Commandes du programme officiel (voie S)

Extrait du programme officiel en Voie Scientifique en première année :

Les constantes, variables, opérations, fonctions et commandes spécifiées dans le programme de première année sont les suivantes :

Commandes	Remarques
%pi %e	constantes pré-définies
Affectation : nom = expression	L'expression peut être du type numérique, matricielle ou du type chaîne de caractères
// commentaire	permet de commenter une commande
Vecteurs lignes : [, ,…,] Vecteurs colonnes : [; ;…;] Matrices : [,…, ;…; ,…,]	Constructions de vecteurs et de matrices numériques
+ - * / ^	Opérations arithmétiques (s'appliquent à des variables numériques ou matricielles)
== > < >= <= <>	Comparaisons - tests (s'appliquent à des variables numériques ou matricielles)
& and | or	Opérations logiques : et ou (variables numériques ou matricielles)
log, exp, floor, abs, sqrt, sin, cos	Fonctions numériques usuelles, toutes ces fonctions peuvent s'appliquer à des variables numériques ou à des matrices élément par élément
Fonction rand	La fonction `grand` pourra être utilisée avec les paramètres correspondant aux lois de probabilité présentes dans le programme
rank(A), inv(A), A'	Fonctions matricielles Extraction ou modification d'un élément, d'une ligne ou d'une colonne d'une matrice
size(A), find	On pourra utiliser les fonctions dans le cadre de simulations
plot, plot2d, bar, histplot	Courbes représentatives de fonctions usuelles, de densités et de fonctions de répartition. Tracé d'histogrammes.
linspace(a,b,n) ./ .* .^	Commande linspace opérations pointées
if …then …end if …then …else …end	Structure conditionnelle
while …then …end for k=…: :…end	Structures répétitives
function y = nom(...,...) endfunction	Fonctions-arguments-retour de résultats
input()	Fonction d'entrée des données
disp()	Fonction de sortie de résultat(s)

Extrait du programme officiel en Voie Scientifique en deuxième année :

Toutes les commandes du programme de première année sont exigibles.
Les seules nouvelles commandes exigibles des candidats sont indiquées dans ce paragraphe.

La connaissance des commandes suivantes ainsi que de leurs arguments est exigible des candidats :

`sum`, `cumsum`, `mean`, `min`, `max`, `zeros`, `ones`, `eye`, `spec`.

Les commandes suivantes devront avoir été manipulées par les étudiants mais la connaissance détaillée de leurs arguments n'est pas exigible des candidats :

`cdfnor`, `plot2d`, `fplot2d`, `plot3d`, `fplot3d`.

Dans la liste des différents thèmes, figurent les commentaires suivants :

On pourra également utiliser les commandes :

`dsearch`, `tabul`, `pie`, `stdeviation`, `median` (statistiques descriptives univariées)

On pourra utiliser les commandes :

`stdeviation`, `corr` (statistiques descriptives bivariées)

Simulation et mise en évidence d'états stables avec la commande :

`grand(n, 'markov', M, x0)` (chaînes de Markov).

Dans les thèmes "simulations de lois" et "estimation ponctuelle ou par intervalle de confiance" :

utilisation du générateur `grand`.

Commentaires généraux sur le programme :

Dans certains thèmes, il s'avérera nécessaire d'introduire de nouvelles notions ou approches mathématiques.
Celles-ci devront être explicitées en préambule des séances d'informatique et ne pourront en aucun cas être exigibles des étudiants. Certaines seront propres à un thème particulier, d'autres (comme par exemple les méthodes de Monte-Carlo) pourront au contraire être envisagées de manière transversale. Toutes les précisions nécessaires devront toujours être données lors de leur utilisation.

Toute la richesse du logiciel Scilab ne peut pas être entièrement maîtrisée par un étudiant, aussi seules les fonctions et commandes introduites en première année et celles figurant dans la sous-partie "Commandes exigibles" sont exigibles. Néanmoins, se contenter de ces seules commandes, en ignorant les nombreuses possibilités et commodités du logiciel, se révélerait rapidement contraignant et limitatif. De nouvelles commandes Scilab peuvent donc être introduites, mais cela devra se faire avec parcimonie, l'objectif principal de l'activité informatique reste la mise en pratique des connaissances mathématiques. Les commandes introduites devront être présentées en préambule et toutes les précisions nécessaires seront données lors de leur utilisation et leur interprétation.

