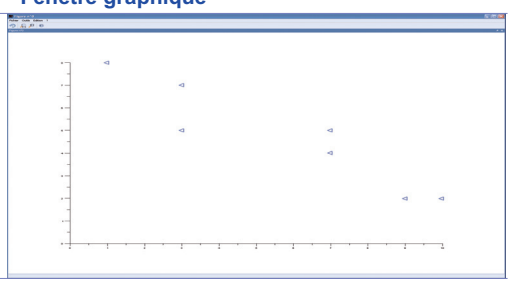


Statistiques doubles

Les nuages statistiques sont donnés sous la forme de deux listes (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. Par exemple :

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_j et d'ordonnée y_i.

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

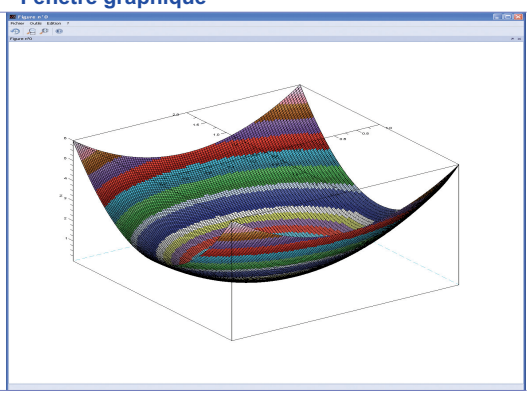
- définir la fonction f
- calculer **`z=(feval(x,y,f))'`** :
  `feval(x,y,f)` retourne la matrice $m \times n$ dont l'élément ij est $f(x_i, y_i)$ que l'on va transposer en utilisant l'apostrophe « **'** ».
- appliquer **`surf(x,y,z)`**.

Par exemple, pour le tracé de la surface $z = 2x^2 + y^2$ (paraboloïde elliptique) :

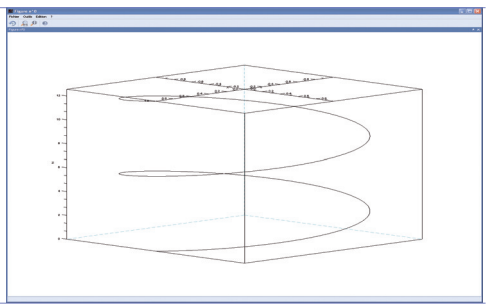
Éditeur Scilab

```
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)
```

Fenêtre graphique



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.
Par exemple, 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 ont été créées dans le module lycée pour faciliter les simulations de façon rapide et performante.

Tirages aléatoires avec ordre et remise

- **tirage_entier(p,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=tirage_entier(4,1,6)
 t  =
    3.    1.    3.    6.
```

- **tirage_reel(p,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=tirage_reel(2,-1,1)
 tr  =
  - 0.7460263762623    0.9377355421893
```

- **`frequence(n,s)`** retourne la fréquence de n dans la suite de nombres s avec n entier.
  Par exemple, pour obtenir la fréquence d'apparition du 6 dans 1000 lancers de dé :

```
-->t=tirage_entier(1000,1,6);

-->frequence(6,t)
 ans  =
    0.173
```

Tirages aléatoires sans ordre ni remise

Définir un ensemble

- La fonction **`ensemble`** permet de créer un ensemble :

```
-->E=ensemble("e1","e2","e3")
 E  =
 {e1,e2,e3}

-->E(1)
 ans  =
 e1
```

À noter
`ensemble()` crée un ensemble vide

Les éléments de l'ensemble **`e1`**, **`e2`**,... sont des chaînes de caractères. Un ensemble est non ordonné et n'a pas d'éléments dupliqués. Par exemple, {a,b,c} et {b,a,c,a} représentent le même ensemble. Lorsqu'un ensemble est créé, les éléments dupliqués sont supprimés et, par commodité, ces éléments sont ordonnés par ordre alphabétique.

Il est possible d'attribuer des valeurs à des éléments d'un ensemble, par exemple pour des ensembles de pièces, de billets ou de cartes à jouer avec des valeurs. La valeur est donnée en la mettant entre parenthèses à la fin du nom de l'élément.

Par exemple, pour un ensemble U contenant trois boules rouges numérotées 1, 2, 3 et deux boules noires numérotées 1,2, tapez :

```
-->U=ensemble("n(1)","n(2)","r(1)","r(2)","r(3)");
```

- La fonction **valeur** donne alors le vecteur des valeurs des éléments, permettant ainsi de les utiliser dans des calculs.

```
-->valeur(U)
 ans  =
    1.    2.    1.    2.    3.
```

- La fonction **taille** donne le nombre d'élements d'un ensemble.

Pour comparer deux ensembles, on utilise l'opérateur habituel « **==** ».

À noter
On dispose aussi des fonctions **ajouter**, **appartient**, **complémentaire**, **enlever**, **inclus**, **intersection**, **union**.

- Faire un tirage aléatoire dans un ensemble

On utilise la fonction **tirage_ensemble** en indiquant combien d'éléments on souhaite tirer.

Par exemple, supposons que l'on tire 2 boules dans l'ensemble U précédent et que l'on souhaite vérifier si elles sont rouges :

```
-->U=ensemble("n(1)","n(2)","r(1)","r(2)","r(3)");
-->R=ensemble("r(1)","r(2)","r(3)");
-->T=tirage_ensemble(2,U)
 T  =
 {n(1),r(3)}
-->if inclus (T,R)==%T then
-->afficher ("Les deux boules sont rouges")
-->else
-->afficher("Les deux boules ne sont pas rouges")
 Les deux boules ne sont pas rouges
-->end
```

Statistiques

Toutes les fonctions statistiques habituelles sont listées à la page 78.

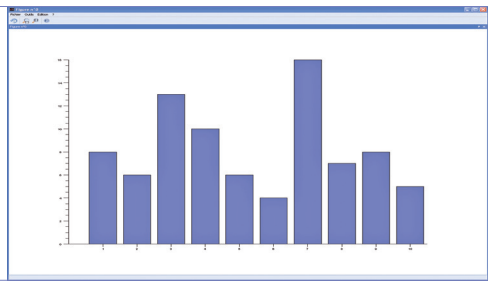
Gardez particulièrement en mémoire :

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

Éditeur Scilab

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

Fenêtre graphique

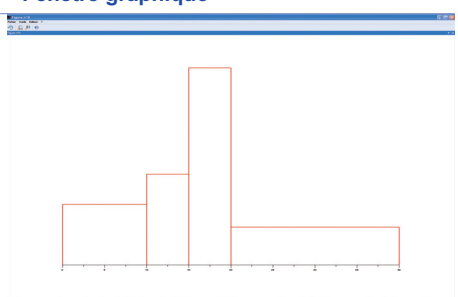


- La fonction **`histogramme(a,n,couleur)`** qui permet de tracer l'histogramme d'une série statistique où les valeurs de la variable sont regroupées dans des intervalles. **`a`** est le vecteur donnant les bornes des intervalles dans l'ordre croissant et **`n`** est le vecteur des effectifs ou des fréquences correspondants. Le vecteur **`a`** a un élément de plus que le vecteur **`n`**. L'argument optionnel **`couleur`** définit la couleur comme dans la fonction **`plot`**.

Éditeur Scilab

```
a=[0,10,15,20,40];
n=[8,6,13,10];
clf; histogramme(a,n,"r")
```

Fenêtre graphique



L'ARITHMÉTIQUE

Toutes les fonctions arithmétiques habituelles sont récapitulées à la page 77.

Gardez particulièrement en mémoire :

- La fonction **reste** qui donne le reste dans la division euclidienne.

Par exemple :

```
-->reste(75,4)
 ans  =
    3.

-->reste(-75,4)
 ans  =
    1.
```

- La fonction **diviseurs** qui donne tous les diviseurs positifs.

Par exemple :

```
-->diviseurs(75)
 ans  =
    1.    3.    5.    15.    25.    75.
```

- La fonction **factorise** qui donne la décomposition en facteurs premiers.

