

Tutoriel Gnuplot

Bernard Desgraupes
Université Paris Ouest

Février 2011

1	Introduction	4
1.1	Exemples simples	4
1.2	Spécification d'options	5
1.3	Lecture depuis un fichier de données	5
2	Généralités	7
2.1	Aide en ligne	7
2.2	Systèmes de coordonnées	7
2.3	Fonctions mathématiques	8
2.4	Variables et fonctions définies par l'utilisateur	10
2.4.1	Fonctions et variables	11
2.4.2	Substitution par macros	11
3	Les options de Gnuplot	12
3.1	Options globales	13
3.1.1	Unités	14
3.1.2	Dates	14
3.1.3	Chemins d'accès	16
3.1.4	Informations d'état	16
3.2	Options concernant les entrées et sorties	16
3.2.1	Sorties graphiques	16
3.2.2	Redirections	18
3.2.3	Sorties tabulées	19
3.3	Options concernant les axes	19
3.3.1	Origine	19
3.3.2	Échelle	20
3.3.3	Axes du repère	21
3.3.4	Marques de graduation	21
3.3.5	Données temporelles	24
3.3.6	Orientation des axes	24
3.4	Options concernant les coordonnées	25
3.4.1	Nom des coordonnées	25
3.4.2	Systèmes de représentation	26
3.4.3	Intervalles de valeurs	28

3.5	Options concernant les titres et légendes	28
3.5.1	Titres	28
3.5.2	Étiquettes	29
3.5.3	Légendes	31
3.6	Options concernant les dimensions	33
3.7	Styles graphiques	35
3.8	Ornements et disposition	36
3.8.1	Grille	36
3.8.2	Bordure	38
3.8.3	Flèches	39
3.8.4	Graphique vide	40
3.8.5	Graphiques multiples	41
3.8.6	Lignes de niveau d'une surface	42
3.8.7	Paramètres de dessin	44
3.9	Objets graphiques	48
3.9.1	Rectangles	48
3.9.2	Cercles	49
3.9.3	Ellipses	50
3.9.4	Polygones	51
3.10	Options concernant les nuanciers	52
4	Fichiers de données	54
4.1	Types de fichiers	54
4.1.1	Fichiers texte	54
4.1.2	Fichiers binaires	55
4.2	L'option <i>datafile</i>	59
4.3	Représentation à partir d'un fichier de données	59
4.3.1	Transformation des données	59
4.3.2	Lissage des données	61
5	Les types de graphiques	65
5.1	Le style <i>labels</i>	66
5.2	Le style <i>filledcurves</i>	67
5.3	Le style <i>circles</i>	70
5.4	Points et lignes	70
5.5	Le style <i>vectors</i>	71
5.6	Diagrammes de dispersion	71
5.7	Fonctions en escalier	72
5.8	Histogrammes	73
5.9	Barres d'erreur	75
5.10	Le style <i>pm3d</i>	79
5.10.1	L'algorithme <i>pm3d</i>	79
5.10.2	L'option <i>pm3d</i>	80
5.11	Les images	83
6	Ajustement de données	83
6.1	Exemple simple	84
6.2	La commande fit	86
6.3	L'option <i>fit</i>	87
6.4	La commande update	88

7 Autres commandes de Gnuplot	88
7.1 Les commandes cd et pwd	88
7.2 La commande clear	88
7.3 La commande history	89
7.4 La commande if	90
7.5 Les commandes save , load et call	90
7.6 La commande pause	91
7.7 Les commandes exit et quit	91
7.8 La commande reread	91
7.9 Les commandes shell et system	92
7.10 La commande test	93
8 Annexes	95
8.1 Les dates	95
8.2 Les chaînes de caractères	96
8.3 Les couleurs	97
8.3.1 Exemples	97
8.3.2 Réglages de la palette	98
8.3.3 Correspondance gris-couleurs	100
8.4 Gnuplot et L ^A T _E X	103
Index	105

Dernière mise à jour : 30 mars 2011

1 Introduction

La commande **gnuplot** exécute les instructions contenues dans un ou plusieurs fichiers scripts ou peut être utilisée de manière interactive en exécutant des instructions une à une depuis une console. Ces fichiers scripts contiennent des instructions permettant la réalisation de graphiques en deux ou trois dimensions.

La syntaxe de la commande est :

```
gnuplot [ X11 options ] [file ...]
```

Si aucun fichier n'est passé en argument, la commande fonctionne de manière interactive : elle présente une ligne de commande sur laquelle on peut exécuter des instructions une à une.

Pour quitter la ligne de commande, il suffit de taper **exit** ou **quit**.

Les deux commandes principales de **gnuplot** sont **plot** pour le tracé de graphiques en 2D et **splot** pour le tracé de surfaces en 3D.

Il y a deux manières différentes de spécifier un graphique à représenter :

- on peut passer une expression algébrique qui définit une fonction à une ou deux variables, c'est-à-dire de la forme $y = f(x)$ ou $z = f(x, y)$.
- on peut aussi faire lire dans un fichier les coordonnées des points à représenter ou des valeurs numériques à partir desquelles calculer ces coordonnées.

Ces deux approches sont présentées dans les sections 1.1 et 1.3 respectivement.

1.1 Exemples simples

Pour représenter la fonction $\sin(x)$, il suffit d'écrire :

```
plot sin(x)
```

Pour représenter plusieurs courbes sur le même graphique, on les indique dans la même commande **plot** en les séparant par une virgule. Par exemple :

```
plot sin(x), cos(x)
```

Par défaut, **gnuplot** utilise des couleurs différentes pour chaque courbe. On peut obtenir le même résultat avec la commande **replot** comme ceci :

```
plot sin(x)
replot cos(x)
```

Ici la commande **replot** sert à ajouter une courbe à un graphique déjà existant.

On peut aussi indiquer, dans la même commande, la définition d'une fonction à tracer, puis passer des paramètres et appeler cette fonction. Par exemple :

```
plot f(x) = sin(a*pi*x), a = 1, f(x), a = 2, f(x)
```

On délimite les intervalles sur lesquels tracer les courbes ou les surfaces au moyen d'une paire de crochets indiquant les extrémités de l'intervalle, séparées par un symbole deux-points. Par exemple :

```
plot [-2*pi:2*pi] sin(x), cos(x)
```

Pour des graphiques en 3 dimensions, on peut indiquer deux intervalles successifs, l'un pour les abscisses et l'autre pour les ordonnées. Par exemple :

```
splot [-2:4] [-5:10] x**2-y**2
```

Les noms des commandes peuvent tous être abrégés tant que le nom utilisé ne cause pas d'ambiguïté. On peut donc écrire

```
pl sin(x)
sp x**2*y
hist 5
```

au lieu de

```
plot sin(x)
splot x**2*y
history 5
```

1.2 Spécification d'options

On peut passer certaines options à la suite des commandes **plot** et **splot**. Les principales options affectant le tracé sont *linetype* (ou *lt*), *linecolor* (ou *lc*) et *linewidth* (ou *lw*). Par exemple :

```
plot sin(x) lt 3 linecolor 1 lw 2
```

Pour les périphériques qui supportent les motifs *point/tiret*, chaque type (*linetype*) définit à la fois un motif et une couleur. L'option *linecolor* permet de modifier la couleur par défaut.

On se sert souvent de la commande **replot** pour modifier certains paramètres et réexécuter la dernière commande **plot** ou **splot**. Les arguments passés à la commande **replot** sont ajoutés à la précédente commande **plot** ou **splot**.

La commande **replot** est aussi utilisée sans arguments pour simplement redessiner un graphique après que des options ont été modifiées par des commandes **set**.

1.3 Lecture depuis un fichier de données

Plutôt que de passer la définition d'une fonction, on peut indiquer le nom d'un fichier de données à représenter. Par exemple :

```
plot 'exemple.dat'
```

Ce sont les apostrophes entourant le nom du fichier qui permettent à **gnuplot** de ne pas confondre cette syntaxe de la commande **plot** avec celle vue à la section 1.1. Sans les apostrophes, **gnuplot** considérerait qu'il s'agit du nom d'une fonction.

Les données sont stockées en colonnes. Chaque ligne du fichier contient plusieurs champs et on se sert de l'option *using* pour indiquer quelles colonnes

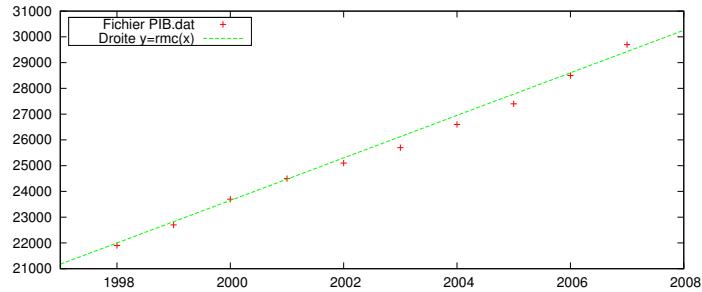


FIGURE 1 – Données du fichier *PIB.dat*.

utiliser. Les colonnes sont désignées par leur indice : la première a l’indice 1, la seconde l’indice 2, etc. L’exemple qui suit suppose que le fichier de données contient au moins trois colonnes et que les colonnes 1 et 3 doivent être utilisées pour définir les abscisses et les ordonnées respectivement :

```
plot 'exemple.dat' using 1:3
```

L’option *using* peut aussi comporter des expressions à calculer à partir des valeurs trouvées dans certaines colonnes. Les expressions à calculer doivent être placées entre parenthèses et les colonnes sont dans ce cas représentées par leur indice précédé d’un symbole dollar suivi du numéro de la colonne, comme par exemple `$1`, `$2`, etc. L’exemple qui suit indique que les ordonnées sont calculées en faisant la moyenne des colonnes 2 et 3 :

```
plot 'exemple.dat' using 1:( ($2+$3)/2 )
```

Considérons un fichier *PIB.dat* contenant les données suivantes :

```
# PIB de la France depuis 1998 (données Eurostat)
1998    21900
1999    22700
2000    23700
2001    24500
2002    25100
2003    25700
2004    26600
2005    27400
2006    28500
2007    29700
```

L’exemple qui suit mélange les deux techniques de représentation des données : lecture depuis un fichier et définition par une expression algébrique. Il crée un graphique à partir des données contenues dans le fichier *PIB.dat* et ajoute une droite de régression définie par une fonction appelée *rmc* :

```
rmc(x) = 825.5*(x-1998) + 22000
plot [1997:2008] 'PIB.dat', rmc(x)
```

Le graphe est représenté sur la figure 1.

2 Généralités

2.1 Aide en ligne

On peut obtenir de l'aide sur les commandes et sur un certain nombre de sujets au moyen de la commande **help**.

Par exemple, pour afficher de l'aide à propos de la commande **plot** :

```
help plot
```

À la fin de l'aide affichée, on trouve parfois une liste de sujets dérivés. On peut aussi obtenir la liste des principaux sujets d'aide en faisant suivre la commande **help** d'un point d'interrogation :

```
gnuplot> help ?
Help topics available:
3D                  environment          newhistogram
automated           examples             object
backwards           expressions          plotting
batch/interactive   fonts               polygon
binary              gd                  pseudocolumns
binary_examples     glossary            quotes
bugs                gnuplot-defined      rectangle
canvas              gprintf             rgbcolor
circle              graphical            set
colornames          help-desk           show
colorspec           image               startup
commands            introduction         strings
comments            iteration            substitution
complete            line-editing         syntax
coordinates         linecolor            time/date
copyright           linetype             using
datastrings         mixing_macros_backquotes xticlabels
ellipse             mouse               new-features
```

2.2 Systèmes de coordonnées

Par convention, les coordonnées cartésiennes sont désignées par x et y pour des graphiques en 2 dimensions, et x , y et z pour des graphiques en 3 dimensions.

Lorsqu'on utilise des coordonnées polaires, le rayon vecteur est désigné par la lettre r et l'angle polaire par la lettre t . La valeur de t est comprise entre 0 et 2π , ce qui se note $[0:2*pi]$ dans les commandes de **gnuplot**. Si jamais les angles sont mesurés en degrés (voir l'option *angles*), la valeur de t est comprise entre 0 et 360, ce qui se note $[0:360]$ dans les commandes de dessin.

La commande **plot** permet d'utiliser les quatre côtés du graphique. Ceux-ci sont désignés par les lettres x (côté inférieur), $x2$ (côté supérieur), y (côté gauche) et $y2$ (côté droit) comme des axes indépendants. L'option *axes* permet de préciser les axes à utiliser.

La syntaxe la plus complète pour désigner des coordonnées dans une commande est :

```
{<système>} <x>, {<système>} <y> {,{<système>} <z>}
```

où $<systeme>$ est un des mots-clés suivants : *first*, *second*, *graph*, *screen*, ou *character*.

Si le système n'est pas spécifié pour la coordonnée x , c'est *first* qui est utilisé par défaut. Si le système n'est pas spécifié pour la coordonnée y , le même système que pour x est utilisé.

Les différences entre les systèmes sont les suivantes :

- *first* utilise les axes du bas et de gauche pour placer x et y ;
- *second* utilise les axes du haut et de droite pour placer x et y ;
- *graph* spécifie une zone entre les axes sachant que le coin inférieur gauche est $(0,0)$ en deux dimensions et $(0,0,0)$ en trois dimensions, et que le coin supérieur droit est $(1,1)$ en deux dimensions et $(1,1,1)$ en trois dimensions ;
- *screen* désigne la zone entière de l'écran avec les mêmes valeurs pour les coins inférieur gauche et supérieur droit qu'avec le système *graph* ;
- *character* donne les coordonnées en fonction de la largeur et de la hauteur des caractères, ce qui dépend donc de la taille de la police choisie.

Dans certains cas, les valeurs des coordonnées sont relatives à d'autres valeurs. Le rapport entre les coordonnées est additif pour des coordonnées normales et multiplicatif pour des coordonnées logarithmiques.

2.3 Fonctions mathématiques

Le programme **gnuplot** comporte une importante bibliothèque de fonctions mathématiques classiques qui peuvent être utilisées dans les définitions des courbes à tracer. Le tableau 1 indique le nom et les arguments des fonctions disponibles.