On favorisera à cette occasion l'autonomie et la prise d'initiatives des étudiants grâce à l'utilisation de l'aide de Scilab, et à l'usage d'opérations de "copier-coller" qui permettent de prendre en main rapidement des fonctions nouvelles et évitent d'avoir à connaître par cœur la syntaxe de commandes complexes.

L'objectif de ces travaux pratiques n'est pas l'écriture de longs programmes mais l'assimilation de savoir-faire et de compétences spécifiés dans la liste des exigibles rappelés en préambule de chaque thème.

Les exemples traités dans un thème devront être tirés, autant que possible, de situations réelles (traitement de données économiques, sociologiques, historiques, démographiques, en lien avec le monde de l'entreprise ou de la finance, etc), en faisant dès que possible un rapprochement avec les autres disciplines.

Commandes générales - boucles - fonctions plot et rand (1)

Commandes	Scilab	Remarques
clear console clear functions	clc clf	efface la console efface les fonctions précédentes
Affichage à l'écran d'une chaîne de caractères	disp('Bonjour '); 	
Affectation =	x = 3	affecte la valeur 3 à la variable x
Affectation et écriture du résultat à l'écran	x = 3 disp(x, 'Valeur de x : ');	Dans la console, l'affichage est le suivant : Valeur de x : 3
Message à l'écran puis lecture de la valeur de x	x = input('x : ');	Permet à l'utilisateur de fournir une valeur
Insertion d'un commentaire	// commentaire	Le commentaire est introduit à droite
Fonctions usuelles et Pi	abs, sqrt, log, exp, cos, sin, floor, %pi	Valeur absolue, racine carrée, **log** désigne le **logarithme népérien**, floor designe la partie entière
Passage à la ligne dans Scinotes	..	Lorsque la ligne de commande est trop longue dans Scinotes, ".." permet d'écrire sur plusieurs lignes
Opérations élémentaires	+ – * / ^	Addition, soustraction, multiplication, divisions opération puissance (^)
Comparaison et tests	== < > <= >= <>	Utilisé dans les instructions conditionnelles
Opérateurs logiques	& |	& pour "et", | pour "ou"

Instruction if ... then	if (x = = 0) then
	s=s+1;
	end
		Si *x* est égal à 0 alors
		on rajoute 1 à *s*

Instruction if ... then ... else	if (x = = 0) then
	s=s+1;
	else
	s=s-1;
	end
		Si *x* est égal à 0 (test)
		alors on incrémente
		s de 1, sinon
		on décrémente *s* de 1

Instruction if ... then ... elseif ... then ... else ... end	x=grand(1,50,"uin",1,6) if x==1 then U = U+1 elseif x<5 then D = D+1 else T = T+1 end	Simulation de 50 tirages avec remise dans une urne contenant des boules portant des numéros égaux à 1, 2, ou 3 dans les proportions $\frac{1}{6}$, $\frac{1}{2}$ et $\frac{1}{3}$

Commandes générales - boucles - fonctions plot et rand (2)

Commandes	Scilab	Remarques
Boucle for	u = .5; a = 1 for i = 2 : n u = a * u * (1-u) end	Calcul du $n^{ème}$ terme de la suite (u_n) définie par $u_1 = \frac{1}{2}$ et $u_{n+1} = au_n(1-u_n)$ si $n \geq 1$

Boucle while	r = 0; p = 0.3
	while (r = = 0)
	u = rand();
	n = n+1;
	if (u < p) then
	r = 1;
	end
	end
	disp(n, 'n : ');
		Simulation d'une variable
		aléatoire suivant la loi
		géométrique de paramètre $p = 0.3$,
		r désigne le résultat du lancer,
		0 désignant un échec
		u suit la loi uniforme sur $[0,1]$
		n est le rang du premier
		succès, l'événement $[u < p]$
		se produit avec une probabilité p