Par exemple :

```
-->factorise(75)
 ans  =
    3.    5.    5.
```

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. Par exemple, 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.                  1.518259871D-16
    3.795187214D-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 41).

```
-->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  =
    0.0333333333333
    0.0666666666667
    0.1
    0.1333333333333
```

Dans ce cas spécifique aux vecteurs, on trouve le vecteur X tel que C*X = 1

```
-->(1)./C
 ans  =
    1.    0.5    0.3333333333333    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, 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 **trier** 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.

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

-->trier(v,">")
 ans  =
  - 4.    0.    2.    2.    6.    6.    9.

-->trier(v,"<")
 ans  =
    9.    6.    6.    2.    2.    0.  - 4.
```

Taille

La fonction **taille** retourne le nombre de coordonnées dans le cas d'un vecteur, et les dimensions (ligne, colonnes) dans le cas d'une matrice :

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

-->taille(m)
 ans  =
    2.    3.

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

```
-->taille(U)
 ans  =
    10.
```

Somme et produit
Les fonctions **sum** et **prod** calculent respectivement la somme et le produit des éléments de leur argument. On reprend le vecteur U, vu au point précédent :

```
-->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 (voir l'exemple 23, page 63).

```
-->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 $2{,}2 \times 10^{-308}$ et $1{,}8 \times 10^{+308}$.

Le nombre **%eps** défini par **%eps = 2.220446049D-16** détermine la plus petite précision relative que l'on puisse espérer dans le calcul, ce qui donne donc environ 16 chiffres.

Exemple 1 - Calcul de sin(π)

```
-->sin(%pi)
 ans  =
    1.224646799D-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 valeur inférieure à **%eps**.

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

```
-->a=sqrt(3)
 a  =
    1.7320508075689
```

```
-->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 16 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 13 décimales, on ne voit pas l'erreur

```
-->format(20)

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

Ici, il y a 17 décimales, on voit l'erreur

L'erreur est de l'ordre de 4,4 x 10^{-16}, soit **`2*%eps`**. Ceci est logique puisque l'erreur est doublée à cause du carré ($(a+\mathcal{E})^2 \approx a + 2\mathcal{E}$), expliquant aussi que le programme réponde faux au test d'erreur absolue.

Chapitre 3
Exemples d'utilisation de Scilab au lycée

Variables, affectation, affichage

Exemple 1

Calculez le prix TTC d'un article à partir de son prix HT, avec un taux de TVA à 19,6 %.

Algorithme

Lire le prix hors taxes et le mettre dans HT
Calculer le prix avec taxes et le mettre dans TTC
Afficher TTC

Console Scilab

- Calcul et affichage simple

```
-->HT=540;
-->TTC=HT*1.196
 TTC  =
    645.84
```

- Demande de valeur et affichage mis en forme

```
-->HT=input("Prix hors taxe : ");
Prix hors taxe : 3789
-->TTC=HT*1.196;
-->afficher("Le prix avec taxes ..
est "+string(TTC))
 Le prix avec taxes est 4531.644
```

- Fonction

```
-->function TTC=f(HT);
-->TTC=HT*1.196;
-->endfunction
-->f(540)
 ans  =
    645.84
-->f(3789)
 ans  =
    4531.644
```

À noter
Les exemples peuvent être illustrés par l'affichage dans la console qui donne les commandes (précédées de « -->») et les retours, par l'affichage dans l'éditeur (sans « -->» et sans retours) et/ou par un graphique. Dans tous les cas, pour reproduire les exemples, ne tapez pas « -->» (celui-ci ne doit pas figurer dans l'éditeur et est affiché par défaut devant les lignes de commande dans la console).

Exemple 2

Convertissez un temps donné en secondes, en heures, minutes et secondes.

Algorithme

Lire le temps en secondes, le mettre dans t
Mettre dans q le quotient de t par 60 (nombre de minutes)
Mettre dans s le reste (nombre de secondes)
Mettre dans h le quotient de q par 60 (nombre d'heures)
Mettre dans m le reste (minutes restantes)
Afficher heures, minutes, secondes

Console Scilab

- Calcul et affichages simple

```
-->t=12680;
-->q=quotient(t,60); s=reste(t,60);
-->h=quotient(q,60); m=reste(q,60);
-->[h,m,s]
 ans  =
    3.    31.    20.
```

- Demande de valeur et affichage mis en forme

```
-->t=input("Heure en secondes : ");
Heure en secondes : 4586
-->q=quotient(t,60);s=reste(t,60);
-->h=quotient(q,60);m=reste(q,60);
-->afficher(string(h)+" heures "+string(m)+" minutes "+..
string(s)+" secondes ")
 1 heures 16 minutes 26 secondes
```

- Fonction

```
-->function y=f(t)
-->q=quotient(t,60);s=reste(t,60);
-->h=quotient(q,60);m=reste(q,60);
-->y=[h,m,s]
-->endfunction
-->f(12680)
 ans  =
```

```
    3.    31.    20.
-->f(4586)
 ans  =
    1.    16.    26.
```

Exemple 3

Calculez l'hypoténuse d'un triangle connaissant les deux côtés de l'angle droit.

Algorithme

Mettre le premier côté dans a
Mettre le deuxième côté dans b
Calculer $\sqrt{a^2+b^2}$, le mettre dans c
Afficher c

Console Scilab

- Calcul et affichage simple

```
-->a=3; b=4;
-->c=sqrt(a^2+b^2)
 c  =
    5.
```

- Demande de valeur et affichage mis en forme

```
-->a = input("Premier côté : ");
Premier côté : 1
-->b = input("Deuxième côté : ");
Deuxième côté : sqrt(3)
-->c = sqrt(a^2+b^2);
-->afficher("Hypoténuse : "+string(c))
 Hypoténuse : 2
```

- Fonction

```
-->function c=Hypotenuse(a,b)
-->c=sqrt(a^2+b^2)
-->endfunction
-->Hypotenuse(4,5)
 ans  =
    6.4031242374328
```

À noter
Le nom d'une fonction ne doit pas contenir d'accent.

BOUCLES

Exemple 4

Calculez les carrés des entiers naturels de 1 à 100.
Les 100 carrés constituent alors les coordonnées du vecteur c.

Algorithme	Éditeur Scilab

Pour n allant de 1 à 100
  c(n) prend la valeur n^2
Fin de pour
	`for n=1:100`
	`  c(n)= n^2;`
	`end`

```
-->c(74)
 ans  =
    5476.
```

Exemple 5

Calculez 30 termes de la suite définie par : $\begin{cases} u_0 = 1 \\ u_{n+1} = u_n + 2n + 3 \end{cases}$

Affichez les indices et les termes de la suite calculés, puis tracez le nuage de points.

Algorithme	Éditeur Scilab

Mettre 4 dans u(1)
(on doit commencer à 1, pas à 0)
Pour n allant de 1 à 30
  Calculer u(n+1) en fonction de n et u(n)
Afficher n en première colonne, u(n) en deuxième
Fin de pour
Effacer la figure, tracer le nuage avec des ronds rouges
	`u(1)=4;`
	`for n=1:30`
	`  u(n+1)=u(n)+2*n+3;`
	`  afficher([n,u(n)])`
	`end`
	`clf; plot(u,"or")`

Exemple 6

Calculez le quotient q d'un nombre positif N par 11, par la méthode des soustractions successives.

Algorithme	Éditeur Scilab

Lire N
Mettre 0 dans q
Tant que N supérieur ou égal à 11
  Mettre N-11 dans N
  Mettre q+1 dans q
Fin de tant que
Afficher q
	`N=input("N = ");`
	`q=0;`
	`while N>=11`
	`  N=N-11;`
	`  q=q+1;`
	`end`
	`afficher("quotient = "+string(q))`

Exemple 7

Bob place 5 000 € à intérêts composés à un taux de 2,7 % par an en 2009.

- Calculez les sommes obtenues pendant les 20 prochaines années
- Tracez le nuage des sommes

Algorithme

Mettre 5 000 dans S(1) qui sera la somme de l'année 2008 + 1
(Par la suite S(n) sera la somme de l'année 2008 + n)
Pour n allant de 1 à 20
  Mettre S(n) multipliée par 1,027 dans S(n+1)
  Afficher l'année et la somme
Fin de pour

Tracé du nuage

- Effacez l'écran graphique
- Tracez le nuage des points (n ; S(n)) avec des croix rouges

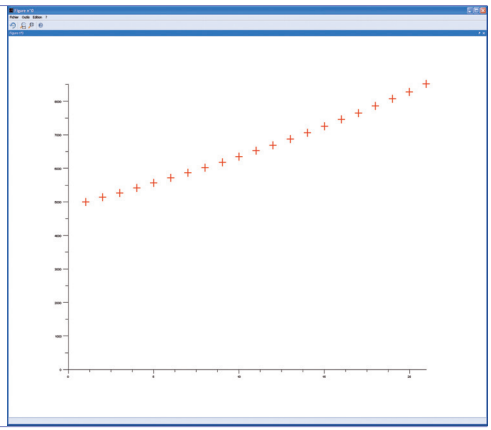
Console Scilab	Fenêtre graphique

```
-->S(1)=5000;
-->for n=1:20
-->S(n+1)=S(n)*1.027;
-->afficher([2008+n,S(n)])
-->end

     2009.    5000.