On peut exécuter les fonctions mathématiques interactivement au moyen de la commande **print**. Voici quelques exemples :

```
gnuplot> print sqrt(2)
1.4142135623731
gnuplot> print cos(pi/3), sin(pi/3), tan(pi/3)
0.5 0.866025403784439 1.73205080756888
gnuplot> print rand(0)
0.369445210944905
gnuplot> print sgn(-9)
-1
gnuplot> print exp(1)
2.71828182845905
gnuplot> print norm(1.96)
0.97500210485178
gnuplot> print invnorm(0.975)
1.95996398454005
gnuplot> print ibeta(1.0,0.5,0.1)
0.0513167019488992
```

Les nombres complexes sont désignés par leurs parties réelle et imaginaire placées entre accolades. Le nombre $4 + 3i$ est ainsi représenté sous la forme $\{4, 3\}$ ou $\{4.0, 3.0\}$. Les fonctions *abs* et *arg* renvoient respectivement le module et l'argument du nombre complexe. Par exemple :

```
gnuplot> print arg(\{4.0, 3.0\})
0.643501108793284
gnuplot> print abs(\{4.0, 3.0\})
5.0
```

$abs(x)$	valeur absolue d'un nombre réel
$abs(z)$	module d'un nombre complexe
$acos(x)$	arc cosinus
$acosh(x)$	arc cosinus hyperbolique
$arg(z)$	argument d'un nombre complexe
$asin(x)$	arc sinus
$asinh(x)$	arc sinus hyperbolique
$atan(x)$	arc tangente de x
$atan2(y,x)$	arc tangente de y/x
$atanh(x)$	arc tangente hyperbolique
$besj0(x)$	fonction de Bessel j_0
$besj1(x)$	fonction de Bessel j_1
$besy0(x)$	fonction de Bessel y_0
$besy1(x)$	fonction de Bessel y_1
$ceil(x)$	valeur entière supérieure
$cos(x)$	cosinus
$cosh(x)$	cosinus hyperbolique
$erf(x)$	fonction d'erreur
$erfc(x)$	fonction d'erreur complémentaire
$exp(x)$	exponentielle
$floor(x)$	valeur entière inférieure
$gamma(x)$	fonction gamma
$ibeta(a,b,x)$	fonction beta incomplète
$igamma(a,x)$	fonction gamma incomplète
$imag(z)$	partie imaginaire d'un nombre complexe
$int(x)$	valeur entière tronquée vers 0
$inverf(x)$	fonction d'erreur réciproque
$invnorm(x)$	fonction quantile de la loi normale $\mathcal{N}(0, 1)$
$lambertw(x)$	fonction W de Lambert
$lgamma(x)$	log de la valeur absolue de la fonction gamma
$log(x)$	logarithme népérien
$log10(x)$	logarithme décimal
$norm(x)$	fonction de répartition de la loi normale $\mathcal{N}(0, 1)$
$rand(x)$	nombre au hasard
$real(z)$	partie réelle d'un nombre complexe
$sgn(x)$	fonction signe
$sin(x)$	sinus
$sinh(x)$	sinus hyperbolique
$sqrt(x)$	racine carrée
$tan(x)$	tangente
$tanh(x)$	tangente hyperbolique

TABLE 1 – Fonctions mathématiques

```
gnuplot> print real({4.0, 3.0})
4.0
gnuplot> print imag({4.0, 3.0})
3.0
```

Le nombre π est désigné par la variable prédéfinie *pi* :

```
gnuplot> print pi
3.14159265358979
```

2.4 Variables et fonctions définies par l'utilisateur

Les variables peuvent être de type numérique ou des chaînes de caractères. Les valeurs numériques peuvent être entières ou réelles : dès qu'une expression contient une valeur réelle, tous les calculs sont effectués en arithmétique à virgule flottante. Au contraire, si elle ne contient que des entiers, les calculs se font en arithmétique entière. Par exemple :

```
gnuplot> print 3/2
1
gnuplot> print 3.0/2
1.5
```

L'expression $1/0$ ne provoque pas d'erreur. Elle représente une valeur indéterminée et on peut l'utiliser explicitement en ce sens. Par exemple :

```
f(x) = (0<=x && x<1) ? x : 1<=x && x<2 ? 2-x : 1/0
```

L'expression précédente définit une fonction triangulaire f égale x sur l'intervalle $[0, 1]$, à $2 - x$ sur l'intervalle $[1, 2]$ et non définie ailleurs.

Il existe une constante interne appelée *NaN* (*Not a Number*) qui joue le même rôle. On aurait pu écrire aussi bien :

```
f(x) = (0<=x && x<1) ? x : 1<=x && x<2 ? 2-x : NaN
```

Le point sert d'opérateur de concaténation pour les chaînes. Par exemple :

```
gnuplot> x="abc"
gnuplot> y="def"
gnuplot> print x.y
abcdef
```

On peut extraire des sous-chaînes d'une chaîne avec un suffixe de la forme $[m:n]$. Cela extrait les caractères entre les position m et n incluses. Par exemple :

```
gnuplot> mot="abracadabra"
gnuplot> print mot[5:8]
    cada
gnuplot> print mot[5:*)
    cadabra
gnuplot> print mot[*:5]
    abrac
```

gnuplot définit de nombreuses variables internes commençant par le préfixe *GPVAL_*. On peut en obtenir la liste avec l'instruction suivante :

```
show variables all
```

Cette liste peut être plus ou moins longue selon que des commandes telles que **plot** ou **splot** ont déjà été exécutées ou pas. Leurs valeurs peuvent servir dans des tests conditionnels : par exemple, la variable *GVAL_MULTIPILOT* permet de déterminer si on est en mode *multiplot* ou pas et un script pourrait définir des actions différentes selon qu'elle vaut 0 ou 1.

2.4.1 Fonctions et variables

On peut définir des fonctions numériques en utilisant la syntaxe schématique suivante :

```
f(arguments) = expression
```

Le nom de la fonction peut contenir des lettres et des chiffres. On peut déclarer de 1 à 12 arguments (des versions anciennes de **gnuplot** limitaient à 5 arguments). La définition de la fonction est une expression écrite en termes des variables de la fonction et de constantes. Les constantes n'ont pas besoin d'être préalablement déclarées tant qu'on ne cherche pas à évaluer la fonction.

Par exemple, la déclaration suivante est acceptée même si *a* et *b* n'ont pas encore été définis :

```
pcs(x,y) = a*cos(x) + b*sin(y)
```

La syntaxe pour définir une variable est simplement :

```
nom_variable = expression_constante
```

Par exemple :

```
a = floor(10*atan(1))
e2 = exp(2)
c = 299792458
```

2.4.2 Substitution par macros

Une macro est une variable contenant une chaîne de caractères : lorsque ce nom est précédé d'un symbole @, le mécanisme de substitution remplace le nom de la variable par sa valeur, c'est-à-dire par la chaîne de caractères qu'elle représente. Pour activer la substitution des macros, il faut préalablement inclure l'instruction suivante :

```
set macros
```

Voici un exemple :

```
set macros
sty = "lines lt 4 lw 2"
plot sin(x) with @sty
```

On désactive la substitution au moyen de l'instruction *unset macros*.

L'utilisation d'apostrophes obliques (accents graves) constitue un autre type de substitution. Cette syntaxe est analogue à celle utilisée par les shells Unix. Les apostrophes obliques entourent une commande Unix qui est exécutée, puis elles sont remplacées par le résultat de cette commande. Par exemple, la commande Unix

```
ls | wc -l
```

est une manière classique de calculer le nombre de fichiers dans le répertoire courant. On peut obtenir et stocker cette valeur dans une variable *x* avec **gnuplot** de la manière suivante :

```
x= 'ls | wc -l'
```

Attention : il s'agit d'apostrophes obliques comme des accents graves, non d'apostrophes droites ordinaires.

3 Les options de Gnuplot

gnuplot supporte de très nombreuses options qui sont décrites dans les sections qui suivent.

Les trois commandes de base permettant de manipuler ces options sont **show**, **set** et **unset**. Voici quelques exemples simples illustrant l'utilisation de ces commandes.

Pour connaître les valeurs courantes d'une option, on se sert de la commande **show**. Par exemple, les valeurs des options *encoding* et *border* s'obtiennent de la manière suivante :

```
gnuplot> show encoding  
encoding is default  
  
gnuplot> show border  
border 31 is drawn in front of the plot elements with  
linetype -1 linewidth 1.000
```

On peut obtenir les valeurs de toutes les options disponibles avec la commande suivante :

```
show all
```

Cette commande renvoie aussi des informations sur le programme lui-même (taille des types de variables sur la machine utilisée) et dresse la liste des variables et fonctions définies par l'utilisateur.

La commande **set** permet de fixer une nouvelle valeur pour une option. Par exemple, pour passer en coordonnées polaires, on utilise la commande suivante :

```
set polar
```

On repassera ensuite en coordonnées cartésiennes avec la commande *unset* comme ceci :

```
unset polar
```

arrowstyle	as
fillcolor	fc
fillstyle	fs
linecolor	lc
linespoints	lp
linestyle	ls
linetype	lt
linewidth	lw
pointinterval	pi
pointsize	ps
pointtype	pt
textcolor	tc

TABLE 2 – Abréviations supportées par les options de **gnuplot**.

La syntaxe pour spécifier les valeurs des options dépend de chaque option. Certaines options attendent une valeur numérique, d'autres une valeur symbolique prédéfinie ou une expression plus compliquée. Par exemple :

```

1   set boxwidth 0.5
2   set angles radians
3   set style line 5 lt rgb "cyan" lw 3 pt 6

```

Il existe aussi une commande **reset** qui rétablit les valeurs par défaut des diverses options qui auront été modifiées par une commande **set**. Elle laisse cependant inchangées les options *terminal*, *output*, *loadpath* et *fontpath*.

Les sections qui suivent indiquent le fonctionnement de chacune des options et donnent des exemples de leur utilisation.

Les options peuvent toutes être abrégées tant que le nom utilisé ne cause pas d'ambiguïté. On peut donc indifféremment écrire

```

set terminal
set termi
set term
set ter

```

Il y a, d'autre part, un certain nombre d'arguments optionnels fréquemment utilisés et pour lesquels **gnuplot** fournit des abréviations commodes. Ils sont rassemblés dans le tableau 2.

Les abréviations permettent d'écrire des commandes plus compactes, au risque de les rendre difficiles à lire. Par exemple :

```
plot x w p ps 3
```

est synonyme de

```
plot x with points pointsize 3
```

3.1 Options globales

Cette section regroupe les réglages d'ordre général qu'on peut effectuer concernant le fonctionnement du programme.

%d	jour du mois, 1-31
%m	mois de l'année, 1-12
%y	année, 0-99
%Y	année, 4 chiffres
%j	jour de l'année, 1-365
%H	heure, 0-24
%M	minute, 0-60
%s	secondes depuis l'époque Unix
%S	secondes, 0-60
%b	abréviation du nom du mois
%B	nom du mois

TABLE 3 – Spécificateurs de temps et heure

L’option *encoding* permet de sélectionner l’encodage utilisé pour les chaînes de caractères. Les valeurs possibles sont : *cp1250*, *cp437*, *cp850*, *cp852*, *default*, *iso_8859_1*, *iso_8859_15*, *iso_8859_2*, *koi8r*, *koi8u*.

L’option *historysize* permet de fixer la taille de l’historique des commandes. La valeur par défaut est 500. Si on utilise la commande *unset historysize*, on supprime toute limite à l’historique.

L’option *macros* permet d’activer la substitution des macros contenues dans les commandes. Voir le paragraphe 2.4.2.

L’option *mouse* permet d’activer certaines interactions avec la souris comme, par exemple, cliquer sur un graphique afin d’obtenir les coordonnées d’un point.

3.1.1 Unités

On peut choisir l’unité dans laquelle les angles sont mesurés au moyen de l’option *angles*. Les valeurs possibles sont *degrees* et *radians*.

L’option *decimalsign* permet de définir le symbole à utiliser pour représenter le séparateur des valeurs décimales (en général une virgule ou un point). La valeur de cette option est un symbole ou bien le mot-clé *locale* suivi du nom de la locale à utiliser pour définir ce symbole. Voici deux exemples qui ont pour effet de déclarer la virgule comme séparateur décimal :

```
set decimalsign ","
set decimalsign locale "fr_FR"
```

L’option *zero* indique le seuil en-dessous duquel une valeur est considérée comme nulle.

```
gnuplot> show zero
zero is 1e-08
```

3.1.2 Dates

L’option *timefmt* est utilisée pour les séries chronologiques (*time series*). Elle n’a de sens que si l’instruction *set xdata time* a été spécifiée. Elle permet de déclarer le format à utiliser pour lire des données représentant une date et une heure. La syntaxe est :

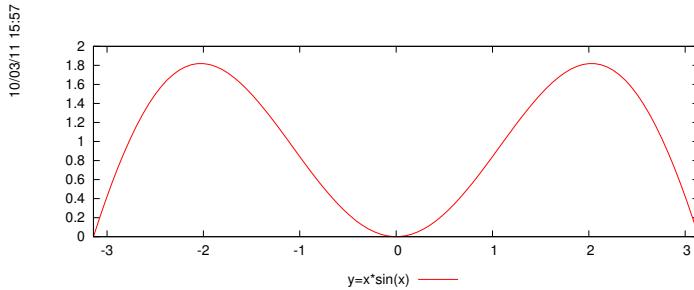


FIGURE 2 – Exemple d’option *timestamp*.

```
set timefmt "<format string>"  
show timefmt
```

Le format est une chaîne de caractères utilisant les symboles appelés spécificateurs rassemblés dans le tableau 3.

Par exemple :

```
set timefmt "%d/%m/%Y %H:%M"
```

L’option *timestamp* permet d’inclure la date dans la marge de gauche. La syntaxe est :

```
set timestamp {"<format>"} {top/bottom} {{no}rotate}  
    {offset {<xoff>} {<yoff>}} {font "<fontspec>"}  
unset timestamp  
show timestamp
```

Les arguments ont la signification suivante :

- *top* et *bottom* indiquent l’emplacement de la date
- *rotate* permet d’écrire la date verticalement
- *offset* permet de positionner en un point donné
- *font* permet de spécifier la police à utiliser

Voici un exemple :

```
set timestamp "%a/%m/%y %H:%M" rotate top  
plot [-pi:pi] x*sin(x)
```

Le graphe est représenté sur la figure 2.

L’option *locale* permet de déclarer la locale à utiliser afin que les options *xtics*, *ytics*, etc. écrivent les jours et les mois selon l’usage en vigueur. La syntaxe est

```
set locale "nom_de_la_locale"
```

Par défaut, la valeur utilisée est celle fournie par les variables d’environnement `LC_TIME`, `LC_ALL` ou `LANG`.¹

La section 8.1 en annexe comporte des informations complémentaires concernant les dates et les heures.

1. Le choix d’une locale ne suffit pas pour modifier automatiquement le format des nombres : pour modifier le symbole de séparation des chiffres décimaux, il faut utiliser l’option *decimalsign*.

3.1.3 Chemins d'accès

L'option *fontpath* permet de définir des chemins d'accès supplémentaires pour trouver des définitions de polices. Cette option n'est utilisée actuellement que par le terminal *postscript*. La syntaxe est :

```
set fontpath {"pathlist1" {"pathlist2"..."}}  
show fontpath
```

Par exemple :

```
gnuplot> show fontpath  
fontpath is  
system fontpath is "/usr/X11R6/lib/X11/fonts/Type1"
```

L'option *loadpath* permet de définir des emplacements supplémentaires pour rechercher des fichiers de données ou des scripts. Elle est utilisée par les commandes **call**, **load**, **plot** et **splot**. La syntaxe est :

```
set loadpath {"pathlist1" {"pathlist2"..."}}  
show loadpath
```

3.1.4 Informations d'état

Les options suivantes sont en lecture seule et permettent d'obtenir des informations sur le statut courant du programme. On les affiche au moyen de la commande **show**.

- l'option *version* renvoie la bannière initiale affichée par **gnuplot**. Elle comporte le numéro de version, les informations de copyright, etc. Si on rajoute le mot-clé *long* on obtient en plus des informations sur les options de compilation.
- l'option *variables* renvoie la liste de toutes les variables définies par l'utilisateur et leur valeur. Avec le mot-clé *all* on obtient en plus les variables internes de **gnuplot** préfixées avec **GPVAL_**.
- l'option *functions* renvoie la liste de toutes les fonctions définies par l'utilisateur et leur définition.
- l'option *plot* renvoie le contenu de la dernière commande **plot**.

Par exemple :

```
show version  
show version long  
show variables  
show functions  
show plot
```

3.2 Options concernant les entrées et sorties

3.2.1 Sorties graphiques

Le programme **gnuplot** utilise la notion de terminal pour diriger les sorties des commandes de dessin. Un terminal est une abstraction pour désigner un périphérique de sortie dans lequel écrire les commandes de dessin. Ces commandes de dessin seront très différentes selon le terminal choisi : si le terminal est *png*, le

graphique sera produit comme un fichier d'image au format PNG. Si le terminal est *postscript*, le graphique sera produit comme une suite d'instructions en langage PostScript stockées dans un fichier. Le terminal peut aussi bien désigner un fichier qu'un mode d'affichage à l'écran : si le terminal est *x11*, le graphique sera affiché à l'écran dans une fenêtre X11 (sous Unix).