Déclaration générale d'une fonction	function z = f(x, y) z = x^2 + 2*y^2 endfunction	Fonction qui calcule $x^2 + 2y^2$ en fonction de x et de y
Tracé de la courbe représentative d'une fonction avec plot	x=[-2:0.1:4]; plot(x, x^2+1)	Tracé de la courbe représentative de $x \mapsto x^2 + 1$, pour $x \in [-2,4]$ 0.1 correspond au pas choisi
Tracé de deux courbes représentatives	x=[-2:0.1:4]; plot(x, x^2 + 1 , 'red' , x, 2*x, 'black')	On trace sur un même graphique les courbes de $x \mapsto x^2 + 1$ et $x \mapsto 2x$. x=[-2:0.1:4] peut être remplacé par x=linspace(-2,4,60)
Loi uniforme sur $[0,1]$	U = rand()	Simulation d'une seule variable La loi par défaut est la loi uniforme sur $[0,1]$
Loi uniforme sur $[0,1]$ matrice aléatoire	r = rand(4, 6, "uniform")	r est une matrice à 4 lignes et 6 colonnes, composée de nombres aléatoires de distribution uniforme
Loi normale centrée réduite	x = rand(1,3, "normal")	r est une matrice à 1 lignes et 3 colonnes

Remarque : Les fonctions plot et rand proviennent d'anciennes versions de Scilab. Elles figurent au programme mais sont limitées, on utilisera plutôt plot2d et fplot2d, ainsi que grand, qui présentent des fonctionnalités plus riches (cf. pages suivantes).

Calcul matriciel (1)

Commande	Scilab	Remarques
Tableau à une ligne ou vecteur ligne	`A=0:2:10;` A=linspace(0,10,6)	Création de $A = \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 \end{pmatrix}$ Le vecteur obtenu comporte 6 valeurs Le segment [0,10] est divisé en 5 sous-segments
Matrice quelconque	A=[2, 3, 0;1, 4,-1];	Création de $A = \begin{pmatrix} 2 & 3 & 0 \\ 1 & 4 & -1 \end{pmatrix}$
Matrices particulières Initialisation	M=zeros(n,p); M=ones(n,p); M=eye(n,n);	$M = (0)_{1\leq i\leq n, 1\leq j\leq p}$ $M = (1)_{1\leq i\leq n, 1\leq j\leq p}$ $M = I_n$
Longueur d'un vecteur ligne ou colonne	Longueur = length(L)	L est un vecteur ligne ou colonne Longueur est un entier
Taille de la matrice quelconque	[nb_ligne,nb_col]=size(A)	size renvoie une matrice ligne comportant deux coefficients Dans l'exemple ci-dessus on aurait : [nb_ligne,nb_col]=[2, 3]
Coefficient d'une matrice	A(i,j) = 2	On affecte la valeur 2 au coefficient de la matrice A situé à la $i^{ème}$ ligne et à la $j^{ème}$ colonne
Extraction d'une ligne Extraction d'une colonne	L=A(2,1:3) ou L=A(2,:) C=A(1:2,3) ou C=A(:,3)	Extraction de la deuxième ligne de la matrice A, on obtient : $L = \begin{pmatrix} 1 & 4 & -1 \end{pmatrix}$ Extraction de la troisième colonne de A. : désigne toutes les lignes (ou toutes les colonnes)
Extraction - exemple	v=[1,3] B=A(:,v)	Extraction des colonnes 1 et 3 de la matrice *A*, le résultat est : $B = \begin{pmatrix} 2 & 0 \\ 1 & -1 \end{pmatrix}$
Extraction - cas général $A \in \mathcal{M}_{n,m}(\mathbb{R})$ $B \in \mathcal{M}_{k,p}(\mathbb{R})$	$u=[u_1,u_2,\ldots,u_k]$ $v=[v_1,v_2,\ldots,v_p]$ B=A(u,v)	La matrice B est constituée des éléments de *A* situés aux lignes $u_1,u_2,\ldots,u_k$ et aux colonnes $v_1,v_2,\ldots,v_p$.
Concaténation horizontale Concaténation verticale	C=[A,B] E=[F;G]	Si nb_lignes_A = nb_lignes_B Si nb_colonnes_A = nb_colonnes_B
Tri des éléments	gsort(A,'r','i')	Tri sur les lignes ('r' pour "row"), par ordre croissant ("increase")
Opérations Opérations pointées	+, −, ∗, /, ^ .+, .−, .∗, ./, .^	Opérations usuelles, ^ désigne la puissance Opérations coefficient par coefficient
Fonctions matricielles	rank(M), inv(M), M'	rang, inverse, transposée (adjointe)