     2010.    5135.

     2011.    5273.645
etc...
-->clf
-->plot(S,"+r")
```



Écrivez un programme lui permettant de savoir en quelle année il aura 7 000 €

Algorithme	Console Scilab

Mettre la somme 5 000 dans S
Mettre l'année 2009 dans N
Tant que la somme reste inférieure à 7 000,
  La somme est multipliée par 1,027 (elle remplace la précédente)
  L'année augmente de 1 (elle remplace la précédente)
Fin de tant que
Afficher l'année

```
-->S=5000;
-->N=2009;
-->while S<7000
-->S=S*1.027;
-->N=N+1;
-->end

-->afficher("S dépasse 7000 € ..
en : "+string(N))

 S dépasse 7000 € en : 2022
```

Exemple 8

Calculez 40 termes des suites a_n et b_n définies par : $\begin{cases} a_1 = 20 \\ b_1 = 60 \end{cases}$

et pour tout entier naturel n, $\begin{cases} a_{n+1} = \dfrac{2a_n + b_n}{4} \\ b_{n+1} = \dfrac{a_n + 2b_n}{4} \end{cases}$

puis calculez les termes des suites (u_n) et (v_n) définies par : $u_n = a_n + b_n$ et $v_n = b_n + a_n$

Algorithme	Éditeur Scilab

Initialiser a(1) et b(1)
Pour n allant de 1 à 40
  Calculer a(n+1) en fonction de a(n) et b(n)
  Calculer b(n+1) en fonction de a(n) et b(n)
  Afficher n en première colonne, a(n) en deuxième, b(n) en troisième
Fin de pour
Calculer les vecteurs u et v
Afficher u en première colonne, v en deuxième
	`a(1)=20 ; b(1)=60;`
	`for n=1:40`
	`  a(n+1)=(2*a(n)+b(n))/4;`
	`  b(n+1)=(a(n)+2*b(n))/4;`
	`  afficher([n,a(n),b(n)])`
	`end`
	
	`u=a+b; v=b-a;`
	`afficher([u,v])`

On pressent que u et v sont géométriques, vérifiez alors que le quotient de deux termes consécutifs est constant :

```
for n=1:40
  U(n)=u(n+1)/u(n);V(n)=v(n+1)/v(n)
end
afficher([U,V])
```

On remarque, à partir d'un certain rang, l'affichage suivant dû aux valeurs de v qui deviennent trop petites. La machine commence par faire des calculs faux puis écrit **Nan** (« Not a number » en anglais) :

```
0.75    0.25
0.75    0.3
0.75    0.1666666666667
0.75    1.
0.75    0.
0.75   Nan
0.75   Nan
```

Exemple 9

En l'an 2000, le lycée A compte 2 000 élèves et le lycée B compte 8 000 élèves. Une étude montre que, chaque année :

- 10 % des élèves du lycée A quittent leur lycée pour aller au lycée B ;
- 15 % des élèves du lycée B quittent leur lycée pour aller au lycée A.

Au bout de combien de temps le lycée A comptera-t-il plus d'élèves que le lycée B ?

Algorithme	Éditeur Scilab

Mettre 2 000 dans a
Mettre 8 000 dans b
Mettre 2000 dans n
Tant que a<b
  c prend la valeur 0,9* a+ 0,15*b
  b prend la valeur 0,1*a + 0,85*b
  a prend la valeur c
  n prend la valeur n+1
Fin de Tant que
Afficher (A dépasse B en n)
	`a=2000;`
	`b=8000;`
	`n=2000;`
	`while a<b`
	`  c=0.9*a+0.15*b;`
	`  b=0.1*a+0.85*b;`
	`  a=c;`
	`  n=n+1;`
	`end`
	`afficher("A dépasse B en ..`
	`"+string(n))`

Exemple 10

Les suites (u_n) et (v_n) sont définies par :

$$u_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} - \ln n \quad \text{et} \quad v_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} - \ln(n+1)$$

Calculez 200 termes de chaque suite et trouvez à partir de quel rang leur différence est inférieure à 0,01, puis une valeur approchée de la limite commune aux deux suites à 0,01 près.

Algorithme	Éditeur Scilab

Pour n allant de 1 à 200
  Calculer u(n)
  Calculer v(n)
Fin de pour
Trouver n tel que u(n)-v(n) < 0,01
 	
Afficher le rang trouvé et la limite commune
	`for n=1:200`
	`  u(n)=sum([1:n]^(-1))-ln(n);`
	`  v(n)=sum([1:n]^(-1))-ln(n+1);`
	`end`
	`F=find(u-v < 0.01);`
	`afficher("Le rang est : "+string (F(1)))`
	`afficher("Limite : "+string (u(F(1))))`

À noter
- `[1:n]` définit le vecteur des entiers de 1 à n.
- `[1:n]^(-1)` définit le vecteur des inverses des entiers de 1 à n.
- `sum([1:n]^(-1))` donne alors la somme des inverses.

Cet exemple peut être traité plus simplement en affichant uniquement le rang à partir duquel la différence est inférieure à 0,01 et non pas les valeurs des termes des suites.

Algorithme	Éditeur Scilab

Initialiser n à 1 et d à u(1)-v(1)
Tant que d > 0,01
  n augmente de 1
  d est la différence au rang suivant
Fin de tant que
Afficher la réponse
	`n=1; d=ln(2);`
	`while d > 0.01`
	`  n=n+1;`
	`  d=ln((n+1)/n);`
	`end`
	`afficher("Le rang est : "+string (n))`

TESTS

Exemple 11

Calculez la distance entre deux nombres.

Algorithme	Éditeur Scilab

Soit y l'image du couple (a , b) par la fonction d qui calcule la distance.
  Si a supérieur ou égal à b, alors
  Mettre a-b dans y
  Sinon
  Mettre b-a dans y
  Fin de si
Fin de fonction
	`function y=d(a,b)`
	`  if a>=b then`
	`  y= a-b;`
	`  else`
	`  y=b-a;`
	`  end`
	`endfunction`

Distance entre 3 et -5 :