Lorsque **gnuplot** démarre, il sélectionne un périphérique de sortie par défaut et il l'indique dans son message initial. Par exemple :

```
Terminal type set to 'aqua'
```

On modifie le terminal de sortie au moyen de l'option *terminal* dont la syntaxe est

```
set terminal [type_de_terminal]
```

Par exemple :

```
set terminal x11
set terminal aqua
set terminal png
```

Pour connaître la liste de tous les types de terminal disponibles pour une installation de **gnuplot**, on peut utiliser la commande *set terminal* sans argument supplémentaire. Voici un extrait de la liste obtenue :

```
Available terminal types:
aed512 AED 512 Terminal
aed767 AED 767 Terminal
aifm Adobe Illustrator 3.0 Format
aqua Interface to graphics terminal server for Mac OS X
bitgraph BBN Bitgraph Terminal
cgm Computer Graphics Metafile
corel EPS format for CorelDRAW
dumb ascii art for anything that prints text
dxf dxf-file for AutoCad (default size 120x80)
eepic EEPIC -- extended LaTeX picture environment
emf Enhanced Metafile format
emtex LaTeX picture environment with emTeX specials
epslatex LaTeX picture environment using graphicx package
...
tkcanvas Tk/Tcl canvas widget [perlTk] [interactive]
tpic TPIC -- LaTeX picture environment with tpic \specials
unknown Unknown terminal type - not a plotting device
vttek VT-like tek40xx terminal emulator
x11 X11 Window System
xlib X11 Window System (gnulib_x11 dump)
```

Dans le cas d'une sortie vers un périphérique de type fichier, on peut aussi spécifier le nom du fichier au moyen de l'option *output* dont la syntaxe est :

```
set output "nom_du_fichier"
```

Le fichier est créé dans le répertoire courant. Pour connaître le répertoire courant, on utilise la commande **pwd** et pour le modifier la commande **cd**² (voir la section 7.1).

2. Comme les commandes Unix de même nom.

Il est recommandé d'utiliser la commande **set terminal** avant la commande **set output**. Par exemple :

```
set terminal png
set output "test.png"
plot sin(x)
```

Une commande **set terminal** peut aussi comporter des options supplémentaires après le nom du terminal. Par exemple, les instructions suivantes ajoutent les options *eps* et *enhanced* dans le cas d'un terminal *postscript* :

```
set terminal postscript eps enhanced
```

Ces options sont variables selon le type de terminal utilisé et il faut consulter la documentation de chaque type de terminal pour connaître les options disponibles. On peut pour cela utiliser la commande **help**. Par exemple, voici le début de l'aide concernant le terminal *png* :

```
gnuplot> help png
Syntax:
    set terminal png
        {{no}transparent} {{no}interlace}
        {{no}truecolor} {rounded/butt}
        {tiny / small / medium / large / giant}
        {font <face> {<fontsize>}}
        {size <x>,<y>} {{no}crop}
        {{no}enhanced}
        {<color0> <color1> <color2> ...}
...
...
```

Voici un exemple de déclaration pour une sortie vers un fichier *png* :

```
set terminal png transparent enhanced size 1024,768
```

L'option *termoption* permet de modifier certaines valeurs sans avoir à déclarer à nouveau le périphérique de sortie. Seulement quelques valeurs peuvent ainsi être modifiées.

```
set termoption {no}enhanced
set termoption font "<fontname>{,<fontsize>}"
```

3.2.2 Redirections

L'option **print** permet de rediriger les valeurs renvoyées par la commande **print**. Par défaut, la commande **print** écrit sur le canal d'erreur *stderr*. Si on exécute la commande

```
set print '-'
```

c'est le canal de sortie standard *stdout* qui sera utilisé.

On peut aussi spécifier un fichier ordinaire. Par exemple

```
set print "resultats.txt"
```

Le mot-clé *append* indique que les commandes **print** ajoutent leurs résultats à la fin du fichier existant plutôt que de l'effacer :

```
set print "resultats.txt" append
```

Enfin, on peut aussi envoyer les résultats de la commande **print** comme flux d'entrée d'une autre commande avec le symbole `|` pour les systèmes de type Unix. Par exemple :

```
set print '| lpr'
```

3.2.3 Sorties tabulées

L'option *table* fait en sorte que **gnuplot** écrive les coordonnées des points produits par une commande **plot** ou **splot** sous forme d'une table dont les colonnes représentent les coordonnées *x*, *y* et, en 3D, *z*. Une colonne constituée de caractères *i*, *o* ou *u* est ajoutée à la table : la lettre *i* indique qu'un point est à l'intérieur de l'intervalle de valeurs actif, *o* qu'il est à l'extérieur et *u* que c'est indéterminé.

Par exemple :

```
set table "table.txt"
plot [-pi:pi] sin(x)
unset table
```

Voici le début du fichier produit :

```
# Curve 0 of 1, 100 points
# Curve title: "sin(x)"
# x y type
-3.14159 -1.22465e-16 i
-3.07813 -0.0634239 i
-3.01466 -0.126592 i
-2.95119 -0.189251 i
-2.88773 -0.251148 i
-2.82426 -0.312033 i
etc.
```

Le fichier obtenu est au format standard des fichiers de données de **gnuplot** (voir à la section 4.1.1) et peut donc être réutilisé par la suite dans d'autres commandes **plot** ou **splot**.

3.3 Options concernant les axes

3.3.1 Origine

L'option *origin* permet de contrôler l'emplacement de l'origine du système de coordonnées sur le canevas. L'emplacement est indiqué en coordonnées d'écran.³ Par exemple, l'instruction suivante place l'origine au centre du graphique :

```
set origin 0.5,0.5
```

3. Coordonnées de type *screen*. Voir au paragraphe 2.2.

Pour un graphique 3D, l'option *xyplane* permet d'autre part de déterminer l'emplacement du plan horizontal (plan des x, y) par rapport à l'axe des z . La syntaxe peut prendre deux formes :

```
set xyplane <frac>
set xyplane at <zvalue>
```

Un argument *<frac>* place le plan horizontal sous l'intervalle des z et la fraction indique la distance du plan à la valeur minimale de z par rapport à l'étendue totale de l'intervalle des z . La valeur par défaut de cette fraction est 0,5 : cela signifie que la distance du plan horizontal à la valeur z_{min} est la moitié de la longueur $z_{max} - z_{min}$ de l'intervalle des z .

Dans la deuxième forme, l'argument *<zvalue>* indique une valeur spécifique de la variable z . Par exemple :

```
set xyplane at 10
```

placera le plan horizontal à la cote $z = 10$, quel que soit l'intervalle des valeurs prises par les z dans le graphique.

3.3.2 Échelle

L'option *autoscale* indique à **gnuplot** de calculer lui-même pour chaque axe l'intervalle de valeurs à utiliser. Si l'option est active pour un axe, **gnuplot** détermine automatiquement un intervalle raisonnable pour les variables indépendantes (x pour un graphique 2D et (x, y) pour un graphique 3D) et choisit un intervalle approprié pour les variables dépendantes (y pour un graphique 2D et z pour un graphique 3D). Cette option est active par défaut pour chaque axe, y compris les axes secondaires :

```
gnuplot> show autoscale
    autoscaling is      x: ON,      y: ON,
                      x2: ON,     y2: ON,
                      z: ON,      cb: ON,
```

On peut la désactiver pour certains axes afin d'imposer d'autres valeurs directement sous forme d'un intervalle $[a : b]$ dans les commandes **plot** et **splot**. La syntaxe est :

```
unset autoscale {<axe>}
```

où l'argument *<axe>* peut être $x, y, z, cb, x2, y2$ ou xy .

Inversement, pour spécifier l'option, on utilise la syntaxe :

```
set autoscale {<axe>}
```

Dans ce cas, le nom de l'axe peut comporter un des suffixes suivants :

- *min* ou *max* afin que **gnuplot** applique *autoscale* seulement au minimum ou au maximum de l'axe plutôt qu'aux deux à la fois ;
- *fixmin*, *fixmax* ou *fix* pour désactiver l'extension des intervalles à la prochaine marque de graduation (*tic*) d'un côté ou de l'autre de l'intervalle, dans le cas où le graphique utilise des graduations équidistantes.

Par exemple :

```
set autoscale x
set autoscale ymin
set autoscale x2fixmin
```

L'option *logscale* permet d'utiliser une échelle logarithmique sur un axe plutôt que l'échelle linéaire usuelle. La syntaxe est :

```
set logscale <axes> <base>
unset logscale <axes>
show logscale
```

où l'argument *<axes>* peut être n'importe quelle combinaison de *x*, *x2*, *y*, *y2*, *z*, *cb* et l'argument optionnel *base* désigne la base de logarithme à utiliser. Par défaut, il s'agit de logarithmes décimaux (base 10). Par exemple :

```
set logscale xyz 2
```

impose l'usage d'une échelle logarithmique en base 2 pour chacun des axes *x*, *y* et *z*.

3.3.3 Axes du repère

Les axes du repère sont les droites passant par l'origine et représentant le système de coordonnées. Ils ne doivent pas être confondus avec les bords du graphique qui portent les graduations. Par défaut, ces axes ne sont pas tracés comme on peut le voir avec la commande suivante :

```
gnuplot> show zeroaxis
xzeroaxis is OFF
x2zeroaxis is OFF
yzeroaxis is OFF
y2zeroaxis is OFF
zzeroaxis is OFF
z2zeroaxis is OFF
```

Pour qu'un axe particulier soit dessiné, il faut utiliser l'une des options *xzeroaxis*, *x2zeroaxis*, *yzeroaxis*, *y2zeroaxis*, *zzeroaxis*.

Ces options supportent aussi des arguments *linestyle*, *linetype* et *linewidth* permettant de spécifier respectivement un style, ou le type et l'épaisseur des axes. Par exemple :

```
set xzeroaxis linetype 2 linewidth 1.5
set yzeroaxis linetype 3 linewidth 1.5
```

3.3.4 Marques de graduation

L'option principale permettant de placer des marques de graduation sur les axes est l'option *tics*. Elle agit sur tous les axes à la fois. Si on veut opérer des réglages pour un axe particulier, il existe des commandes spécifiques obtenues en mettant le nom de l'axe en préfixe : les options s'appellent alors *xtics*, *ytics*, *x2tics*, *y2tics* et *ztics*.

On distingue deux types de marques : les marques majeures qui portent une valeur numérique et les marques mineures qui sont simplement un petit trait marquant une sous-graduation. Les marques mineures sont contrôlées par des options similaires mais commençant par la lettre *m* : *mxtics*, *mytics*, *mx2tics*, *my2tics* et *mztics*.

La syntaxe complète de l'option *tics* est :

```
set tics {axis | border} {{no}mirror}
    {in | out} {scale {default | <major> {,<minor>}}}
    {{no}rotate {by <ang>}} {offset <offset> | nooffset}
    {format "formatstring" } { font "name{,<size>}" }
    { textcolor <colorspec> }
set tics {front | back}
unset tics
show tics
```

Les marques peuvent figurer sur les bords du graphe ou sur les axes de coordonnées si ceux-ci sont représentés : on utilise pour cela l'argument *border* ou *axis* respectivement. Par exemple :

```
set xtics axis
set ytics border
set zeroaxis lt 8
plot [-2*pi:2*pi] sin(x)
```

D'autres arguments permettent de régler certaines caractéristiques :

- l'argument *mirror* dessine des marques (sans la valeur numérique) en vis-à-vis sur le bord opposé. Si ce comportement n'est pas souhaité, on utilisera au contraire l'option *nomirror* ;
- les arguments *in* et *out* ont pour effet de dessiner les marques vers l'intérieur ou vers l'extérieur du graphique ;
- l'argument *rotate* fait pivoter le texte porté par la graduation de 90 degrés. On peut aussi spécifier un angle particulier en ajoutant un argument *by* comme, par exemple, *rotate by 45* ;
- l'argument *offset* permet de décaler le texte porté par la graduation par rapport à sa position par défaut. La valeur de cet argument prend la forme *x,y* ou *x,y,z* avec des valeurs exprimées dans un des systèmes de coordonnées reconnus par **gnuplot** (voir paragraphe 2.2).

Les réglages par défaut pour les axes des *x* et des *y* correspondent aux arguments *border*, *mirror* et *norotate*.

Il existe aussi des options qui transforment les marques en jours de la semaine, de 0 pour le dimanche jusqu'à 6 pour le samedi. Ces options comportent la lettre *d* après le nom de l'axe : *xdtics*, *ydtics*, *x2dtics*, *y2dtics* et *zdtics*.

Des options similaires transforment les marques en mois de l'année, de 1 pour janvier jusqu'à 12 pour décembre. Ces options comportent la lettre *m* après le nom de l'axe : *xmtics*, *ymtics*, *x2mtics*, *y2mtics* et *zmtics*.

Le programme est capable de déterminer automatiquement des marques de graduation mais il est aussi possible de les indiquer explicitement avec chacune des options *xtics*, *ytics*, etc. Plusieurs syntaxes sont supportées. En voici quelques exemples :

```
set xtics ("x1" 0, "x2" 50, "x3" 100)
set xtics (1,2,4,8,16,32,64,128,256,512,1024)
```

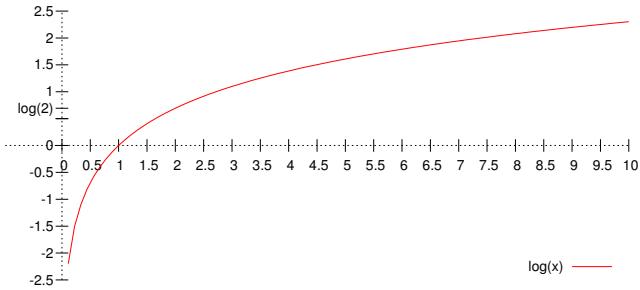


FIGURE 3 – Exemple de *tics*.

```
set ytics ("min" 0, "" 10, "max" 20)
set ytics ("min" 0, "" 10 1, "max" 20)
set xtics 0,.5,10
```

Dans le troisième exemple, la deuxième marque est simplement dessinée (sans texte). Dans le quatrième exemple, le chiffre 1 indique que c'est une marque mineure. Le cinquième exemple produit des marques allant de 0 à 10 par pas de 0,5 unités.

D'autres marques peuvent être ajoutées ponctuellement avec un argument *add*, comme par exemple :

```
set ytics add ("log(2)" 0.693)
```

Voici un exemple complet utilisant ces diverses options :

```
set xtics axis 0,0.5,10
set ytics axis
set border 0
set xrange [-1:10]
set xzeroaxis lw 1.5
set yzeroaxis lw 1.5
set ytics add ("log(2)" 0.693, "" 0.5)
plot log(x)
```

Le graphe est représenté sur la figure 3.

Le format adopté pour écrire les valeurs numériques sur les marques majeures est spécifié au moyen de l'option *format*. La syntaxe est :

```
set format {<axe>} {"<format>"}
```

L'expression décrivant le format utilise le même format que la fonction *printf* du langage C mais **gnuplot** a étendu cette syntaxe et offre des spécificateurs supplémentaires (voir la section 8.2).