Calcul matriciel (2)

Commande	Scilab	Remarques
Utilisation d'une fonction pré-définie	B = f(A)	Si $A = (a_{i,j})$ alors $B = (f(a_{i,j}))$ où f est une fonction numérique comme $\cos, \exp, \log$ etc.
Utilisation d'une fonction qu'on a soi-même créé	B = feval(A, f)	Si $A = (a_{i,j})$ alors $B = (f(a_{i,j}))$ où f est une fonction définie par l'utilisateur
Résolution d'un système	X=A\B (division à gauche)	Résolution de $AX = B$
Commande find (matrice ligne) Renvoie les numéros des éléments vérifiant la condition	X = grand(1, 100, "exp", 1); n = length(find(X >= 2)	Echantillon de 100 valeurs prises par une variable aléatoire suivant une loi exponentielle d'espérance 1, on compte le nombre de fois où l'on a obtenu une valeur supérieure à 2
Commande find matrice quelconque	A=rand(2,3) x=find(A < 0.5)	Renvoie les numéros (cf. (*)) des éléments vérifiant la condition
Valeurs propres	val_propres=spec(A)	val_propres est un vecteur colonne dont les coefficients sont les valeurs propres de A
Diagonalisation d'une matrice carrée	[vect_p,mat_diag]=spec(A);	vect_p : matrice P des vecteurs propres de A mat_diag : matrice diagonale D obtenue vect_p et mat_diag ont la même taille que A

(*) Les matrices de Scilab sont stockées colonne par colonne, ce qui donne l'ordre dans lequel les éléments sont comptés

Probabilités et statistiques (1)

Commandes	Scilab	Remarques
Simulation avec grand : lois binomiales, de Poisson, normales, uniformes à densité, uniformes discrètes, exponentielles, géometriques, Gamma	y1 = grand(m, p, ”bin”, n, p); y2 = grand(m, p, ”poi”, lambda); y3 = grand(m, p, ”nor”, mu, sigma); y4 = grand(m, p, ”unf”, a, b); y5 = grand(m, p, ”uin”, 1, n); y6 = grand(m, p, ”exp”, 1/lambda); y7 = grand(m, p, ”geom”, p); y8 = grand(m, p, ”gam”, b, nu);	Génère une matrice de taille m × p constituée de nombres aléatoires distribués selon la loi spécifiée. **Remarques** : pour la **loi normale** c’est l’**écart-type** qui est donné, pour la **loi exponentielle** c’est **l’espérance**.
Commandes : min, max	x =grand(1, 100, ”nor”, 4, 2); m = min(x) M = max(x)	Simulation d’une variable *X* suivant une loi normale d’espérance 4 et d’écart-type 2, m est la plus petite valeur prise par X M est la plus grande
Commandes : sum, cumsum	x = [1 3 5 2] s = sum(x) s_cumul = cumsum(x)	La valeur prise par s est 11, s_cumul est une matrice ligne égale à [1 4 9 11]
Commande : median	x =grand(1,100,”nor”,4,2); med = median(x)	med est égale à la médiane il y a autant de valeurs inférieures à med que de valeurs supérieures à med
Moyenne empirique, Variance empirique commande : mean,	lambda = 2 x =grand(1,100,”exp”,1/lambda) m = mean(x) V = mean ((x-m).^2)	Simulation d’une variable *X* suivant une loi exponentielle de paramètre 2 avec grand (x est une matrice ligne contenant 100 valeurs prises par *X*), puis calcul de la moyenne m et de la variance empirique $V = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$
commande : variance (estimateur débiaisé)	lambda = 2 x =grand(1,100,”exp”,1/lambda) V = variance(x)	La commande variance donne : $V = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
commande : stdev (estimateur débiaisé)	lambda = 2 x =grand(1,100,”exp”,1/lambda) ecart = stdev(x)	La commande stdev donne : $ecart = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$
Covariance empirique d’une série statistique à deux variables	x =grand(1,100,”nor”,0,1) y =grand(1,100,”nor”,0,1) m_x = mean(x) m_y = mean(x) cov = mean((x-m_x).*(x-m_y))	Calcul de $\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \frac{1}{n}\sum_{i=1}^{n}x_i y_i - \bar{x}\bar{y}$ à l’aide de mean