```
-->d(3,-5)
 ans  =
    8.
```

Exemple 12

Virginie lance trois dés numérotés de 1 à 6. Si elle obtient une somme de 18, elle gagne 50 euros, entre 10 et 17, elle gagne 5 euros, sinon elle ne gagne rien.

Algorithme	Éditeur Scilab

Mettre dans T trois nombres entiers tirés au hasard entre 1 et 6
Mettre dans S la somme de ces trois entiers
Si S<10 alors afficher :
  « Virginie ne gagne rien »
Sinon, si S<18 alors afficher :
  « Virginie gagne 5 euros »
Sinon (forcément S=18) afficher :
  « Virginie gagne 50 euros »
Fin de si
	`T=tirage_entier(3,1,6);`
	`S=sum(T)`
	`if S<10 then`
	`  afficher("Virginie ne gagne rien")`
	`elseif S<18 then`
	`  afficher("Virginie gagne 5 euros")`
	`else`
	`  afficher("Virginie gagne 50 euros")`
	`end`

Exemple 13

Étant donné les trois côtés d'un triangle, élaborer un programme qui permet de dire si ce triangle est isocèle, équilatéral, ou quelconque.

Algorithme	Éditeur Scilab

Lire le premier côté, le mettre dans a
Lire le deuxième côté, le mettre dans b
Lire le troisième côté, le mettre dans c
Si a=b et b=c alors
  Afficher « Triangle équilatéral »
Sinon si a=b ou a=c ou b=c alors
  Afficher « Triangle isocèle »
Sinon
  Afficher « Triangle quelconque »
Fin de si
	`a=input("Premier côté = ");`
	`b=input("Deuxième côté = ");`
	`c=input("Troisième côté = ");`
	`if a==b & b==c then`
	`  afficher("Triangle équilatéral")`
	`elseif a==b|a==c|b==c then`
	`  afficher ("Triangle isocèle")`
	`else`
	`  afficher("Triangle quelconque")`
	`end`

Exemple 14

Dichotomie.

Recherche d'une valeur approchée du nombre d'or, solution positive de l'équation $x^2 = x + 1$

- Définissez la fonction $f : f(x) = x^2 - x - 1$
- Tracez la courbe sur l'intervalle [-5 ; 5].

On remarque que la solution recherchée est entre 1 et 2.

- Écrivez un programme permettant par dichotomie d'encadrer la solution recherchée dans un intervalle d'amplitude 10^{-4}, en partant des valeurs 1 et 2.

Algorithme	Éditeur Scilab

Définition de la fonction f	`function y=f(x);`
	`  y=x^2-x-1`
	`endfunction`

Intervalle des valeurs de x	`x=linspace(-5,5,100);`
Tracé de la courbe	`clf; plot(x,f)`
Mettre 1 dans a, mettre 2 dans b	`a=1; b=2;`
Tant que b-a > 10- 4	`while b-a>10^(-4)`
c prend la valeur (a+b)/2	`  c=(a+b)/2;`
si f(a) et f(c) de même signe alors	`  if f(a)*f(c)>0 then`
a prend la valeur c	`    a=c;`
Sinon	`  else`
b prend la valeur c	`    b=c;`
Fin de si	`  end`
Fin de tant que	`end`
Afficher l'encadrement trouvé entre les dernières valeurs de a et b	`afficher("La solution est entre .. "+string(a)+" et "+string(b))`

Exemple 15

La suite de Syracuse est définie par : u_1 est donné, et pour $n \geq 1, u_{n+1} = \begin{cases} \frac{u_n}{2} \ si \ u_n \ est \ pair \\ 3u_n + 1 \ sinon \end{cases}$

Calculez les termes et vérifiez que, quel que soit u_1, on finit toujours par arriver à 4, 2, 1. Pour cela, on définit la fonction **`syracuse`** qui donnera, en fonction du premier terme, le rang auquel on arrive à 1 et les termes de la suite.

Algorithme	Éditeur Scilab
Fonction syracuse	`function [u,n]=syracuse(u1)`
Initialiser n et u(1)	`  u(1)=u1; n=1;`
Tant que u(n) ≠1	`  while u(n)<>1`
Si u(n) est pair, alors	`    if pair(u(n))==%T then`
u(n+1)=u(n)/2	`      u(n+1)=u(n)/2;`
Sinon	`    else`
u(n+1) = 3u(n) +1	`      u(n+1)=3*u(n)+1;`
Fin de si	`    end`
n augmente de 1	`    n=n+1;`
Fin de tant que	`  end`
Afficher n	`  afficher("Nombre de termes = "+string(n))`
Afficher u	`  afficher(u)`
Fin de fonction	`endfunction`
Exemple pour u(1)=10000	`syracuse(10000)`

TRACÉS DE COURBES

Exemple 16

Tracez la courbe de la fonction **cube** entre - 2 et +2.

Commentaires	Éditeur Scilab

Attention à ne pas confondre f qui est la fonction et y qui est l'image de x.
On remarque que y est une variable muette.
On définit les valeurs de x.
clf = *clear figure efface la figure précédente.*
Par défaut, le tracé se fait en bleu.
	`function y=f(x)`
	`  y=x^3`
	`endfunction`
	`x=linspace(-2,2,100);`
	`clf; plot(x,f)`

Tracez en rouge la fonction f définie par $f(x) = x + \ln\left(\frac{x-1}{2x+3}\right)$ pour x entre 1 et 5.
Calculez les images de 2 et de $\sqrt{3}$.

Console Scilab

```
-->function y=f(x); y=x+ln((x-1)/(2*x+3)); endfunction
-->x=linspace(1,5,100);
-->clf
-->plot(x,f,"r")

-->f(2)
 ans  =
    0.0540898509447
-->f(sqrt(3))
 ans  =
  - 0.4461185918724
```

Exemple 17

Tracez les courbes des fonctions f_p définies sur [0 ; 4] par : $f_p(x) = p\sqrt{x^2+9} + 4 - x$ pour $p = 2$, puis pour p variant de 1 à 2 par pas de 0,1, et repérez un minimum éventuel.

Éditeur Scilab

```
function y=f(x); y=p*sqrt(x^2+9)+4-x; endfunction
x=linspace(0,4,100);
p=2;
clf; plot(x,f)
for p=1:0.1:2
  plot(x,f);
end
cliquer
```

À noter

- Si on tape **`cliquer`** dans la console, on clique alors avec la souris sur un point dans la fenêtre graphique et les coordonnées du point s'affichent dans la console.
- Si on tape **`quadrillage`** dans la console, on fait apparaître un quadrillage dans la fenêtre graphique.

Exemple 18

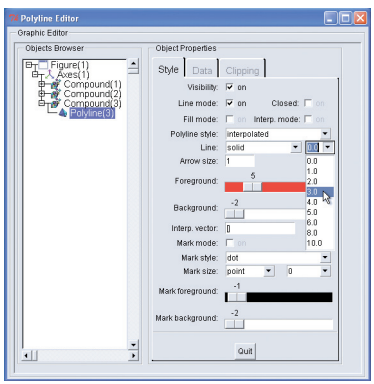
Tracez la représentation graphique de la fonction f définie par : $\begin{cases} f(x) = 0 \ \ si \ x \in [0;2[\\ f(x) = x - 2 \ \ si \ x \in [2;4[\\ f(x) = 2x - 6 \ \ si \ x \in [4;7] \end{cases}$

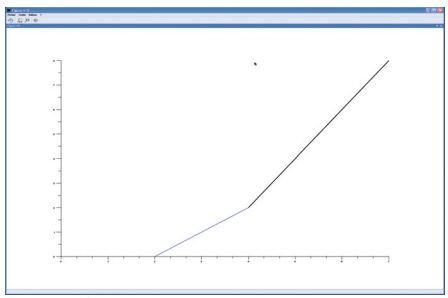
Éditeur Scilab

```
function y=f(x); y=0; endfunction
function y=g(x); y=x-2; endfunction
function y=h(x); y=2*x-6; endfunction
clf
x=linspace(0,2,100); plot(x,f,"r")
x=linspace(2,4,100); plot(x,g)
x=linspace(4,7,100); plot(x,h,"k")
```

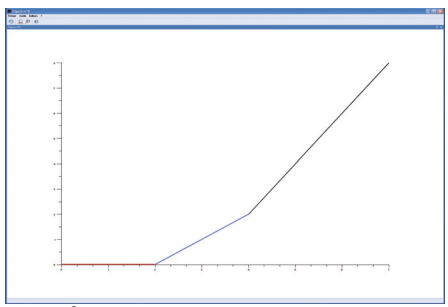
Le tracé n'est pas très lisible, en particulier celui du segment sur l'axe des abscisses.

Cliquez sur **Edition > Propriétés des axes**, une nouvelle fenêtre s'ouvre. Cliquez sur **Compound (3) > Polyline (3)** et choisissez d'augmenter l'épaisseur à 3 (**Line**). Vous remarquerez que les courbes sont numérotées dans l'ordre inverse de celui où elles ont été définies.





avant



après

À noter
Cette option n'est pas encore disponible sur Mac OS.

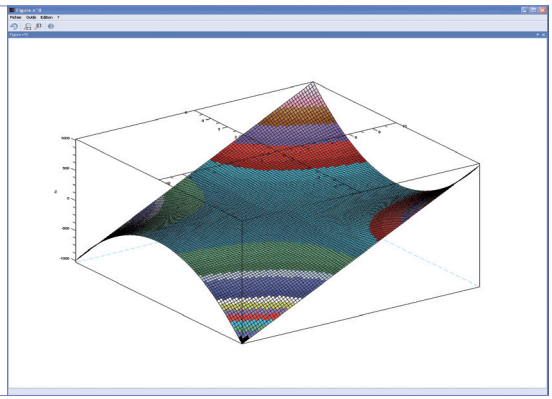
Exemple 19

Tracez la surface définie par : $z = 4\,x\,y^2 + x + 3\,y$, pour x entre -10 et 10 et y entre -5 et 5.

Éditeur Scilab