Si on veut supprimer les marques sur les axes secondaires, il faut utiliser les commandes suivantes :

```
set xtics nomirror
set ytics nomirror
```

et non pas *unset x2tics* comme on pourrait le penser intuitivement. En effet, en-dehors de tout réglage spécifique au moyen des options *x2tics* ou *y2tics*, les marques dessinées par défaut sur les axes secondaires sont seulement des miroirs des marques des axes principaux.

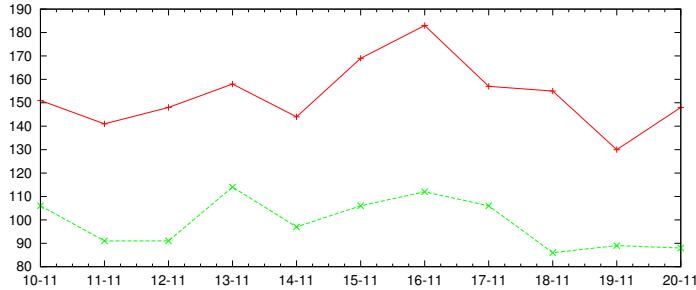


FIGURE 4 – Exemple d’option *xdata*.

3.3.5 Données temporelles

Les options *ydata*, *xdata*, *x2data*, *y2data* et *zdata* permettent d’indiquer que les valeurs numériques le long des axes *x*, *y*, *x2*, *y2* ou *z* respectivement sont de type temporel, autrement dit représentent une date et/ou une heure.

Elles concernent les données qui sont lues depuis un fichier afin d’indiquer à **gnuplot** comment lire certaines colonnes. Ces options fonctionnent avec l’option *timefmt* qui précise le format adopté pour ces données. On ne peut spécifier qu’un seul format à la fois, ce qui signifie que toutes les données concernées doivent se conformer exactement à ce format.

Par exemple, supposons qu’un fichier *data.txt* contienne les données suivantes :

2009-11-10	151	106
2009-11-11	141	91
2009-11-12	148	91
2009-11-13	158	114
2009-11-14	144	97
2009-11-15	169	106
2009-11-16	183	112
2009-11-17	157	106
2009-11-18	155	86
2009-11-19	130	89
2009-11-20	148	88

On tracera le graphe correspondant aux colonnes 2 et 3 par rapport à la colonne 1 au moyen des instructions suivantes :

```
set xdata time
set timefmt "%Y-%m-%d"
set xtics format "%d-%m"
set datafile separator "\t"
plot 'data.txt' using 1:2 with linespoints,
      'data.txt' using 1:3 with linespoints
```

Le graphe est représenté sur la figure 4.

3.3.6 Orientation des axes

L’option *view* contrôle l’orientation du repère dans un graphique 3D, c’est-à-dire l’angle sous lequel la surface représentée est “vue”. Une représentation en 3D est une *projection orthogonale* sur le plan du canevas. La syntaxe peut prendre plusieurs formes :

```
set view <rot_x>{,{<rot_z>}{{<scale>}{{<scale_z>}}}}
set view map
set view {no}equal <axes>
```

Les arguments *rot_x*, *rot_z*, *scale* et *scale_z* déterminent les angles de rotation et l'échelle. Par défaut on a les valeurs suivantes :

```
gnuplot> show view
view is 60 rot_x, 30 rot_z, 1 scale, 1 scale_z
axes are independently scaled
```

On les interprète de la manière suivante. Au départ, le repère est supposé vu avec l'axe des *x* horizontal et l'axe des *y* vertical comme dans un graphique 2D, et l'axe des *z* est censé être perpendiculaire au canevas. On applique ensuite une rotation de *rot_x* degrés autour de l'axe des *x* puis une rotation de *rot_z* degrés autour de l'axe des *z*. Le repère est alors projeté orthogonalement sur le plan du canevas. Les valeurs de *rot_x* sont comprises entre 0 et 180 (si les angles sont mesurés en degrés), celles de *rot_z* entre 0 et 360.

La valeur *scale* permet de modifier l'échelle du graphique dans son ensemble tandis que *scale_z* concerne seulement l'échelle relative à l'axe des *z*. Ces valeurs sont optionnelles et valent 1 par défaut. Par exemple, pour diviser la taille du graphique par 2 sans modifier l'angle de vue, on utilise l'instruction :

```
set view ,0.5
```

La commande *set view map* projette la surface directement sur le plan des axes *x* et *y*, ce qui revient à ne faire aucune rotation.

La commande *set view equal* force certains axes à utiliser la même échelle. Normalement l'échelle adoptée par défaut par **gnuplot** est différente sur les trois axes : elle est calculée automatiquement afin que les valeurs tiennent dans le canevas. Avec une instruction

```
set view equal xy
```

on force les axes des *x* et des *y* à adopter la même échelle. Avec une instruction

```
set view equal xyz
```

on force les trois axes (*x*, *y* et *z*) à adopter la même échelle. Dans ce cas, les valeurs des *z* risquent de sortir en partie du canevas.

3.4 Options concernant les coordonnées

3.4.1 Nom des coordonnées

Par défaut, en coordonnées cartésiennes, les variables sont désignées par les lettres *x* pour une courbe et (*x*, *y*) pour une surface. Si on utilise des coordonnées paramétriques, les coordonnées, par défaut, sont *u* pour une courbe et (*u*, *v*) pour une surface. Une courbe peut aussi être définie en coordonnées polaires et la variable par défaut, dans ce cas, est *t* qui représente l'angle polaire.

On peut modifier ces conventions au moyen de l'option *dummy*. La syntaxe de cette option est :

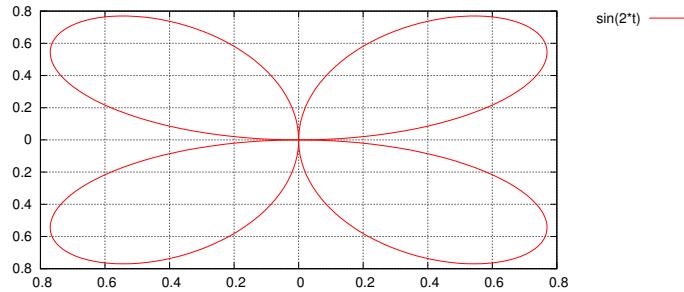


FIGURE 5 – Exemple d'option *polar*.

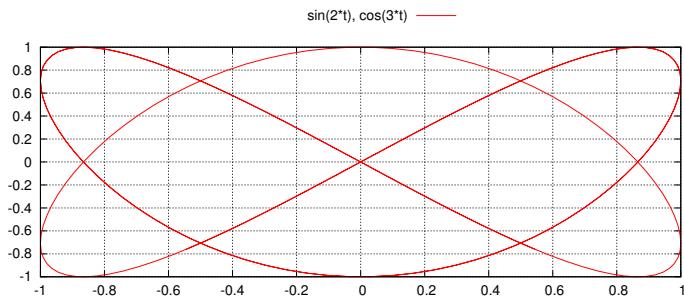


FIGURE 6 – Exemple de courbe en mode paramétrique.

```
set dummy {<dummy-var>} {<dummy-var>}
show dummy
```

Par exemple :

```
gnuplot> set dummy r,s
gnuplot> show dummy
    dummy variables are "r" and "s"
gnuplot> set polar
    dummy variable is t for curves
gnuplot> set dummy a
gnuplot> show dummy
    dummy variables are "a" and "s"
```

3.4.2 Systèmes de représentation

Les options *polar* et *parametric* permettent de passer respectivement en coordonnées polaires ou en coordonnées paramétriques. Par exemple :

```
set polar
plot sin(2*t)
```

Le graphe est représenté sur la figure 5.

En mode paramétrique, dans le cas d'une courbe, on spécifie une paire de valeurs représentant les coordonnées (x, y) exprimées en fonction du paramètre t . Dans le cas d'une surface, on spécifie un triplet de valeurs représentant les coordonnées (x, y, z) en fonction de t . Voici un exemple d'utilisation de l'option *parametric* pour une courbe :

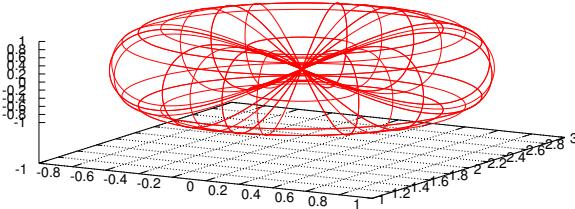


FIGURE 7 – Exemple d’option *parametric* avec *splot*.

```
set grid xtics ytics
set samples 300
set parametric
plot sin(2*t), cos(3*t)
```

Le graphe est représenté sur la figure 6. L’exemple suivant illustre l’option *parametric* dans le cas d’une surface :

```
set parametric
splot cos(u)*cos(v), 2-cos(u)*sin(v), sin(2*u)
```

Le graphe est représenté sur la figure 7.

On ne peut pas, avec **gnuplot**, définir une surface au moyen d’une formule exprimée en coordonnées sphériques ou en coordonnées cylindriques mais, en revanche, un fichier de données numériques peut contenir des points exprimés dans un de ces systèmes. C’est l’option *mapping* qui permet de spécifier le type de coordonnées à utiliser lorsque les données sont lues depuis un fichier. La syntaxe est :

```
set mapping {cartesian / spherical / cylindrical}
```

On indique donc l’un des mots-clés *cartesian*, *spherical* ou *cylindrical* selon que les données doivent être interprétées comme des coordonnées cartésiennes, sphériques ou cylindriques.

En coordonnées sphériques, le fichier de données peut comporter deux ou trois colonnes. Les deux premières colonnes représentent les angles polaires θ et ϕ . La troisième colonne représente les valeurs du rayon vecteur r ; si elle est absente, **gnuplot** considère que $r = 1$. La correspondance avec les coordonnées cartésiennes est définie par :

$$\begin{cases} x = r \times \cos \theta \times \cos \phi \\ y = r \times \sin \theta \times \cos \phi \\ z = r \times \sin \phi \end{cases}$$

En coordonnées cylindriques, le fichier de données peut aussi comporter deux ou trois colonnes. Les deux premières colonnes représentent l’angle polaire θ et la cote z . La troisième colonne représente les valeurs du rayon vecteur r ; si elle

est absente, **gnuplot** considère que $r = 1$. La correspondance avec les coordonnées cartésiennes est définie par :

$$\begin{cases} x = r \times \cos \theta \\ y = r \times \sin \theta \\ z = z \end{cases}$$

3.4.3 Intervalles de valeurs

On peut limiter, pour chaque variable, l'intervalle sur lequel est effectuée la représentation graphique. Selon la variable concernée, on dispose des options suivantes :

- *xrange*, *yrange* et *zrange* pour délimiter les valeurs en coordonnées cartésiennes sur les axes des x , des y et des z respectivement ;
- *x2range* et *y2range* pour délimiter les axes secondaires : l'axe des $x2$ est le côté supérieur et l'axe des $y2$ le côté droit ;
- *rrange* pour délimiter les valeurs du rayon vecteur en coordonnées polaires ;
- *trange* permet de délimiter les valeurs du paramètre t pour une courbe plane en coordonnées paramétriques ou celles de l'argument dans le cas d'une courbe en coordonnées polaires ;
- *urange* et *vrange* pour une surface en coordonnées paramétriques.

Si aucun intervalle de valeur n'est spécifié, **gnuplot** calcule automatiquement des valeurs par défaut raisonnables.

L'option *offsets* permet de modifier les valeurs d'intervalle calculées automatiquement lorsque les options telles que *xrange* ou *yrange* n'ont pas été spécifiées. La syntaxe est :

```
set offsets <left>, <right>, <top>, <bottom>
unset offsets
show offsets
```

Par exemple, si on trace le graphe de la fonction $\cos(x)$, l'intervalle de la variable y sera automatiquement fixé à $[-1 : 1]$. On peut l'étendre à $[-1.5 : 1.8]$ avec l'instruction suivante :

```
set offsets 0,0,0.8,0.5
```

3.5 Options concernant les titres et légendes

3.5.1 Titres

L'option *title* permet de déclarer un titre général pour le graphique. Ce titre est placé, par défaut, dans la marge supérieure du canevas. La syntaxe est :

```
set title {"<title-text>"} {offset <offset>} {font "<font>{,<size>}"}
    {{textcolor | tc} {<colorspec> | default}} {{no}enhanced}
show title
```

L'argument optionnel *offset* permet de déplacer le titre par rapport à sa position par défaut. La valeur du décalage est exprimée sous la forme x, y ou x, y, z et peut être précédée des mots-clés *first*, *second*, *graph*, *screen* ou *character* afin

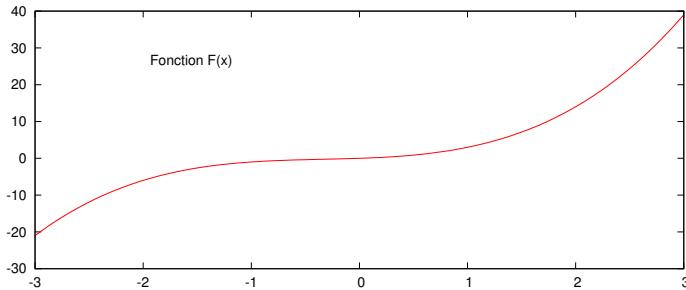


FIGURE 8 – Exemple d’option *title offset*.

de préciser le système de coordonnées dans lequel ces valeurs sont exprimées.⁴ Dans l’exemple suivant le titre est décalé de 20 caractères vers la gauche et de 5 lignes sous le bord supérieur :

```
set title offset -20,-5 "Fonction F(x)"
plot [-3:3] x**3+x**2+x
```

Le graphe est représenté sur la figure 8.

3.5.2 Étiquettes

Chacun des axes d’un graphique (axes principaux x , y , z et axes secondaires $x2$, $y2$) peut être étiqueté par un texte. On déclare ces étiquettes au moyen des options suivantes :

- *xlabel*
- *ylabel*
- *zlabel*
- *x2label*
- *y2label*

Ces options supportent les mêmes arguments que l’option *title*, en particulier *offset*, *font*, *textcolor*.

Il existe aussi un argument *rotate* qui permet de faire pivoter d’un certain angle le texte attaché à un axe. Cette fonction n’est pas supportée par tous les types de périphériques de sortie.

Voici un exemple simple qui crée des légendes pour chacun des quatre axes :

```
set xlabel "Label sur l'axe des x"
set ylabel "Label sur l'axe des y"
set x2label "Label sur l'axe des x2"
set y2label "Label sur l'axe des y2"
plot [-pi:pi] sin(x) + cos(2*x)
```

Le graphe est représenté sur la figure 9.

On peut aussi placer des indications sur le graphique lui-même au moyen de l’option *label*. La syntaxe complète de cette option est :

4. Par défaut le système *character* est utilisé. Voir la section 2.2 concernant les systèmes de coordonnées.

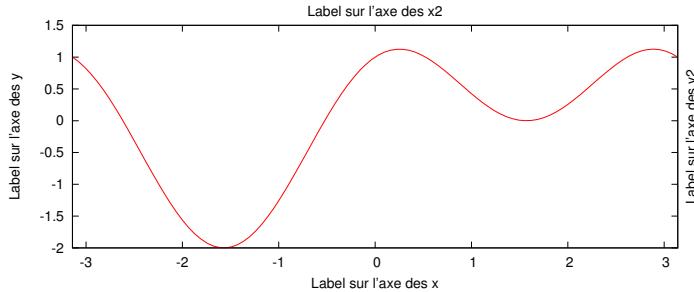


FIGURE 9 – Exemple d'options *xlabel*, *ylabel*, *x2label*, *y2label*.

```
set label {<tag>} {"<label text>"} {at <position>}
    {left | center | right}
    {norotate | rotate {by <degrees>}}
    {font "<name>{,<size>}"}
    {noenhanced}
    {front | back}
    {textcolor <colorspec>}
    {point <pointstyle> | nopoint}
    {offset <offset>}
unset label {<tag>}
show label
```

Dans sa forme la plus simple, on l'utilisera comme ceci :

```
set label "texte" at x,y
```

où *x* et *y* désignent les coordonnées du point où sera ajouté le texte sur le graphe.