Probabilités et statistiques (2)

Commandes	Scilab	Remarques
Covariance commande : corr (estimateur débiaisé)	x =grand(1,100,"nor",0,1) y =grand(1,100,"nor",0,1) covar = corr(x,y,1)	covar est égale à : $\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})$
Commande binomial Diagramme en bâtons Commandes plot2d3	x = binomial(0.6, 20) plot2d3(0:20, x)	La commande binomial permet d'obtenir les coefficients d'une loi binomiale, plot2d3 permet de tracer un diagramme en bâtons
Diagramme en barres Commande bar	x = binomial(0.6, 20) bar(0:20, x)	0:20 donne les abscisses des barres
Tracés d'histogrammes Commandes : histplot(b,x)	x =grand(1,100,"nor",1,2) b=[-9,-4,0,1,2,5,10] histplot(b,x)	Trace l'histogramme de x en utilisant les classes dont les bornes sont définies par le vecteur b ($[b_1,b_2],[b_2,b_3],\ldots$)
Tracés d'histogrammes Commandes : histplot(n,x)	x =grand(1,100,"nor",1,2); histplot(10, x);	Trace l'histogramme de x en utilisant 10 classes de même largeur, l'effectif de chaque classe est normalisé par l'effectif total

Répartition en classes
Lois quelconques
Commande :
dsearch(x,b,"c")
	x=grand(1,20,"exp",3);
	b=[0,1,2,3,4,6,10];
	[pos,eff]=dsearch(x,b,"c");
		Recherche parmi les éléments du vecteur x,
		ceux qui se trouvent dans une
		des classes définies par b.
		pos est un vecteur de même taille que x,
		qui indique le numéro de la classe
		à laquelle appartient chaque élément.
		eff donne l'effectif de chaque classe.

Répartition en classes
Lois discètes.
Commande :
dsearch(x,b,"d")
	x=grand(1,20,"bin",10, 0.7);
	v=[0,1,2,3,4,5,6,7,8,9,10];
	[pos,eff]=dsearch(x,b,"d");
		Même fonction que ci-dessus, sauf que
		la recherche se fait par rapport aux
		valeurs entières définies dans le vecteur v

Commande : tabul (tabulation) "i" pour "increase" "d" pour "decrease"	x=grand(1,30, "bin" ,20 , .3) m = tabul(x, "i") plot2d3(m(:,1) ,m(:,2)/30)	Simulation d'un 30-échantillon d'une loi binomiale de paramètres 20 et 0.3 m(:, 1) donne les valeurs prises classées par ordre croissant, m(:, 2) donne les effectifs Tracé d'un diagramme en bâtons
Commande : pie	pie([1 2 5])	Dessine un cercle graphique comportant 3 parts, la première représente $\frac{1}{8}$, la deuxième $\frac{1}{4}$, la troisième $\frac{5}{8}$

Probabilités et statistiques (3)

Commandes	Scilab	Remarques
Fonction de répartition Commande : P=cdfnor(”PQ”,X,Mean,Std) (”cumulative distribution function - normal”), cdfbin ... (binomiale) cdfpoi... (Poisson) ...	mu = 2 sigma = 5 x = 6 P = cdfnor(’PQ’,x,mu,sigma)	Calcule pour une loi normale de paramètres 2 et 25, la probabilité $P = \Phi(x)$ connaissant le seuil x
Quantiles, inversion des fonctions de répartition. Commande : X=cdfnor(”X”,Mean,Std,P,Q)	a = 0.05 x = cdfnor(’X’,0,1,1-a/2,a/2)	Calcul de la valeur de x tel que $\Phi(x) = 1 - \frac{a}{2}$