```
function Z=g(x,y)
  Z=4*x*y^2+x+3*y
endfunction
x=linspace(-10,10,100);
y=linspace(-5,5,100);
z=feval(x,y,g)';
clf; surf(x,y,z)
```

Fenêtre graphique



À noter
On peut faire tourner la figure en cliquant sur l'icône , ou dans la barre de menus sur **Outils > 2D/3D Rotation**.

SIMULATIONS

Exemple 20

Calculez la fréquence d'apparition du 6 lors de la simulation de 10 000 lancers d'un dé à 6 faces numérotées de 1 à 6.

Algorithme	Éditeur Scilab
Simuler 10 000 lancers de dés Mettre les résultats dans S Mettre la fréquence du 6 dans f	`S=tirage_entier(10000,1,6);` `f=frequence(6,S)`

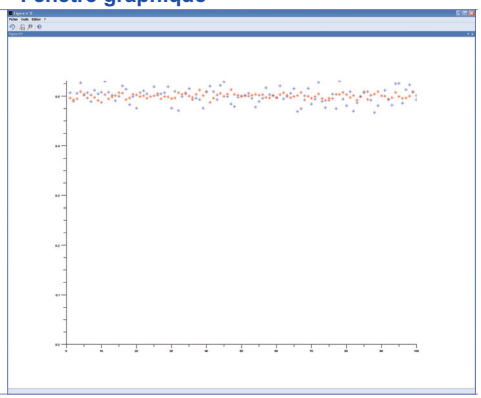
Exemple 21

Simulez le lancer d'une pièce de monnaie sur 100 échantillons de taille 1 000.
Tracez en bleu le nuage des 100 fréquences d'apparition du côté pile.
Simulez avec des échantillons de taille 10 000.
Tracez en rouge le nuage pour comparer les fluctuations.

Éditeur Scilab

```
for n=1:100
  T=tirage_entier(1000,0,1);
  p(n)=frequence(0,T)
end
clf; plot(p,"*")
for n=1:100
  T=tirage_entier(10000,0,1);
  p(n)=frequence(0,T)
end
plot(p,"r*")
```

Fenêtre graphique



Exemple 22

Une puce se déplace sur un axe gradué. À chaque saut, elle se déplace d'une unité, de manière aléatoire et équiprobable vers la droite ou vers la gauche. Elle part de l'origine et effectue une marche de 30 sauts.
Proposez un algorithme donnant la position d'arrivée de la puce.

Algorithme	Éditeur Scilab

Mettre 0 dans x (qui sera l'abscisse de la puce)
Pour i allant de 1 à 30
  On choisit aléatoirement le nombre -1 ou le nombre 1
  x est augmenté du nombre précédent
Fin de pour
Afficher x
	`x=0;`
	`for i=1:30`
	`  x=x+2*tirage_entier(1,0,1)-1;`
	`end`
	`afficher("Position de la puce : "..`
	`+string(x))`

Enrichissez l'algorithme précédent pour donner la liste des positions d'arrivée de N marches aléatoires. Tracez les fréquences des différentes positions.

Algorithme

La fonction f associe à N marches aléatoires les N positions de la puce,
mises dans la liste P
  Pour n allant de 1 à N
    Mettre 0 dans P(n) : au départ la puce est à 0
    Pour i allant de 1 à 30
      On choisit aléatoirement le nombre -1 ou le nombre 1
      P(n) est augmenté du nombre précédent
    Fin de pour
  Fin de pour
Fin de fonction

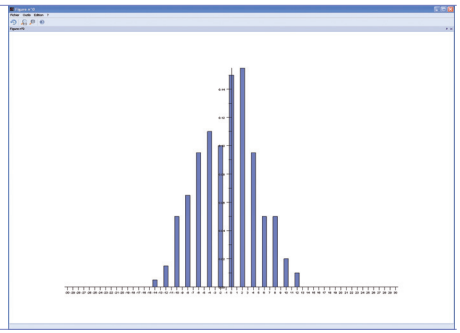
Calcul et tracé des fréquences pour N = 200
Pour i allant de -30 à 30
  Mettre la fréquence de la position i dans fr(i+31) car on doit commencer à 1
Fin de pour
Effacer l'écran
Tracer le diagramme en bâtons de fr

Éditeur Scilab

```
function P=f(N)
  for n=1:N
    P(n)=0;
    for i=1:30
      P(n)=P(n)+2*tirage_entier(1,0,1)-1;
    end
  end
endfunction
```

Calcul et tracé des fréquences pour N = 200

Éditeur Scilab	Fenêtre graphique

`a=f(200);`
`for i=-30:30`
`  fr(i+31)=frequence(i,a);`
`end`
`clf`
`bar([-30:30],fr)`
	

Exemple 23

Élaborez un programme pour approcher la probabilité que dans une même classe de 30 élèves, 2 élèves au moins soient nés le même jour, en faisant 10 000 fois la simulation.
Définissez une fonction associant au nombre d'élèves de la classe la fréquence de cet événement, pour 1 000 simulations. À partir de combien d'élèves cette fréquence est-elle supérieure à 0,5 ?

Algorithme 1	Éditeur Scilab

Mettre 0 dans y (qui donnera la fréquence de l'événement « 2 élèves au moins ont la même date de naissance »)
Pour n allant de 1 à 10 000
  Mettre dans « dates » 30 dates tirées aléatoirement parmi les 365 jours de l'année
  Si au moins 2 dates sont semblables, alors
  Ajouter 1/10000 à y
  Fin de si
Fin de pour
	`y=0;`
	`for n=1:10000`
	`  dates=tirage_entier(30,1,365);`
	`  if taille(unique(dates))<>30 then`
	`  y=y+1/10000;`
	`  end`
	`end`

À noter
La fonction **date** existant déjà dans Scilab, nous choisissons de nommer ici la variable **dates**.

Algorithme 2	Éditeur Scilab

Appelons fr la fonction qui calcule la fréquence de l'événement « 2 élèves au moins ont la même date de naissance », en fonction du nombre e d'élèves.
L'image de e par fr s'appelle y.
Mettre 0 dans y
Pour n allant de 1 à 1 000
Mettre dans « dates » e dates tirées aléatoirement parmi les 365 jours de l'année
Si au moins 2 dates sont semblables, alors
Ajouter 1/1000 à y
Fin de si
Fin de pour
Fin de fonction
Tracer le nuage des points pour e entre 20 et 28, en choisissant 9 points pour avoir les valeurs entières de e. Couleur magenta, style de point triangle.
Faire apparaître un quadrillage
	`function y=fr(e)`
	`  y=0;`
	`  for n=1:1000`
	`  dates=tirage_entier(e,1,365);`
	`  if taille(unique(dates))<>e then`
	`  y=y+1/1000;`
	`  end`
	`  end`
	`endfunction`
	`e=linspace(20,28,9);`
	`clf; plot(e,fr,"m<")`
	`quadrillage`

Exemple 24

J'ai dans ma poche 2 pièces de 10 centimes, 1 pièce de 20 centimes, 3 pièces de 50 centimes, 1 pièce de 1 euro et 2 pièces de 2 euros. Quelle est la probabilité pour qu'en sortant deux pièces au hasard de ma poche, je puisse payer une baguette à 1 euro ?

Algorithme	Éditeur Scilab

Définir l'ensemble P des pièces de la poche
Mettre 0 dans f
Pour k allant de 1 à 1 000
  Tirer deux pièces au hasard, mettre dans t le couple obtenu
  Mettre dans s la somme des valeurs des deux pièces
  Si la somme est >= 1 alors
  f prend la valeur f+1/1000
  Fin de si
Fin de pour
Afficher f
	`P=ensemble("a(0.1)","b(0.1)",..`
	`"c(0.2)","d(0.5)","e(0.5)",..`
	`"f(0.5)","g(1)","h(2)","i(2)");`
	`f=0;`
	`for k=1:1000`
	`  t= tirage_ensemble(2,P);`
	`  s=sum(valeur(t));`
	`  if s >=1 then`
	`  f=f+1/1000;`
	`  end`
	`end`
	`afficher(f)`

Arithmétique

Exemple 25

Déterminez le quotient et le reste dans la division de a par b, a et b positifs.

Algorithme	Éditeur Scilab

Entrer a et b
Mettre 0 dans q
Tant que a >= b alors
  a prend la valeur a – b
  q prend la valeur q+1
Fin de tant que
Afficher q, a
Exemple pour a=1224 et b=15
	`function d=diveucl(a,b)`
	`  q=0;`
	`  while a>=b`
	`  a=a-b;`
	`  q=q+1;`
	`  end`
	`  d=[q a]`
	`endfunction`
	`diveucl(1224,15)`

On définit la fonction **`diveucl`** qui, aux deux variables a et b (a>b>0), associe deux valeurs : le quotient et le reste.

Exemple 26

Déterminez si un nombre entier a est premier, avec a > 2.

On définit la fonction **prem** qui, au nombre entier a, associe la phrase « a est premier » si a est premier et « a est composé » sinon. On regarde à part la divisibilité par 2 qui permet ensuite de ne tester que les diviseurs impairs.

Algorithme	Éditeur Scilab