Il peut y avoir autant d'instructions *label* qu'on le souhaite. Si on veut pouvoir se référer par la suite à l'une d'entre elles en particulier, il est possible de lui affecter un identifiant : il s'agit d'un *tag* qui n'apparaît pas sur le graphique et sert uniquement à identifier un label afin, par la suite, de modifier ses caractéristiques dans un script. Dans l'exemple suivant, trois labels sont placés sur le graphique et sont identifiés au moyen de tags 1, 2 et 3 respectivement :

```
set label 1 "point d'infexion en (0,0)" at 0,1
set label 2 "zone y > x^3" at -1.5,4
set label 3 "zone y < x^3" at 0.5,-5
plot [-2:2] x**3
```

Le graphe est représenté sur la figure 10. Si on veut ensuite modifier la couleur de la deuxième étiquette pour qu'elle s'affiche en rouge, il suffira d'écrire :

```
set label 2 textcolor rgb 'red'
replot
```

L'option *clabel* concerne les courbes de niveau obtenues avec l'option *contour*. Lorsque cette option est active, chaque ligne de niveau est dessinée avec un type de ligne différent et une légende vient préciser le style utilisé pour chaque ligne de niveau représentée. Par exemple, par défaut, on a les réglages suivants :

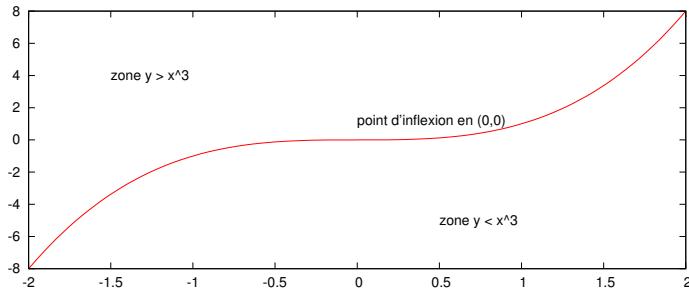


FIGURE 10 – Exemple d'options *label* multiples.

```
gnuplot> set contour
gnuplot> show clabel
    contour for surfaces are drawn in 5 levels on grid base
        as linear segments
        approx. 5 automatic levels
    contour line types are varied & labeled with format '%8.3g'
```

3.5.3 Légendes

L’option *key* contrôle les légendes qui accompagnent un graphique : il s’agit d’un bloc de spécimens indiquant quel style (type de ligne, de point, de couleur etc.) est associé à chaque courbe sur le graphique. Cette option supporte de nombreux arguments pour spécifier les caractéristiques de la légende, son emplacement, sa taille, etc. On peut obtenir les réglages courants au moyen de l’instruction *show key* comme ceci :

```
gnuplot> show key
    key is ON, position: top right vertical inside
    key is right justified, not reversed, not inverted,
        enhanced and not boxed
    sample length is 4 characters
    vertical spacing is 1 characters
    width adjustment is 0 characters
    height adjustment is 0 characters
    curves are automatically titled with filename
    key title is ""
```

Voici quelques exemples simples. Pour supprimer la légende, la commande est :

```
set key off
```

Pour faire en sorte que la légende soit placée en-dehors du graphique, la commande est :

```
set key on outside
```

La syntaxe complète de l’option *key* est :

```
set key {on/off} {default}
    {{inside | outside} | {lmargin | rmargin | tmargin | bmargin}
    | {at <position>}}
```

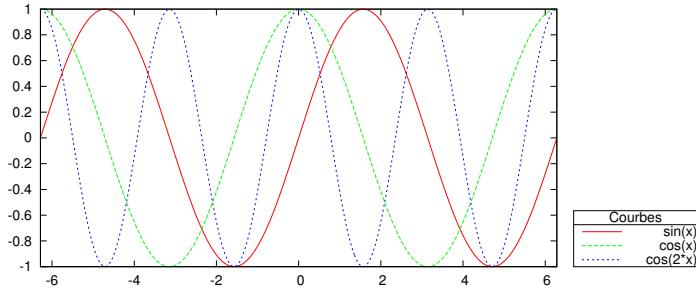


FIGURE 11 – Exemple d’option *key*.

```

{left | right | center} {top | bottom | center}
{vertical | horizontal} {Left | Right}
{{no}reverse} {{no}invert}
{samplen <sample_length>} {spacing <vertical_spacing>}
{width <width_increment>}
{height <height_increment>}
{{no}autotitle {columnheader}}
{title "<text>"} {{no}enhanced}
{{no}box { {linestyle | ls <line_style>}
           | {linetype | lt <line_type>}
           {linewidth | lw <line_width>}}}

unset key
show key

```

Les mots-clés *inside* et *outside* indiquent si la légende doit être placée à l’intérieur ou à l’extérieur du graphique. Les mots-clés *above*, *over*, *below*, *under* précisent la position par rapport au graphique. On peut aussi ajouter l’un des mots-clés *left*, *right*, *top*, *bottom* qui précisent l’alignement par rapport aux marges du graphique. Les mots-clés *vertical* et *horizontal* indiquent si les légendes doivent être empilées verticalement ou juxtaposées horizontalement lorsqu’il y en a plusieurs.

On peut aussi positionner une légende à un emplacement précis avec le mot-clé *at*. Dans ce cas, les mots-clés *left*, *right* et *center* précisent l’ajustement par rapport au point.

Le mot-clé *reverse* intervertit étiquette et échantillon.

Pour tracer un rectangle autour de la légende, on dispose du mot-clé *box*. Dans ce cas, les mots-clés *linestyle* (*ls*), *linetype* (*lt*) et *linewidth* (*lw*) affectent le tracé des côtés du rectangle.

Le bloc de légendes peut aussi recevoir un titre avec l’argument *title*.

Voici quelques exemples :

```

set key outside left
set key rmargin vertical
set key at 0,0
set key box reverse at 0,0 right
set key box lw 2 lc 4

```

Le graphe représenté sur la figure 11 est obtenu avec le code suivant :

```

set key box reverse rmargin vertical bottom
set encoding iso_8859_1
set key title "Courbes"

```

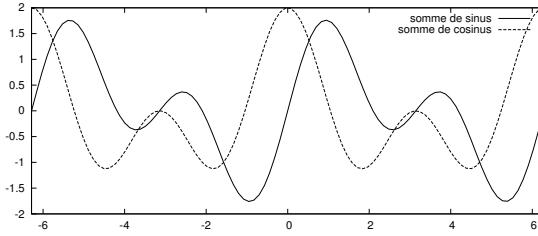


FIGURE 12 – Exemple d’option *title* dans une commande *plot*.

```
set samples 200
plot [-2*pi:2*pi] sin(x), cos(x), cos(2*x)
```

Le texte utilisé pour chaque courbe dans le bloc de légendes peut être contrôlé par une option de la commande **plot** qui s’appelle aussi *title*. Il faut bien distinguer cette option de l’argument *title* vu précédemment pour donner un titre global aux légendes (instruction *set key title*). Il s’agit ici d’un mot-clé qu’on peut ajouter à chaque courbe dans le cadre d’une commande **plot**. C’est le texte qui apparaîtra avec un spécimen du style graphique utilisé : par exemple, si le style est *lines*, une ligne de la couleur utilisée est dessinée après le texte. Par défaut, c’est le nom de la fonction représentée qui est utilisé pour le texte ou bien le nom du fichier lorsqu’il s’agit de données lues depuis un fichier. Voici un exemple :

```
set key on
plot [-2*pi:2*pi] sin(x)+sin(2*x) title "somme de sinus", \
cos(x)+cos(2*x) title "somme de cosinus"
```

Le graphe est représenté sur la figure 12.

On peut supprimer texte et échantillon pour une courbe particulière au moyen de l’option opposée *notitle*. Les options *title* et *notitle* sont attachées à une courbe, donc si plusieurs courbes figurent sur le même graphique certaines peuvent ainsi être marquées, d’autres pas.

Remarque : le mot-clé *title* peut apparaître dans trois contextes différents avec des significations différentes :

- l’option *title*, dans une instruction *set title*, définit le titre du graphique (voir la section 3.5) ;
- l’option *title*, dans une instruction *set key title*, définit le titre du bloc de légendes ;
- l’option *title*, dans une instruction *plot f(x) title*, définit l’intitulé de la légende associée à la courbe *f(x)*.

3.6 Options concernant les dimensions

Chaque graphique comporte une marge qui sépare le graphique proprement dit (à l’intérieur des axes de coordonnées) de son cadre. Cette marge est calculée automatiquement par **gnuplot** en fonction de divers éléments (titre, étiquettes sur les axes, etc.). Il y a quatre marges : en haut, en bas, à gauche, à droite. Elles sont désignées respectivement par les lettres *t* (pour *top*), *b* (pour *bottom*),

l (pour *left*) et *r* (pour *right*). Les valeurs de ces marges peuvent être imposées au moyen des options *tmargin*, *bmargin*, *lmargin* et *rmargin*. Les valeurs sont exprimées en nombres de caractères en hauteur ou en largeur. Par exemple :

```
set bmargin 5
```

signifie que la marge du bas peut contenir 5 lignes (c'est-à-dire cinq caractères en hauteur) tandis que

```
set lmargin 15
```

signifie que la marge de gauche peut contenir 15 caractères.

On peut obtenir leurs valeurs au moyen de l'option *margin*. Par exemple, avec les réglages précédents, on obtient :

```
gnuplot> show margin
lmargin is set to 15
bmargin is set to 5
rmargin is computed automatically
tmargin is computed automatically
```

L'option *size* permet de fixer les dimensions du graphe par rapport aux dimensions de l'image dans le périphérique de sortie (terminal). Les dimensions de l'image elle-même (appelée aussi canevas) sont fixées au moment où on déclare le périphérique de sortie par une commande telle que :

```
set terminal type_de_terminal size xx,yy
```

Ensuite, on utilise l'option *size* pour indiquer la taille du graphique *relativement* à celle du canevas. La syntaxe est

```
set size {ratio <r> | {no}square | noratio} {<xscale>,<yscale>}
show size
```

Les quantités *xscale* et *yscale* sont des valeurs relatives à la taille du canevas. Sous sa forme la plus simple, cette commande fixe simplement les valeurs de *xscale* et *yscale*. Des valeurs inférieures à 1 conduisent à un graphique qui ne remplit pas complètement l'image. Avec des valeurs supérieures à 1, le graphique sera tronqué car il excédera la taille du canevas. Par exemple :

```
set size 0.5,0.5
```

permet de créer un graphique qui occupe la moitié du canevas dans les deux directions *x* et *y*.

L'option *size* permet aussi de spécifier un rapport hauteur/largeur (rapport *y/x* appelé *aspect ratio* en anglais). On utilise pour cela le mot-clé *ratio* suivi de la valeur du rapport. Par exemple :

```
set size ratio 1.2
```

Cette instruction a pour effet que la dimension verticale est de 20% supérieure à la dimension horizontale. Le mot clé *square* indique un aspect ratio de 1. Pour annuler des réglages faits avec *ratio* ou *square*, on utilise *noratio* et *nosquare* comme ceci :

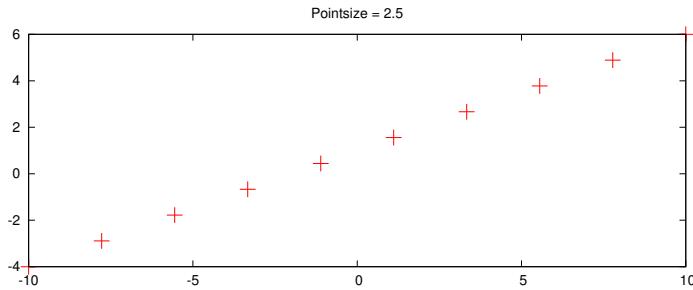


FIGURE 13 – Exemple d’option *pointsize*.

```
set size noratio
set size nosquare
```

Voici un exemple pour illustrer l’option *size* :

```
set xrange [-pi:pi]
set size square 0.5, 0.5
plot sin(2*x)
```

On peut aussi spécifier une valeur négative pour l’option *ratio*. Si on fixe *ratio* -1 , **gnuplot** essaie d’utiliser la même unité de longueur sur les deux axes. Plus généralement, si on fixe *ratio* $-n$ avec $n > 0$, il fait en sorte que l’unité de longueur sur l’axe des y soit n fois celle sur l’axe des x .

L’option *pointsize* permet d’ajuster la taille des points dans des graphiques de type bitmap. Par défaut, cette option vaut 1.0. Quel que soit le symbole utilisé pour représenter les points, leur taille peut être ainsi augmentée ou diminuée. Par exemple :

```
set samples 10
set pointsize 2.5
plot 0.5*x+1 with points
```

Le graphe est représenté sur la figure 13.

L’option *boxwidth* permet de spécifier la largeur des blocs lorsqu’on crée des graphiques de type *boxes*, *boxerrorbars*, *candlesticks* ou *histograms*. Elle peut être utilisée de manière relative ou absolue. Par exemple :

```
set boxwidth 0.5 relative
set boxwidth 2 absolute
```

3.7 Styles graphiques

La notion de style concerne ici le type et l’épaisseur des lignes ainsi que le type et la taille des points. Il s’agit des paramètres de dessin *linetype*, *linewidth*, *linecolor*, *pointsize*, *pointtype* ou, en abrégé, *lt*, *lw*, *lc*, *ps* et *pt*.

gnuplot offre la possibilité de définir des combinaisons de réglages de ces paramètres et de les utiliser par la suite au moyen de l’option *linestyle* ou, en abrégé, *ls*. Chaque style est caractérisé par un indice (nombre entier) qu’on lui attribue au moment de sa création.

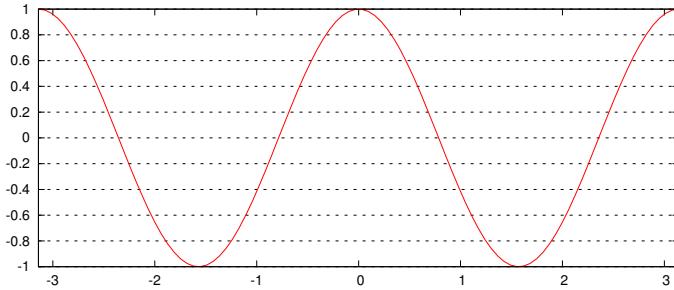


FIGURE 14 – Exemple d’option *grid* avec **plot**.

La commande pour créer un nouveau style est *set style line*. La syntaxe schématique est :

```
set style line <index> options
```

Voici un exemple :

```
set style line 13 lt 3 lw 2 ps 0.5 pt 2 lc 3
```

La valeur 13 est donc l’indice qui caractérise ce style. Pour l’utiliser dans une commande **plot** il faudra spécifier l’option *ls 13* ou *linestyle 13*. Par exemple :

```
plot sin(x) ls 13
```

On peut vérifier les réglages au moyen de l’instruction *show style line*. On obtient ici :

```
gnuplot> show style line 13
    linestyle 13, linetype 3 linecolor 3 linewidth 2.000
    pointtype 2 pointsize 0.500 pointinterval 0
```

La signification des valeurs utilisées pour les différents paramètres dépend du terminal utilisé. La commande **test** permet d’obtenir des informations pour chaque type de terminal (voir la section 7.10).