Cas d’une matrice rectangulaire
Moyenne, médiane, maximum, colonne par colonne
commandes :
mean, median, max
	x =grand(100,50,”nor”,4,2);
	moyennes = mean(x, "r");
	maxima = max(x, "r");
	minima = min(x, "r");
	medianes = median(x, "r");
		Simulation de 100 échantillons des valeurs prises par 50 variables aléatoires X_i indépendantes suivant une loi normale d’espérance 4 et d’écart-type2, x est une matrice contenant 100 lignes et 50 colonnes (chaque colonne correspond aux valeurs prises par une variable X_i)
		Calcul de la moyenne empirique, du maximum, du minimum, de la médiane pour chacune des variables aléatoires
		moyennes est une matrice ligne à 50 colonnes ("r" pour "row" ou rangée la moyenne est effectuée colonne par colonne)

Cas d’une matrice rectangulaire Moyenne, médiane, maximum, ligne par ligne commandes : mean, median, max	x =grand(50,100,”nor”,4,2) moyennes = mean(x, "c") maxima = max(x, "c") minima = min(x, "c") medianes = median(x, "c")	Même simulation que ci-dessus, mais les moyennes, maximum etc. sont calculés pour chaque ligne, les résultats figurent dans des matrices colonne
Simulation d’une chaîne de Markov	Y = grand(p, ’markov’, M, X0)	X0 est une matrice contenant les numéros des états initiaux (par exemple s’il y a 3 états et que l’on veut effectuer 8 simulations avec un état initial 1, X0 = [1 1 1 1 1 1 1 1]) p est le nombre de pas effectué M est la matrice de transition (stochastique en lignes)

Tracés de figures et de surfaces avec plot2d et plot3d (1)

<table>
<tr><th>Commandes</th><th>Scilab</th><th>Remarques</th></tr>
<tr><td>Tracé du graphe de
$f : x \mapsto xe^{-x}$
Première méthode :
utilisation des
opérations coefficient
par coefficient
(.*, ./, .^, .+)
commandes : plot2d
et xtitle</td><td>x = [-1:0.1:5]
y = x .* exp(-x)
plot2d(x, y, style = 2)
xtitle(”Graphe de f :”)</td><td>x est une matrice ligne dont les coefficients
varient de -1 à 5 par pas de 0.1
y est une matrice ligne de même taille que x
Penser à utiliser .*, le paramètre 2 donne
la couleur de la courbe :
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>noir</td><td>bleu</td><td>vert</td><td>cyan</td><td>rouge</td><td>magenta</td><td>jaune</td></tr></table>
xtitle permet de rajouter un titre</td></tr>
<tr><td>Tracé du graphe de
$f : x \mapsto xe^{-x}$
Deuxième méthode :
pour éviter les
opérations coefficient
par coefficient,
utilisation de deff
et de fplot2d</td><td>deff(’y=f(x)’, ’y= x*exp(-x)’)
x = [-1:0.1:5]’
fplot2d(x, f, style = 2)
xtitle(”Graphe de f :”)</td><td>La commande deff permet de définir
rapidement une fonction simple (sinon on
peut utiliser function y=f() ... endfunction)
x est une matrice colonne
fplot2d permet d’éviter de définir la matrice
colonne y (sinon on peut aussi écrire :
y = feval(x,f); plot2d(x, y, 2);)</td></tr>
<tr><td>Tracé du graphe de
$f : x \mapsto xe^{-x}$
Troisième méthode :
Utilisation de la
commande : feval
et de la déclaration
générale d’une
fonction</td><td>x = [-1:0.1:5]
function y=f(x)
y = x * exp(-x)
endfunction
y = feval(x, f)
plot2d(x, y, 3, rect=[0.7,0.8,4,3])
xtitle(”Graphe de f ”)</td><td>On commence par définir la fonction f
x est ici une variable locale (x est un réel)
La commande feval permet de créer
la matrice ligne y
rect[$x_{\min}, y_{\min}, x_{\max}, y_{\max}$] permet de définir
la taille de la fenêtre graphique</td></tr>
<tr><td>Tracé du graphe de
$f : x \mapsto \frac{x^2}{2x-1}$ en vert
et de l’asymptote
(en rouge)
dans la même fenêtre
avec un titre et une
légende</td><td>x = [1:0.01:4]
y1 = (x.^2) ./ (2*x-1)
y2 = 0.5 * x + 0.25
plot2d(x, y1, style=5)
plot2d(x, y2,style=5, rect=[1,0,4,3])
xtitle(”Graphe de f et asymptote”)
legend(”Graphe de f”, ”asymptote”)</td><td>Pour tracer plusieurs courbes sur un même
graphique, on peut soit définir x, $y1$ et $y2$
comme des matrices lignes comme ici,
soit procéder comme ci-dessous (*)
On rajoute un titre et une légende</td></tr>
<tr><td>(*) Même exemple
que ci-dessus, mais
avec des matrices
colonnes, frameflag,
axesflag, xgrid</td><td>x = [1:0.01:4]’
y1 = (x.^2) ./ (2*x-1)
y2 = 0.5 * x + 0.25
plot2d([x x],[y1 y2], [3 5], ..
frameflag = 6, axesflag = 5)
xgrid(1)
xtitle(’Variante’)
legend(”Graphe de f”, ”asymptote”)</td><td>x est ici une matrice colonne de même que
y1 et y2, ce qui permet d’utiliser une seule
instruction plot2d, .. permet de passer à
la ligne dans une commande (lorsqu’elle
est trop longue à écrire sur une ligne),
frameflag=3 : échelle isométrique,
frameflag=6 : graduations simples,
axesflag=5 : axes qui se croisent au milieu
xgrid(1) : ajout d’une grille</td></tr>
</table>