```
Entrer a
Si 2 divise a alors
   Afficher « a est composé »
   Arrêter le programme
Fin de si
Mettre partie entière de a dans n
Pour d allant de 3 à n de 2 en 2
   Si d divise a alors
      Afficher « a est composé»
      Arrêter le programme
   Fin de si
Fin de pour
Sinon Afficher « a est premier »
Exemple pour a=2^32+1
```

```
function prem(a)
  if reste(a,2) == 0 then
    afficher("a est composé")
    return
  end
  n=floor(sqrt(a));
  for d=3:2:n
    if reste(a,d)==0 then
      afficher ("a est composé")
      return
    end
  end
  afficher("a est premier")
endfunction
prem(2^32+1)
```

À noter

- Il existe dans Scilab la fonction **premier**.

premier(a) retourne **%T** si **a** est premier et **%F** sinon.

- La fonction **return** permet d'arrêter le programme et de sortir de la fonction. On n'exécute donc pas les instructions suivantes, qui sont inutiles.

Exemple 27

Décomposez un entier positif a en facteurs premiers.

On définit la fonction **decomp** qui, au nombre entier positif a, associe la liste de ses diviseurs premiers.

Algorithme

```
Entrer a
Commencer une liste vide
Tant que 2 divise a
   Rajouter 2 à la liste
   a prend la valeur a/2
Fin de tant que
Mettre la partie entière de racine de a
dans n
Pour d allant de 3 à n de 2 en 2
   Tant que d divise a
      Rajouter d à la liste
      a prend la valeur a/d
   Fin de tant que
Fin de pour
Si a≠1 alors
   Rajouter a à la liste
Fin de si
Exemple pour a=4560
```

Éditeur Scilab

```
function y=decomp(a)
  y=[]
  while reste(a,2)==0
    y=[y,2]
    a=a/2;
  end
  n=floor(sqrt(a))
  for d=3:2:n
    while reste(a,d)==0
      y=[y,d];
      a=a/d;
    end
  end
  if a<>1 then
    y=[y,a]
  end
endfunction
decomp(4560)
```

À noter
Il existe dans Scilab la fonction **factorise**. **factorise(a)** donne la liste des diviseurs premiers de **a**.

Exemple 28

Déterminez le plus grand commun diviseur de deux entiers strictement positifs a et b.
On définit la fonction **PGCD** qui, au couple (a,b), associe son plus grand commun diviseur.

Algorithme

```
Entrer a et b
Tant que b non nul alors
   r prend la valeur  du reste de la
division de a par b
   a prend la valeur b
   b prend la valeur r
Fin de tant que
Afficher a (dernier reste non nul)
Exemple pour a=595 et b =10200
```

Éditeur Scilab

```
function d=PGCD(a,b)
  while b<>0
    r=reste(a,b);
    a=b;
    b=r;
  end
  d=a
endfunction
PGCD(595,10200)
```

À noter
La fonction **pgcd** existant déjà dans Scilab, nous choisissons de nommer ici la fonction **PGCD**.

Exemple 29

Déterminez les coefficients de Bézout.
On définit la fonction **BEZOUT** qui, au couple (a,b) avec a et b strictement positifs, associe le couple (u,v) des entiers vérifiant au + bv = pgcd (a,b) obtenus avec l'algorithme d'Euclide.

Algorithme

Nous ne le détaillerons pas. Il est complexe et nécessite l'introduction de variables intermédiaires.

Exemple pour a=595 et b =10200

Éditeur Scilab