Pour rétablir des valeurs par défaut dans un style, on dispose du mot-clé *default*. Par exemple :

```
set style line 13 default
```

3.8 Ornements et disposition

3.8.1 Grille

L’option *grid* permet d’obtenir qu’une grille soit tracée sur le graphe. Pour un graphique en 2D, c’est un ensemble de lignes parallèles aux axes et passant par les points servant de graduations sur ces axes (les *tics*). Dans le cas d’un graphique 3D, il s’agit d’une grille similaire qui est tracée sur le plan de base.

Divers arguments optionnels permettent de préciser les caractéristiques de la grille. En particulier, les options *xtics*, *ytics*, *ztics*, ou bien *noxtics*, *noytics*, *noztics* contrôlent quelles lignes doivent être tracées.

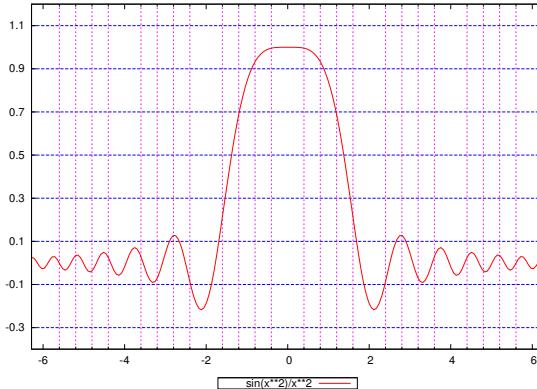


FIGURE 15 – Exemple de grille paramétrée.

Voici un exemple en 2D où seules les lignes horizontales sont dessinées avec une épaisseur de 3 pixels :

```
set grid noxtics ytics lw 3
plot [-pi:pi] cos(2*x)
```

Le graphe est représenté sur la figure 14.

Les lignes tracées par la grille passent par les graduations (*tics*). Il y a des graduations principales et des graduations secondaires et les lignes correspondantes sont différentes. Les graduations secondaires (appelées aussi mineures) sont contrôlées par les mots-clés *mxtics*, *mytics*, *mztics*, *mx2tics*, *my2tics*, avec le préfixe *no* lorsqu'on veut les supprimer (*nomxtics*, *nomytics*, *nomztics*, *nomx2tics*, *nomy2tics*).

On peut aussi spécifier le type de lignes à tracer, la couleur et l'épaisseur avec les paramètres *linestyle* (*ls*), *linetype* (*lt*), *linecolor* (*lc*) et *linewidth* (*lw*) habituels.

Le graphe de la figure 15 a été obtenu de la manière suivante :

```
set key on bmargin center box
set samples 200
set yrang [-0.4:1.2]

set xtics -6,2,6
set ytics -0.3,0.2,1.2
set mxtics 5
set grid back noxtics mxtics nomytics ytics lt 2 lc 3 lw 2, lt 3 lc 4

plot [-2*pi:2*pi] sin(x**2)/x**2
```

L'exemple suivant est un graphique en 3D qui comporte des lignes de grille relatives à l'axe des *z*, d'épaisseur 1 pixel (*lw 1*) et de couleur bleu (*lt 3*) :

```
set grid xtics ytics ztics lw 1 lt 3
splot x**2 + y**2
```

Le graphe est représenté sur la figure 16.

3.8.2 Bordure

L'option *border* concerne le cadre dessiné autour des graphiques : pour un graphique 2D, il s'agit d'un carré et pour un graphique 3D, il s'agit d'un cube. La valeur de l'option est un entier qui permet de spécifier quelles arêtes du carré ou du cube doivent être tracées. Ce nombre entier résulte de l'addition de bits (puissances de 2) représentant les diverses arêtes.

Dans le cas d'un graphique 2D, les valeurs sont les suivantes :

côté inférieur	1
côté gauche	2
côté supérieur	4
côté droit	8

Pour que les quatre côtés soient dessinés, il faut donc que le nombre contienne la valeur $15 = 8 + 4 + 2 + 1$, autrement dit que la décomposition du nombre en base 2 comporte les valeurs précédentes. La valeur par défaut de l'option *border* est 31 qui peut s'écrire $16 + 8 + 4 + 2 + 1$ et contient donc effectivement les valeurs des quatre côtés. Inversement, si on souhaite ne dessiner que les côtés droit et supérieur, on fixera l'option à la valeur $12 = 8 + 4$.

Dans le cas d'un graphique 3D, il y a 12 valeurs possibles puisqu'un cube comporte 12 arêtes. Elles sont indiquées dans le tableau suivant :

inférieur avant gauche	1
inférieur arrière gauche	2
inférieur avant droit	4
inférieur arrière droit	8
vertical gauche	16
vertical arrière	32
vertical droit	64
vertical avant	128
supérieur arrière gauche	256
supérieur arrière droit	512
supérieur avant gauche	1024
supérieur avant droit	2048

Avec la valeur par défaut fixée à 31, un graphique 3D comporte les quatre arêtes de base et l'arête verticale gauche. Inversement, pour que les 12 arêtes du cube soient tracées, il faut fixer l'option *border* à la somme de tous les bits, soit 4095.

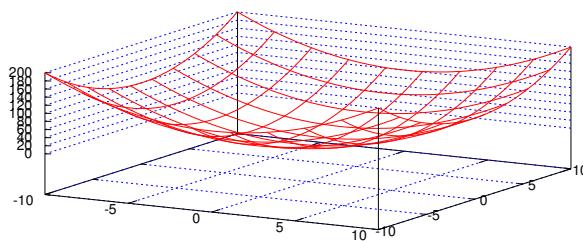


FIGURE 16 – Exemple d'option *grid* avec **splot**.

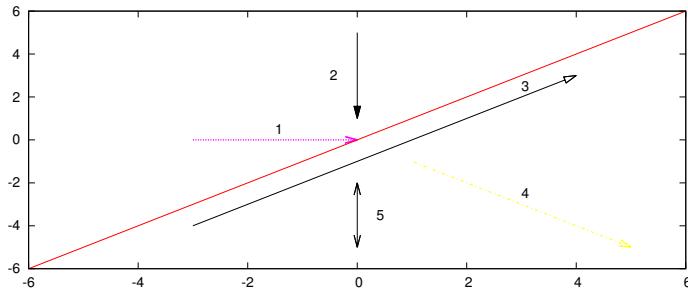


FIGURE 17 – Exemple d’option *arrow*.

Pour supprimer les arêtes, on utilise l’une des deux instructions suivantes :

```
set border 0
unset border
```

L’option *border* supporte d’autres arguments :

- *front* et *back* concernent les graphiques 2D seulement et indiquent si les arêtes doivent être tracées au-dessus des éléments graphiques ou derrière ceux-ci.
- *linewidth (lw)*, *linestyle (ls)* et *linetype (lt)* ont la signification habituelle : ils permettent de spécifier l’épaisseur, le style et le type des droites.

Par exemple :

```
set border 12 lw 3 front
unset xtics
unset ytics
plot sin(x)
```

3.8.3 Flèches

L’option *arrow* permet de placer des flèches sur le graphique. La syntaxe la plus simple est :

```
set arrow <position> to <position>
```

Les positions peuvent être spécifiées dans les systèmes de coordonnées habituels expliqués au paragraphe 2.2 : elles peuvent donc être précédées des mots-clés *first*, *second*, *graph*, *screen*, ou *character*.

On peut utiliser l’option *arrow* autant de fois que nécessaire : les flèches créées peuvent recevoir un identifiant (un *tag*) pour le cas où on souhaite les modifier par la suite. La syntaxe dans ce cas devient :

```
set arrow <tag> from <position> to <position>
```

Le tag est simplement une valeur numérique entière. Pour supprimer une flèche désignée par un tag, on utilise la commande suivante :

```
unset arrow <tag>
```

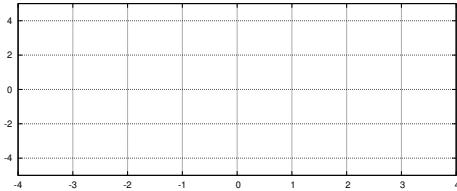


FIGURE 18 – Exemple de graphique vide.

D’autres arguments peuvent être passés dans l’option *arrow* afin de définir les caractéristiques graphiques :

- *arrowstyle* ou *as* pour indiquer un style ;
- *arrowstyle*, *nohead*, *head*, *backhead*, *heads* pour préciser la pointe de la flèche ;
- *size* pour préciser les dimensions de la pointe de la flèche (longueur et angle) ;
- *filled*, *empty*, *nofilled* pour indiquer si la pointe est fermée et remplie ;
- *front* ou *back* pour indiquer si la flèche est dessinée au premier plan ou en arrière-plan ;
- *linestyle* (*ls*), *linetype* (*lt*), *linewidth* (*lw*) pour spécifier le style, le type et l’épaisseur du segment.

Voici un exemple utilisant quelques-unes de ces options :

```
set xrange [-6:6]
set yrange [-6:6]
set arrow 1 from -3,0 lw 3 lt 4
set label "1" at -1.5,0.5
set arrow 2 from 0,5 to 0,1 head filled
set label "2" at -0.5,3
set arrow 3 from -3,-4 to 4,3 head empty
set label "3" at 3,2.5
set arrow 4 from 5,-5 to 1,-1 backhead lw 2 lt 6
set label "4" at 3,-2.5
set arrow 5 from 0,-5 to 0,-2 heads
set label "5" at 0.35,-3.5
plot x
```

Le graphe est représenté sur la figure 17.

3.8.4 Graphique vide

Il peut arriver que l’on veuille générer un graphique ne contenant aucune courbe et aucun point, par exemple pour afficher seulement des axes et des graduations ou une grille. On peut obtenir ce résultat de la manière suivante :

```
set yrange [0:1]
plot NaN
```

Il est indispensable de fixer une valeur pour *yrange*, quelle qu’elle soit, sinon **gnuplot** se plaint de ne pas pouvoir calculer les ordonnées.

Le graphe vide de la figure 18 est obtenu comme ceci :

```
set xrange [-4:4]
set yrange [-5:5]
```

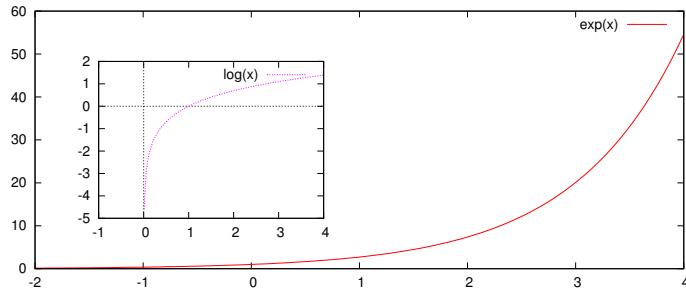


FIGURE 19 – Premier exemple d'option *multiplot*.

```
set grid xtics ytics
plot NaN
```

3.8.5 Graphiques multiples

L'option *multiplot* permet de dessiner plusieurs graphiques sur le même canevas. Il s'agit ici de graphiques complets avec éventuellement des axes, une bordure, des graduations, etc. sur chacun d'entre eux. Cette option ne sert pas à dessiner plusieurs courbes sur un même graphique : pour obtenir ce résultat, il faut spécifier les définitions de ces courbes dans la même commande **plot**.

Les commandes de dessin des différents graphes sont placées entre une instruction *set multiplot* et une instruction *unset multiplot*. Pour chaque graphe, on peut exécuter des options particulières, spécifier une nouvelle origine et de nouvelles dimensions, etc.

Les étiquettes et les flèches sont dessinées dans chaque graphique en fonction des dimensions courantes sauf si elles ont été définies dans le système de coordonnées d'écran (*screen*).

Voici un premier exemple qui place le graphe de la fonction logarithme en incrustation sur le graphe de la fonction exponentielle :

```
set multiplot
set xrange [-2:4]
plot exp(x)
set origin 0.1,0.1
set size 0.4,0.4
set xrange [-1:4]
set xzeroaxis
set yzeroaxis
plot log(x) lt 4
unset multiplot
```

Le graphe est représenté sur la figure 19.

L'option *multiplot* supporte quelques arguments optionnels supplémentaires :

- *layout* indique une subdivision du canevas en lignes et colonnes. Si la valeur de cet argument est, par exemple, 3,2 on pourra dessiner 6 graphiques : 3 rangs avec 2 graphiques par rang.
- *rowsfirst* et *columnsfirst* indiquent, dans le cas où l'argument *layout* a été spécifié, si les graphiques successifs sont placés sur la grille en ligne ou en colonne.

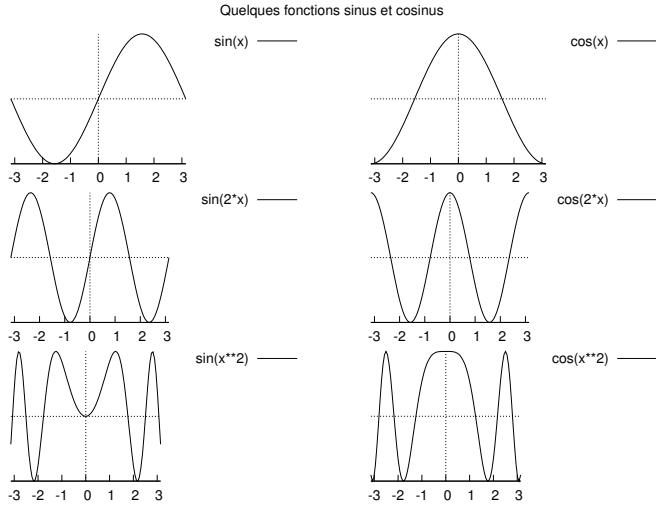


FIGURE 20 – Deuxième exemple d’option *multiplot*.

- *downwards* et *upwards* indiquent, dans le cas où l’argument *layout* a été spécifié, si le placement se fait vers le bas ou vers le haut.
- *title* permet de définir un titre pour l’ensemble du canevas.
- *scale* et *offset* permettent de définir une échelle et un décalage applicables à chaque graphique.

Voici un exemple utilisant ces diverses options :

```

set size 1,1
set origin 0,0
set border 1
unset tics
set xtics border
set xzeroaxis
set yzeroaxis
set multiplot layout 3,2 columnsfirst scale 0.9,1.1 \
    title "Quelques fonctions sinus et cosinus"
set xrange [-pi:pi]
set key on outside
plot sin(x)
plot sin(2*x)
plot sin(x**2)
plot cos(x)
plot cos(2*x)
plot cos(x**2)
unset multiplot

```

Le graphe est représenté sur la figure 20.

3.8.6 Lignes de niveau d’une surface

L’option *contour* permet d’ajouter des lignes de niveau sur un graphique 3D. Les lignes de niveau sont des courbes tracées sur les surfaces représentant l’ensemble des points situés à une cote (ou hauteur) z donnée. Si la surface a pour équation $z = f(x, y)$, la courbe de niveau C est l’ensemble des points (x, y) tels que $f(x, y) = C$. Cet ensemble de points peut être dessiné sur la surface

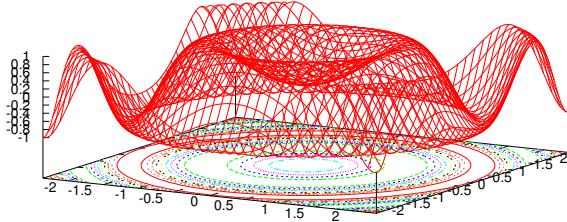


FIGURE 21 – Exemple d’option *contour*.

elle-même ou bien projeté sur le plan de base, ou les deux à la fois. L’option *contour* admet par conséquent un argument optionnel correspondant à ces trois possibilités et qui peut prendre l’une des valeurs suivantes : *surface*, *base* ou *both*. Par défaut, c’est l’argument *base* qui est utilisé.