Tracés de figures et de surfaces avec plot2d et plot3d (2)

<table>
<tr><th>Commandes</th><th>Scilab</th><th>Remarques</th></tr>
<tr><td>Tracé de points sans ligne brisée entre les points
Paramètre
style = n</td><td>x = [-1:0.2:5]
y = x .* exp(-x)
plot2d(x, y, style = -1)
xtitle(”Tracé des points”)</td><td>style = -1 permet de remplacer
les points par des croix :
<table><tr><td>0</td><td>–1</td><td>–2</td><td>–3</td><td>–4</td><td>–5</td></tr><tr><td>•</td><td>+</td><td>×</td><td>⊕</td><td>∗</td><td>◇</td></tr></table>
<table><tr><td>–6</td><td>–7</td><td>–9</td><td>–10</td><td>–11</td></tr><tr><td>△</td><td>▽</td><td>◦</td><td>∗</td><td>□</td></tr></table></td></tr>
<tr><td>Tracés de figures dans des fenêtres différentes
commande :
xset(’window’, n)</td><td>x = [1:0.01:4]
y1 = x.^2
y2 = 3 * x + 1
xset(’window’, 1)
plot2d(x, y1, 3)
xset(’window’, 2)
plot2d(x, y2, 5)</td><td>Trace la courbe représentative de $f : x \mapsto x^2$
dans la fenêtre numéro 1, et la courbe
représentative de $g : x \mapsto 3x + 1$
dans la fenêtre 2</td></tr>
<tr><td>Utilisation de sous-fenêtres,
commande :
subplot(1,2,n)</td><td>x = [1:0.01:4]
y1 = x.^2
y2 = 3 * x + 1
subplot(1,2,1)
plot2d(x, y1, 3)
subplot(1,2,2)
plot2d(x, y2, 5)</td><td>Création de deux sous-fenêtres sur une ligne,
tracés de la première courbe dans la première
sous-fenêtre et de la deuxième courbe dans
la deuxième sous-fenêtre</td></tr>
<tr><td>Commande
plot3d</td><td>function z = f(x,y)
z = x^4 - y^4
endfunction
x=-3:0.2:3; y = x
z = feval(x, y, f)
plot3d(x,y,z,alpha=5,theta=31)</td><td>Tracé du graphe de la fonction
$(x,y) \mapsto x^4 - y^4$
définie sur $[-3,3]^2$
alpha et theta donnent les
coordonnées sphériques
du point d’observation</td></tr>
<tr><td>Commande
fplot3d</td><td>function z = f(x,y)
z = x^4 - y^4
endfunction
x=-3:0.2:3; y = x
fplot3d(x,y,f,alpha=5,theta=31)</td><td>Tracé du graphe de la fonction
$(x,y) \mapsto x^4 - y^4$
définie sur $[-3,3]^2$
alpha et theta donnent les
coordonnées sphériques
du point d’observation</td></tr>
</table>