```
function [u,v]=BEZOUT(a,b)
  u=1; v=0; U=0; V=1;
  while b<>0;
    q=floor(a/b);
    z=u; u=U; U=z-q*U;
    z=v; v=V; V=z-q*V;
    z=a; a=b; b=z-q*b;
  end
  afficher([u,v])
  afficher("Le pgcd vaut "+string(a))
endfunction
BEZOUT(595,10200)
```

À noter
Là aussi la fonction **bezout** existant déjà dans Scilab, nous choisissons de nommer ici la fonction **BEZOUT**.

Exemple 30

Nombres de Mersenne.
Ils sont de la forme $n = 2^p - 1$. Sachant que p est premier, déterminez les nombres de Mersenne qui sont premiers et ceux qui ne le sont pas, pour $p < 50$.

Algorithme	Éditeur Scilab

Pour p allant de 2 à 40
  Si p est premier alors
  n prend la valeur correspondante
  Afficher (p,n, les facteurs de n)
  Fin de si
Fin de pour
	`for p=2:40`
	`  if premier(p)==%T then`
	`  n=2^p-1 ;`
	`  afficher([p,n,factorise(n)])`
	`  end`
	`end`

Exemple 31

Recherchez les nombres parfaits et les nombres amicaux.
Un nombre parfait est un entier naturel égal à la somme de tous ses diviseurs entiers positifs sauf lui-même. Deux nombres amicaux sont chacun égal à la somme des diviseurs entiers positifs de l'autre (sauf lui-même).

Algorithme	Éditeur Scilab

Nombres parfaits
Pour n allant de 1 à 10000
  Si la somme des diviseurs de n vaut 2n alors
  Afficher (n)
  Fin de si
Fin de pour
	`//Nombres parfaits`
	`for n=1:10000`
	`  if sum(diviseurs(n))==2*n then`
	`  afficher(n)`
	`  end`
	`end`

Algorithme	Éditeur Scilab

Nombres amicaux
Pour n allant de 2 à 10000
  s prend la valeur (somme des diviseurs de n)-n
  t prend la valeur (somme des diviseurs de s)-s
  Si t = n alors
  Afficher ([n s])
  Fin de si
Fin de pour
	`//Nombres amicaux`
	`for n=2:10000`
	`  s=sum(diviseurs(n))-n;`
	`  t=sum(diviseurs(s))-s;`
	`  if t==n then`
	`  afficher([n,s])`
	`  end`
	`end`

Exemple 32

Pour tout entier naturel non nul n, on considère les deux nombres entiers $N = 3n^2 - n + 1$ et $D = 2n - 1$.
Le but de l'exercice consiste à déterminer, suivant les valeurs de n, le reste de la division de N par D.
Déterminez les valeurs de ce reste en fonction de n, pour n entre 1 et 50.
Représentez graphiquement ce reste en fonction de n.

Algorithme	Éditeur Scilab

Pour n allant de 1 à 50
  Calculer N(n)
  Calculer D(n)
  Calculer le reste r(n) de la division de N par D
  Afficher n et r(n) en colonnes
Fin de pour
Effacer la figure
Tracer le nuage des points (n ; r(n)) avec des croix
	`for n=1:50`
	`  N(n)=3*n^2-n+1;`
	`  D(n)=2*n-1;`
	`  r(n)=reste(N(n),D(n));`
	`  afficher([n,r(n)])`
	`end`
	`clf`
	`plot(r,"+")`

Divers

Exemple 33

Simulez le jeu du lièvre et de la tortue.
On lance un dé équilibré. Si on obtient le 6, le lièvre a gagné, et une nouvelle partie commence. Sinon, la tortue avance et on relance le dé. La tortue doit avancer 5 fois pour gagner (5 tirages consécutifs sans le 6). On cherche à simuler un grand nombre de parties pour approcher la probabilité de gagner de chacun.

Algorithme	Éditeur Scilab

L représentera le nombre de parties gagnées par le lièvre. Initialiser L à 0.
T représentera le nombre de parties gagnées par la tortue. Initialiser T à 0.
Pour P allant de 1 à 5000
n compte le nombre de lancers de dés, Initialiser n à 0.
Début de boucle sans fin
On lance le dé, la valeur est mise dans d
Si d=6 alors
Le lièvre a gagné, on rajoute 1 à L, on sort de la boucle
Sinon on rajoute 1 à n
Fin de si
Si n atteint 5 alors
La tortue a gagné, on rajoute 1 à T, on sort de la boucle
Fin de si
Fin de la boucle, on recommence une partie
Fin de pour
Afficher les fréquences de parties gagnées par chacun
	`L=0`
	`T=0`
	`for P=1:5000`
	`  n=0;`
	`  while %T`
	`  d=tirage_entier(1,1,6);`
	`  if d==6 then`
	`  L=L+1;`
	`  break;`
	`  else`
	`  n=n+1;`
	`  end`
	`  if n==5 then`
	`  T=T+1;`
	`  break;`
	`  end`
	`  end`
	`end`
	`afficher([L/(L+T),T/(L+T)])`

À noter
`while %T` réalise une boucle sans fin.
La commande **`break`** permet de sortir de la boucle **`while`**. On la met ici dès que la partie est finie.

Exemple 34

Résolvez l'équation différentielle $\begin{cases} y'(x) = y(x) \\ y(0) = 1 \end{cases}$ par la méthode d'Euler et comparez le résultat obtenu à la solution exacte $y(x) = e^x$. La suite obtenue par la méthode d'Euler est donnée par : $\begin{cases} y_1 = 1 \\ y_i = y_{i-1} \end{cases} + dx\, y_{i-1}$ où dx est le pas en x.

Algorithme	Éditeur Scilab

On définit la fonction f : fonction exponentielle
La fonction euler donnera l'approximation de la fonction exponentielle par la méthode d'Euler sur l'intervalle [0 ; b], avec un pas appelé dx.
Mettre la partie entière de b/dx dans n
x prend les valeurs 0, dx, 2dx,…, (n-1)dx
Mettre 1 dans y(1)
Pour i allant de 2 à n
Calculer y(i) en fonction de y(i-1)
Fin de pour
Effacer l'écran, tracer le nuage des y (en bleu relié par défaut) et la courbe de f (en rouge)
Fin de fonction
Application entre 0 et 5 pour un pas de 0,1
	`function y=f(x)`
	`  y=exp(x)`
	`endfunction`
	`function y=euler(b,dx)`
	`  n=floor(b/dx);`
	`  x=0:dx:(n-1)*dx;`
	`  y(1)=1;`
	`  for i=2:n`
	`  y(i)=y(i-1)+dx*y(i-1);`
	`  end`
	`  clf; plot(x,y,x,f,"r")`
	`endfunction`
	`euler(5,0.1)`

Exemple 35

Approximation d'une aire.

Étant donné une fonction f positive et monotone sur [a ; b], on veut encadrer l'intégrale de f entre a et b par les sommes des aires des rectangles.

Application à la fonction racine carrée entre 0 et 10 : $\begin{cases} s_n = \dfrac{b-a}{n} \displaystyle\sum_{k=0}^{n-1} f(a + k\dfrac{b-a}{n}) \\ S_n = \dfrac{b-a}{n} \displaystyle\sum_{k=1}^{n} f(a + k\dfrac{b-a}{n}) \end{cases}$

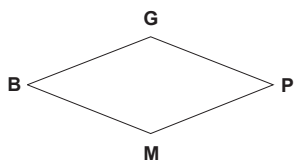
Algorithme	Éditeur Scilab

La fonction sommes retournera les deux sommes.
Initialiser S et s à 0
Pour k allant de 0 à n-1
s augmente de l'image de la borne de gauche
S augmente de l'image de la borne de droite
Fin de pour
S et s sont multipliés par (b-a)/n
Fin de fonction
Définition de la fonction racine
Affichage des deux sommes sur l'intervalle [0 ; 10] avec 10 rectangles
	`function [s,S]=sommes(a,b,n)`
	`  S=0; s=0;`
	`  for k=0:n-1`
	`  s=s+f(a+k*(b-a)/n);`
	`  S=S+f(a+(k+1)*(b-a)/n);`
	`  end`
	`  s=s*(b-a)/n; S=S*(b-a)/n;`
	`endfunction`
	`function y=f(x)`
	`  y=sqrt(x)`
	`endfunction`
	`[s,S]=sommes(0,1,10)`

Exemple 36

Dans une ville, des trains roulant dans les deux sens relient la gare (G), la plage (P), le musée (M) et le belvédère (B) selon le graphe suivant :

- Donnez la matrice A associée au graphe GPMB.



- Calculez A^6 et déduisez combien de chaînes de longueur 6 commencent et se terminent à la gare.

```
-->A=[0 1 0 1;1 0 1 0;0 1 0 1;1 0 1 0]
 A  =
    0.    1.    0.    1.
    1.    0.    1.    0.
    0.    1.    0.    1.
    1.    0.    1.    0.
```

```
-->B=A^6
 B  =
    32.     0.      32.     0.
    0.      32.     0.      32.
    32.     0.      32.     0.
    0.      32.     0.      32.
-->afficher ("Il y a "+string(B(1,1))+" chaînes de ..
longueur 6 commençant et se terminant à la gare.")
 Il y a 32 chaînes de longueur 6 commençant et se
terminant à la gare.
```

- On donne $P = \begin{bmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 1 \\ -1 & 0 & 1 & -1 \\ 0 & -1 & 1 & 1 \end{bmatrix}$. Calculez D = $P^{-1}AP$

```
-->P=[1 0 1 -1;0 1 1 1;-1 0 1 -1;0 -1 1 1]
 P  =
    1.     0.     1.   - 1.
    0.     1.     1.     1.
  - 1.     0.     1.   - 1.
    0.   - 1.     1.     1.
-->P^(-1)*A*P
 ans  =
    0.     0.     0.     0.
    0.     0.     0.     0.
    0.     0.     2.     0.
    0.     0.     0.   - 2.
```

Exemple 37

n personnes numérotées de 1 à n jouent.
Le meneur de jeu donne un ticket à la première, puis saute 1, donne un ticket à la troisième, puis saute 2, donne un ticket à la sixième, etc, et recommence en tournant et en sautant une personne de plus à chaque fois. Si la personne a déjà un ticket, on ne lui en redonne pas.

- Pour quelles valeurs de n tous les joueurs auront-ils un ticket ?
- Si le meneur de jeu fait payer chacun 1 euro et donne 2 euros à toutes les personnes ayant un ticket, pour quelles valeurs de n est-il gagnant ?

Les réponses à ces questions ne sont pas évidentes du tout. On peut prouver qu'au bout de 2n distributions de tickets, on boucle sur les mêmes personnes. On définira une fonction qui au nombre n de joueurs fait correspondre le vecteur u à n coordonnées égales à 1 si le joueur correspondant a un ticket, à 0 sinon.

Éditeur Scilab

```
function u=f(n)
  i=1;
  k=1;
  u=zeros(1,n);
  u(1)=1;
  for k=1:2*n
    j=reste(i+k,n)+1;
    u(j)=1;
    if sum(u)==n then
      break
    end
    i=j;
  end
endfunction
```

- Cherchons dans quel cas tous les joueurs auront un ticket :

```
for a=1:500
  if sum(f(a))==a then
    afficher(a)
  end
end
```

- Le meneur de jeu prend 1 euro par personne et donne 2 euros à ceux qui ont un ticket. Définissions la fonction **gain** de ce qu'il gagne :