L’option *cntrparam* permet de fixer certaines caractéristiques des lignes de niveau dessinées avec l’option *contour*. En particulier l’argument *levels* permet de régler le nombre de niveaux souhaité et de préciser leur valeur.

```
set contour
set cntrparam levels auto 10
set isosamples 40
splot [-3*pi/4:3*pi/4] [-3*pi/4:3*pi/4] sin(x**2 + y**2)
```

Le graphe est représenté sur la figure 21.

L’option *surface* permet d’activer ou désactiver le dessin des surfaces avec la commande **splot**. Elle est bien sûr active par défaut mais on peut la désactiver si on veut, par exemple, faire apparaître seulement les lignes de niveau. Par exemple, les instructions suivantes permettent de tracer seulement les lignes de niveau sur le plan de base sans que la surface elle-même soit dessinée :

```
unset surface
set contour base
set cntrparam levels 10
splot sqrt(x**2 + y**2)
```

Le graphe est représenté sur la figure 22.

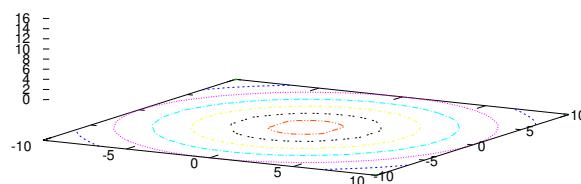


FIGURE 22 – Exemple d’option *surface*.

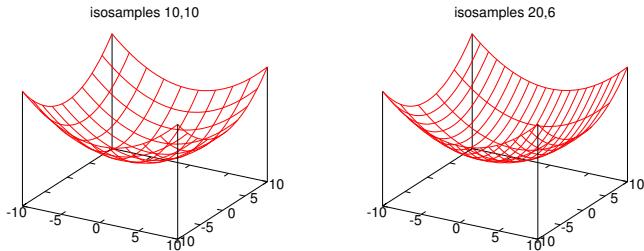


FIGURE 23 – Exemple d’option *isosamples*.

3.8.7 Paramètres de dessin

Option samples

Pour tracer une courbe 2D définie par une formule, **gnuplot** calcule les coordonnées d’un certain nombre de points qui constituent un échantillonnage de la courbe : la courbe elle-même est obtenue en reliant ces points par des segments (lorsque le style de graphique utilisé est *lines*). Le nombre de valeurs de x utilisées pour construire un échantillon est contrôlé par l’option *samples* dont la valeur par défaut est fixée à 100. Si le graphique obtenu ne semble pas suffisamment régulier, on doit augmenter la taille de l’échantillon. Par exemple :

```
set samples 300
```

Dans le cas d’un graphique 3D, on peut fixer la taille de l’échantillon séparément pour les x et les y . Par exemple :

```
set samples 300, 250
```

Si seulement une valeur est spécifiée, elle concerne les x et la même valeur est utilisée pour les y .

Option isosamples

L’option *isosamples* concerne les graphiques 3D et contrôle le nombre de lignes tracées pour le maillage représentant la surface. Ces lignes, appelées iso-courbes, correspondent à des valeurs de x fixées tandis que y varie, ou le contraire (valeurs de y fixées tandis que x varie). Par défaut, on a :

```
gnuplot> show isosamples
iso sampling rate is 10, 10
```

La figure 23 montre côté à côté deux représentations graphiques de la fonction $f(x,y) = x^2 + y^2$. Sur le deuxième graphique, 20 isocourbes sont tracées dans la direction des y et seulement 6 sont tracées dans la direction des x . Le premier graphique utilise les valeurs par défaut : 10 isocourbes dans chaque direction.

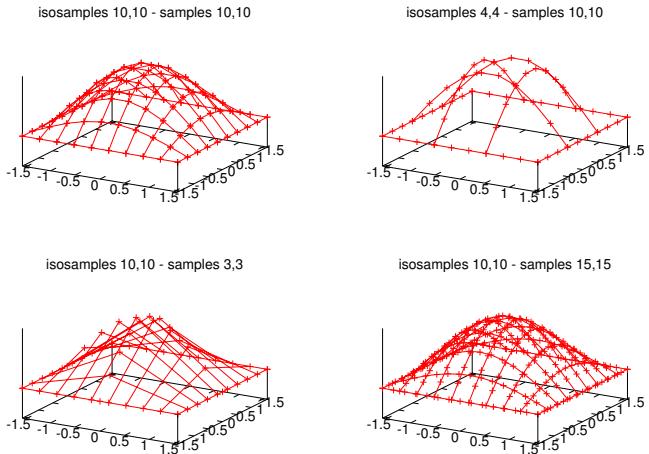


FIGURE 24 – Options *samples* et *isosamples* différentes.

```

set multiplot
set size 0.5,0.8
splot x**2+y**2
set origin 0.5,0
set isosamples 20,6
splot x**2+y**2
unset multiplot

```

Les options *samples* et *isosamples* peuvent prendre des valeurs différentes. L’option *isosamples* indique le nombre d’isocourbes dans les deux directions x et y tandis que l’option *samples* indique combien de points sont représentés sur ces isocourbes. Les points en question ne se trouvent donc pas nécessairement à l’intersection des isocourbes. Les différentes possibilités sont illustrées par l’exemple suivant représenté sur la figure 24 :

```

set multiplot layout 2,2 rowsfirst
set xrange [-pi/2:pi/2]; set yrange [-pi/2:pi/2]
set style function linespoints
set key off
unset ztics
set isosamples 10,10; set samples 10,10;
splot cos(x)*cos(y)
set isosamples 4,4; set samples 10,10;
splot cos(x)*cos(y)
set isosamples 10,10; set samples 3,3;
splot cos(x)*cos(y)
set samples 15,15;
splot cos(x)*cos(y)
unset multiplot

```

Option *hidden3d*

L’option *hidden3d* concerne les graphes de surfaces. Elle permet de ne pas dessiner les parties d’une surface qui sont cachées derrière une autre portion,

autrement dit de ne montrer que les portions de la surface qui sont au premier plan. Le graphe de la figure 25 en donne une illustration en montrant deux surfaces dessinées sans et avec cette option.

Il est obtenu avec le code suivant :

```
f(x,y) = sqrt(1-x**2-y**2)
g(x,y)= (4-x**2-y**2)**2
set multiplot layout 2,2 rowsfirst
set isosamples 20
set xrange [-1:1] ; set yrange [-1:1]
splot f(x,y)
set hidden3d
splot f(x,y)
set isosamples 50
set xrange [-2:2] ; set yrange [-2:2]
set view 80,15,1,1
unset hidden3d
splot g(x,y)
set hidden3d
splot g(x,y)
unset multiplot
```

L'option *hidden3d* possède un argument *trianglepattern* qui permet de préciser la manière dont les cellules sont dessinées. La valeur de cet argument est un nombre entre 1 et 7 calculé en base 2, c'est-à-dire par combinaison des valeurs suivantes : la valeur 1 correspond aux côtés horizontaux, la valeur 2 correspond aux côtés verticaux et la valeur 4 correspond aux diagonales. Le graphe de la figure 26 montre le résultat obtenu avec différentes valeurs.

Le code est le suivant :

```
set view 80,15,1,1
set isosamples 50
set xrange [-2:2] ; set yrange [-2:2]
f(x,y) = sqrt(1-x**2-y**2)
set multiplot layout 2,2 rowsfirst
set hidden3d trianglepattern 1
splot f(x,y)
set hidden3d trianglepattern 2
splot f(x,y)
set hidden3d trianglepattern 5
splot f(x,y)
set hidden3d trianglepattern 7
splot f(x,y)
unset multiplot
```

Option *clip*

L'option *clip* permet de supprimer des points ou des lignes à proximité des bords du graphique. Il existe trois modes de suppression qui sont définis par les valeurs suivantes de l'option *clip* :

- si la valeur est *points*, les points dont les coordonnées sont à l'intérieur du graphique mais qui sont trop proches du bord sont supprimés. Cela peut concerter des points dont la taille est telle qu'ils risqueraient de déborder.
- si la valeur est *one*, les segments dont une des extrémités est en dehors du graphique ne sont dessinés que jusqu'au bord de celui-ci.

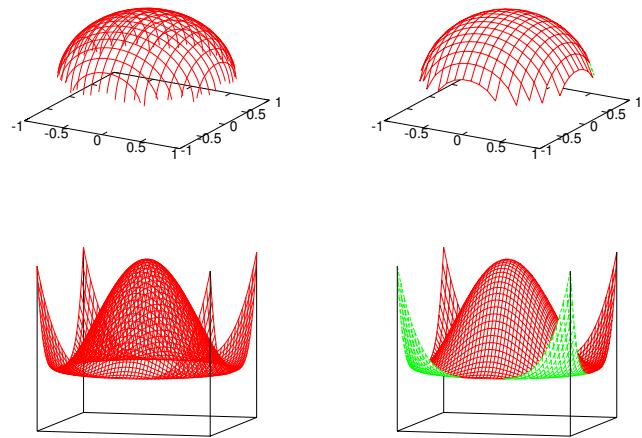


FIGURE 25 – Exemple d'option *hidden3d*.

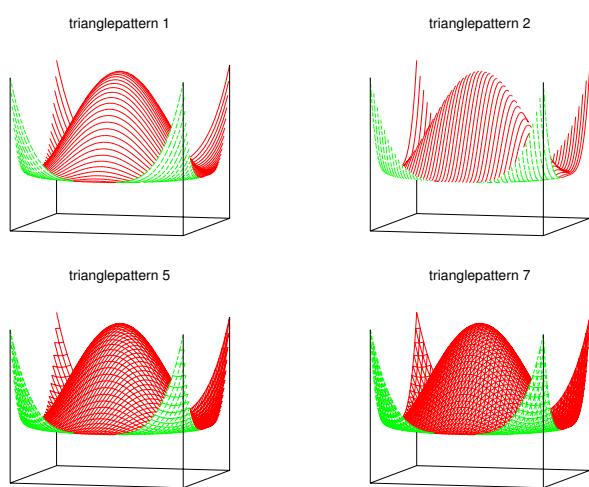


FIGURE 26 – Exemple d'option *trianglepattern*.

- si la valeur est *two*, les segments dont les deux extrémités sont extérieures au graphique sont tronqués : seule la partie qui traverse le graphique est dessinée.

Option **dgrid3d**

L’option *dgrid3d* permet d’influer sur la manière dont **gnuplot** calcule les points servant à représenter une surface. Lorsque la surface est définie par une fonction, **gnuplot** crée une trame à partir d’une séquence d’abscisses x_i et d’ordonnées y_j : tous les points de coordonnées (x_i, y_j) constituent une grille (*mesh grid* en anglais) et le programme calcule les cotes z_{ij} correspondant à ces points.

Lorsque la surface est définie à partir d’un fichier de données, **gnuplot** se contente *a priori* d’utiliser les points fournis par le fichier quelle que soit leur disposition.

L’option *dgrid3d* a pour effet de mêler les deux approches et de faire en sorte que **gnuplot** crée une trame à partir des points fournis et dessine les points correspondant à la trame plutôt que les points du fichier. Ces derniers sont utilisés seulement pour faire des interpolations et générer une trame.

Il existe plusieurs méthodes pour effectuer l’interpolation et le calcul des points de trame. On peut spécifier au moyen de cette option le nombre de coordonnées x et y régulièrement espacées qui constitueront la trame de base et ensuite la méthode à utiliser pour calculer les z . Les méthodes supportées portent les noms suivants : *splines*, *qnorm*, *gauss*, *cauchy*, *exp*, *box*, *hann*.

On se reportera à la documentation de cette option pour en savoir plus sur la définition des méthodes en question.

Exemples de syntaxe :

```
set dgrid3d 20,20 qnorm 1
set dgrid3d 30,30 splines
set dgrid3d 30,30 gauss .75
set dgrid3d 30,30 gauss 0.35,0.5
```

3.9 Objets graphiques

L’option *object* permet de définir des figures géométriques simples telles que des rectangles, des cercles, des ellipses et des polygones. Au moyen de la commande *set object*, on déclare des objets et on les ajoute à un graphique existant.

Chaque objet est identifié par un indice (un tag) et possède des arguments permettant d’en fixer certaines caractéristiques.

Tous les objets admettent des arguments *front*, *back*, *behind* qui permettent de préciser s’ils doivent être dessinés au premier plan ou en arrière-plan. L’argument *behind* est utilisé en général pour créer un arrière-plan pour le graphe tout entier de la manière suivante :

```
set object rectangle from screen 0,0 to screen 1,1 behind
```

3.9.1 Rectangles

Pour les rectangles, la syntaxe est la suivante :

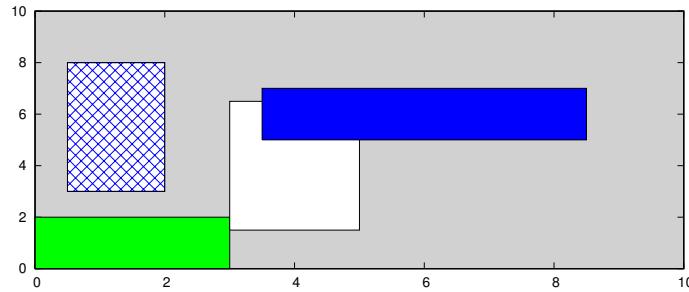


FIGURE 27 – Exemple d’option *object rectangle*

```
set object <index> rectangle
{from <position> {to/rto} <position>
 | center <position> size <w>,<h>
 | at <position> size <w>,<h>}
```

On peut définir un rectangle de trois manières :

- en spécifiant les coordonnées du coin inférieur gauche et du coin supérieur droit ;
- en spécifiant les coordonnées du centre ainsi que la largeur et la hauteur ;
- en spécifiant les coordonnées du coin inférieur gauche ainsi que la largeur et la hauteur.

L’argument *to* indique une position absolue tandis que l’argument *rto* indique une position relative à la position définie dans l’argument *from*.

Les positions peuvent être exprimées dans un des systèmes de coordonnées *axis*, *graph* ou *screen*.

La commande *set style rectangle* permet d’autre part de préciser les caractéristiques graphiques des rectangles selon la syntaxe :

```
set style rectangle {front/back} {lw/linewidth <lw>}
{fillcolor <colorspec>} {fs <fillstyle>}
```

Les indications de style peuvent aussi être déclarées dans la commande *set object rectangle* : dans ce cas, elles portent uniquement sur le rectangle concerné.