```
function y=gain(x);
  y=x-2*sum(f(x));
endfunction
x=linspace(1,50,50);
clf
quadrillage
plot(x,gain,"*")
```

Chapitre 4
Fonctions Scilab utiles au lycée

Les fonctions en noir sont spécifiques au module lycée.

À noter
Il n'y a jamais d'accent dans le nom d'une fonction Scilab.

ANALYSE

- `sqrt(x)` retourne la racine carrée de x pour x réel positif ou nul, et la racine complexe de partie réelle positive sinon.
- `ln(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 $n - 1 < x \leq n$.

ARITHMÉTIQUE

- `pair(n)` retourne `%T` si n est pair et `%F` sinon avec n entier positif ou nul (`%T` signifie True c'est-à-dire vrai et `%F` signifie False c'est-à-dire faux).
- `impair(n)` retourne `%T` si n est impair et `%F` sinon avec n entier positif ou nul.
- `quotient(m,n)` retourne le quotient de m par n avec m entier et n entier non nul.
- `reste(m,n)` retourne le reste de la division de m par n avec m entier et n entier non nul.
- `pgcd(m,n)` retourne le plus grand commun diviseur de m et n avec m et n entiers.
- `ppcm(m,n)` retourne le plus petit commun multiple de m et n avec m et n entiers.
- `premier(n)` retourne `%T` si n est premier et `%F` sinon avec n entier positif ou nul.
- `liste_premiers(n)` retourne la suite des nombres premiers inférieurs à n avec n entier positif ou nul.
- `diviseurs(n)` retourne la suite des diviseurs du nombre n avec n entier positif.
- `factorise(n)` retourne la suite des facteurs premiers de n avec n entier positif ou nul.
- `change_base(m,b1,b2)` transforme le nombre m écrit en base `b1` sous forme de chaîne de caractères en un nombre écrit en base `b2` lui aussi sous forme de chaîne de caractères.

PROBABILITÉS ET STATISTIQUES

- **`factorielle(n)`** retourne la factorielle de n avec n entier positif ou nul.
- **`arrangement(n,p)`** retourne le nombre d'arrangements de p éléments pris parmi n avec n et p entier positifs ou nuls et $p \leq n$.
- **`combinaison(n,p)`** retourne le nombre de combinaisons de p éléments pris parmi n avec n et p entier positifs ou nuls et $p \leq n$.
- **`tirage_entier(p,m,n)`** retourne un vecteur de p tirages entiers pris entre m et n avec p entier positif, m et n entiers et $m \leq n$.
- **`tirage_reel(p,a,b)`** retourne un vecteur de p tirages réels pris entre a et b avec p entier positif, a et b réels et $a \leq b$.
- **`frequence(n,s)`** retourne la fréquence de n dans la suite de nombres s avec n entier.
- **`frequence_tirage_entier(p,m,n)`** retourne la suite des fréquences de p tirages entiers pris entre m et n avec p entier positif, m et n entiers et $m \leq n$.
- **`moyenne(v)`** retourne la moyenne du vecteur de nombres v.
- **`moyenne_ponderee(v,n)`** retourne la moyenne du vecteur de nombres v pondéré par le vecteur de nombres n avec v et n de même dimension et n un vecteur de nombres positifs ou nuls mais non tous nuls.
- **`ecart_type(v)`** retourne l'écart type du vecteur de nombres v.
- **`ecart_type_pondere(v,n)`** retourne l'écart type du vecteur de nombres v pondéré par le vecteur de nombres n avec v et n de même dimension et n un vecteur de nombres positifs ou nuls mais non tous nuls.
- **`variance(v)`** retourne la variance du vecteur de nombres v.
- **`variance_ponderee(v,n)`** retourne la variance du vecteur de nombres v pondéré par le vecteur de nombres n avec v et n de même dimension et n un vecteur de nombres positifs ou nuls mais non tous nuls.
- **`mediane(v)`** retourne la médiane du vecteur de nombres v.
- **`mediane_ponderee(v,n)`** retourne la médiane du vecteur de nombres v pondéré par le vecteur de nombres n avec v et n de même dimension et n un vecteur de nombres positifs ou nuls mais non tous nuls.
- **`quartiles(v)`** retourne les deux quartiles du vecteur de nombres v.
- **`quartiles_ponderes(v,n)`** retourne les deux quartiles du vecteur de nombres v pondéré par le vecteur de nombres n avec v et n de même dimension et n un vecteur de nombres positifs ou nuls mais non tous nuls.
- **`deciles(v)`** retourne les déciles D1 et D9 du vecteur de nombres v.

- **deciles_ponderes(v,n)** retourne les déciles D1 et D9 du vecteur de nombres v pondéré par le vecteur de nombres n avec v et n de même dimension et n un vecteur de nombres positifs ou nuls mais non tous nuls.
- **loi_exp(lambda,t)** retourne la probabilité $p(X \leq t)$ lorsque X suit la loi exponentielle de paramètre λ avec λ positif.
- **loi_normale(t,xbar,sigma)** retourne la probabilité $p(X \leq t)$ lorsque X suit la loi normale de paramètres $\bar{x}$ et σ avec σ positif.
- **regression_y_en_x(x,y)** retourne les coefficients a et b de la droite de régression de y en x par la méthode des moindres carrés, d'équation $y = ax + b$, avec x et y des vecteurs de nombres de même dimension et x un vecteur de nombres non tous égaux.
- **histogramme(a,n,couleur)** permet de tracer l'histogramme d'une série statistique où les valeurs de la variable sont regroupées dans des intervalles. a est le vecteur donnant les bornes des intervalles dans l'ordre croissant. n est le vecteur des effectifs ou des fréquences correspondants. L'argument optionnel **couleur** définit la couleur comme dans la fonction **plot**.

ENSEMBLES

- **ensemble("r(1)","r(2)","r(3)","v(1)","v(2)")** définit un ensemble, ici l'ensemble de trois boules rouges numérotées 1, 2, 3 (leurs valeurs) et deux vertes numérotées 1, 2 (leurs valeurs).
- **tirage_ensemble(n,ens)** retourne un ensemble de n éléments pris parmi ceux de l'ensemble **ens**.
- **valeur(ens)** retourne le vecteur des valeurs des éléments de l'ensemble **ens**.
- **ajouter("a",ens)** ajoute un élément **a** à l'ensemble **ens**.
- **appartient("a",ens)** détermine l'appartenance d'un élément **a** à l'ensemble **ens.**
- **complementaire(A,B)** retourne le complémentaire de l'ensemble **A** dans l'ensemble **B**.
- **enlever("a",ens)** enlève l'élément **a** de l'ensemble **ens**.
- **inclus(A,B)** détermine si un ensemble **A** est inclus dans un ensemble **B**.
- **intersection (A,B)** retourne l'intersection de deux ensembles **A** et **B**.
- **jeu_32**, **jeu_52**, **jeu_54**, **jeu_tarot** retournent les ensembles des cartes respectivement de jeux de 32, 52, 54 cartes ou d'un jeu de tarot.
- **union(A,B)** retourne l'union des deux ensembles **A** et **B**.

GRAPHIQUE

- `plot` permet de tracer des courbes ou des nuages de points en dimension 2.
- `linspace(a,b,n)`, avec a et b réels, $a < b$ et n entier, définit un vecteur de n valeurs régulièrement espacées entre a et b.
- `clf` permet d'effacer la fenêtre graphique.
- `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.
- **`orthonorme`** bascule en une représentation orthonormée ou non orthonormée du tracé de la fenêtre graphique.
- **`quadrillage`** affiche ou masque le quadrillage du tracé de la fenêtre graphique.
- **`cliquer`** retourne les coordonnées du point cliqué sur la fenêtre graphique.

UTILITAIRES

- **`taille(v)`** retourne la taille du vecteur ou de la matrice v.
- **`trier(v)`** ou **`trier(v,">")`** retourne le vecteur de nombres ou de chaînes de caractères v ordonné dans l'ordre croissant.
- **`trier(v,"<")`** retourne le vecteur de nombres ou de chaînes de caractères v ordonné dans l'ordre décroissant.
- `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.
- **`afficher(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.

www.scilab.org