Le graphe représenté sur la figure 27 est obtenu avec le code suivant :

```
set key off
set xrange [0:10]
set yrange [0:10]
set style line 15 lt rgb "blue"
set object 1 rect from graph 0,0 to graph 1,1 \
    behind fc rgb "grey10" fs solid 0.2
set object 2 rectangle from 0,0 to 3,2 fillcolor lt 2
set object 3 rectangle center 4,4 size 2,5
set object 4 rectangle at 6,6 size 5,2 fc lt 3
set object 5 rect from 0.5,3 to 2,8 fc ls 15 fs pattern 1 bo -1
plot NaN
```

3.9.2 Cercles

Pour les cercles, la syntaxe est la suivante :

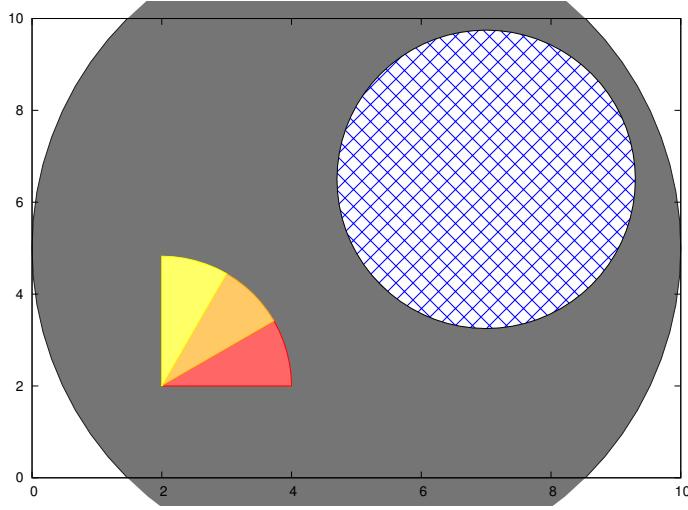


FIGURE 28 – Exemple d’option *object circle*

```
set object <index> circle {at/center} <position> size <radius>
{arc [<begin>:<end>]}
{<other-object-properties>}
```

L’argument *arc* permet de limiter le tracé du cercle à un certain secteur angulaire. Ce secteur est exprimé comme un intervalle de la forme $[a : b]$ où *a* et *b* sont respectivement l’angle de départ et l’angle d’arrivée.

Le graphe représenté sur la figure 28 est obtenu avec le code suivant :

```
set key off
set xrange [0:10]
set yrange [0:10]
set size 1,1
set style fill solid 0.6
set style line 15 lt rgb "blue"
set object 1 circle at graph 0.5,0.5 radius 5 behind fc rgb "grey10"
set object 2 circle arc [0:30] at 2,2 radius 2 fc rgb "red" front
set object 3 circle arc [30:60] at 2,2 radius 2 fc rgb "orange" front
set object 4 circle arc [60:90] at 2,2 radius 2 fc rgb "yellow" front
set object 5 circle at 7,6.5 radius 2.3 fc ls 15 fs pattern 1 border -1 front
plot NaN
```

Les arguments *at* et *center* sont synonymes. De même, on peut désigner le rayon du cercle au moyen de l’argument *size* ou *radius*.

3.9.3 Ellipses

Pour les ellipses, la syntaxe est la suivante :

```
set object <index> ellipse {at/center} <position> size <w>,<h>
{angle <orientation>}
{<other-object-properties>}
```

Le graphe représenté sur la figure 29 est obtenu avec le code suivant :

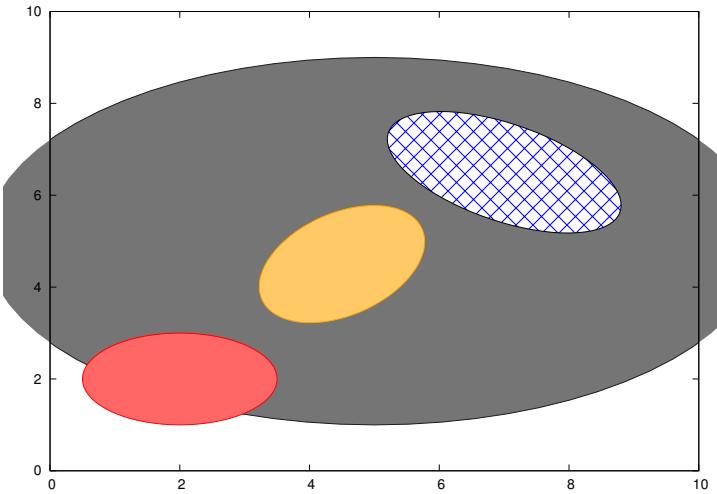


FIGURE 29 – Exemple d’option *object ellipse*

```

set key off
set xrange [0:10]
set yrange [0:10]
set size 1,1
set style fill solid 0.6
set style line 15 lt rgb "blue"
set object 1 ellipse at graph 0.5,0.5 size 12,8 behind fc rgb "grey10"
set object 2 ellipse at 2,2 size 3,2 fc rgb "red" front
set object 3 ellipse at 4.5,4.5 size 3,2 angle 45 fc rgb "orange" front
set object 4 ellipse at 7,6.5 size 4,2 angle -30 fc ls 15 fs pattern 1 \
border -1 front
plot NaN

```

3.9.4 Polygones

Pour les polygones, la syntaxe est la suivante :

```

set object <index> polygon
    from <position> to <position> ... {to <position>}

```

On doit indiquer la séquence des points constituant le polygone. En principe, la dernière position doit être identique à la première, c'est-à-dire que le polygone doit être refermé. Si ce n'est pas le cas, **gnuplot** se charge de le faire et émet un message d'avertissement.

Le graphe représenté sur la figure 30 est obtenu avec le code suivant :

```

set key off
set xrange [0:10]
set yrange [0:10]
set size 1,1
set style fill solid 0.8
set object 1 polygon from 0,2 to 0,4 to 2,5 to 3,3 to 1,1 to 0,2 fc rgb "grey40"
set object 2 polygon from 4,3 to 4,5 to 6,3 to 6,5 to 5,6 to 4,3 fc rgb "yellow"
set style fill empty
set object 3 polygon from 7.5,3 to 7.5,7 to 7,7 to 8,9.5 to 9,7 \

```

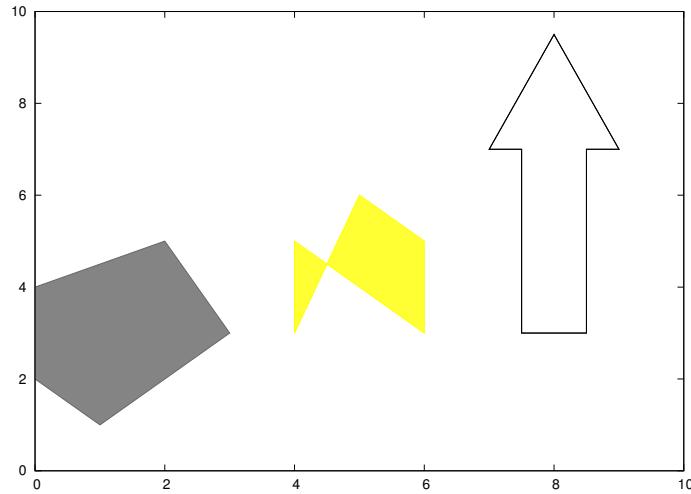


FIGURE 30 – Exemple d’option *object polygone*

```
plot NaN
          to 8.5,7 to 8.5,3 to 7.5,3
```

3.10 Options concernant les nuanciers

Cette section concerne exclusivement l’affichage des nuanciers de couleurs sur un graphique. Les modes de spécification et les réglages de couleurs se font par l’intermédiaire de l’option *palette* et sont détaillés dans la section 8.3. Pour tout ce qui concerne le style de surfaces dit *pm3d* et l’option du même nom, on se reportera à la section 5.10.

On peut faire afficher à côté d’un graphique, le nuancier de gris ou de couleurs utilisé. On utilise pour cela l’option *colorbox*. Plusieurs arguments permettent de contrôler certains aspects. Les arguments *vertical* et *horizontal* spécifient l’orientation. L’argument *default* laisse le programme choisir lui-même l’emplacement : sinon, l’argument *user* permet de spécifier directement l’origine au moyen d’un argument *origin* et la taille au moyen d’un argument *size*. Les valeurs doivent être spécifiées en coordonnées d’écran. Par exemple :

```
set colorbox horiz user origin .1,.02 size .8,.04
```

On peut préciser la bordure au moyen de l’option *border*.

Le graphe de la figure 31 est obtenu avec le code suivant :

```
set xrange [-2:2]
set yrange [-2:2]
set isosamples 100
set colorbox horiz user origin .2,.05 size .65,.02
splot 1/(1+x**2+y**2) with pm3d
```

L’option *cbtics* permet de contrôler les graduations le long du nuancier. La syntaxe autorisée est exactement la même qu’avec l’option *xtics* (voir la section 3.3.4). Par exemple :

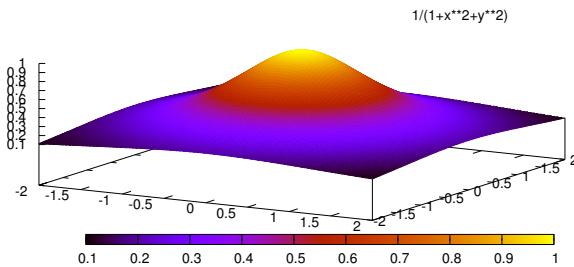


FIGURE 31 – Exemple d’option *colorbox*

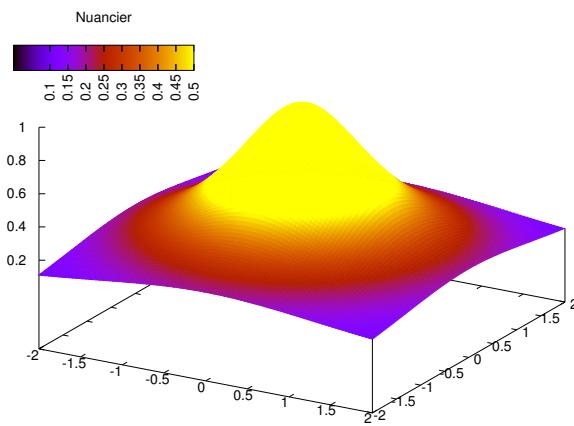


FIGURE 32 – Exemple d’options *cbtics*, *cbrange*, *cblabel*.

```
set cbtics 0.2,0.05,0.8 rotate offset 0,-0.4
```

De même, l’option *mcbtics* permet de contrôler les graduations mineures.

L’option *cbrange* limite l’intervalle des valeurs à colorier. Les valeurs de z extérieures à cet intervalle se voient attribuer la couleur de l’extrémité la plus proche.

L’option *cblabel* permet de placer un texte à proximité du nuancier.

Le graphe de la figure 32 est obtenu avec le code suivant :

```
set xrange [-2:2]
set yrange [-2:2]
set isosamples 100
set colorbox horiz user origin 0.1,0.8 size 0.25,0.05
set ztics 0,0.2,1
set cbtics 0.1,0.05,0.8 rotate offset 0,-0.4
set cbrange [0:0.5]
set cblabel "Nuancier" offset screen 0,0.18
splot 1/(1+x**2+y**2) with pm3d
```

4 Fichiers de données

gnuplot peut créer des graphiques à partir de fonctions ou à partir de données numériques ou textuelles stockées dans des fichiers. Cette section explique les possibilités offertes pour travailler à partir de fichiers de données.

4.1 Types de fichiers

Les fichiers de données peuvent être des fichiers texte (par défaut en encodage ASCII) ou des fichiers binaires.

4.1.1 Fichiers texte

Dans un fichier de données de type texte, les données sont stockées sous forme de colonnes.

Chaque ligne représente un enregistrement (*record* en anglais). Les données d'une ligne sont séparées entre elles par des espacements (séquences de blancs et de tabulations). Les espaces placés à l'intérieur d'une chaîne de caractères entre guillemets ne sont pas considérés comme des séparateurs de champs.

Le séparateur de champs peut être modifié au moyen de l'option *datafile* avec le mot-clé *separator*. Par exemple, pour lire des fichiers csv (*comma-separated value*) ou des fichiers tsv (*tab-separated value*), on utilise respectivement :

```
set datafile separator ","
set datafile separator "\t"
```

Toute ligne commençant par un symbole # est considérée comme un commentaire.

Les lignes vides ont une signification pour **gnuplot**. Elles permettent de délimiter des blocs de données. Par convention, une seule ligne vide est simplement interprétée comme une discontinuité dans les enregistrements : dans le style *lines* aucun segment ne sera dessiné entre les enregistrements séparés par cette ligne.

S'il y a deux lignes vides successives, les données placées avant et après sont traitées comme s'il s'agissait de fichiers différents. On obtient ainsi un découpage en blocs de données qui peuvent être désignés par leur index. Le premier bloc est le bloc 0.

L'argument *index* de la commande *plot* permet de préciser quels blocs doivent être représentés. On peut utiliser la notation *m : n* ou *m : n : p* pour désigner des séries de plusieurs blocs. Par exemple :

```
plot 'fichier.dat' index 1
plot 'fichier.dat' index 3:6
plot 'fichier.dat' index 1:10:2
```

De manière analogue, l'argument *every* permet de sélectionner, selon un schéma périodique, les enregistrements d'un fichier qui doivent être représentés. La syntaxe de cette option est

```
plot 'file' every {<point_incr>}
      {:<block_incr>}
      {:<start_point>}
```

```
{:{<start_block>}
{:{<end_point>}
{:<end_block>}}}}
```

Le schéma est spécifié au moyen de six valeurs séparées par des deux-points. Si une valeur n'est pas spécifiée, elle vaut 1 par défaut pour les incrément, 0 pour les valeurs de départ *start_point* et *start_block*, et la dernière valeur disponible pour *end_point* et *end_block*.

La commande **plot** exécute une boucle d'incrément *point_incr* allant de *start_point* à *end_point* pour les points dans une boucle d'incrément *block_incr* allant de *start_block* à *end_block* pour les blocs de données.

Voici quelques exemples :

```
every ::::3::3      # 4eme bloc
every ::::::9       # 10 premiers blocs
every 2:2          # points de 2 en 2 dans chaque bloc
every ::5:::15      # points 5 à 15 dans chaque bloc
```

Fichiers texte matriciels Le mot-clé *matrix* placé après le nom d'un fichier indique qu'il s'agit d'un fichier texte destiné à la commande **splot** où les données sont écrites sous la forme d'une matrice à M lignes et N colonnes :

$$\begin{array}{cccc} z_{11} & z_{12} & \cdots & z_{1N} \\ z_{21} & z_{22} & \cdots & z_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ z_{M1} & z_{M2} & \cdots & z_{MN} \end{array}$$

Les termes de la matrice sont les coordonnées z . Dans ce cas, les coordonnées x et y sont implicites : on sait seulement qu'il y a M abscisses et N ordonnées, c'est-à-dire que la grille de base comporte MN points régulièrement espacés. L'espacement adopté peut cependant être précisé au moyen des mots-clés *dx* ou *dy* (voir à la section 4.3.1).

4.1.2 Fichiers binaires

Les fichiers binaires sont lus octet par octet. Il y a plusieurs formats possibles de fichiers binaires, c'est-à-dire plusieurs manières, pour **gnuplot**, d'interpréter les données binaires lues séquentiellement.

Pour spécifier qu'un fichier doit être traité comme un fichier binaire et non pas texte, on doit faire suivre son nom du mot-clé *binary* et éventuellement d'options supplémentaires qui précisent le format.

Le format binaire par défaut est appelé *binaire matriciel* et convient pour des données destinées à la commande **splot** (et non pas **plot**). Les données sont des nombres en virgule flottante sur 4 octets.⁵ Le fichier est séquentiel mais on l'interprète en le lisant en ligne et on considère qu'il représente un tableau

5. C'est-à-dire des floats en langage C et non des int.

matriciel de la forme suivante :

N	y_1	y_2	\cdots	y_N
x_1	z_{11}	z_{12}	\cdots	z_{1N}
x_2	z_{21}	z_{22}	\cdots	z_{2N}
\vdots	\vdots	\vdots	\ddots	\vdots
x_M	z_{M1}	z_{M2}	\cdots	z_{MN}

Autrement dit, le premier nombre (4 premiers octets) représente le nombre N d'ordonnées y_j . La taille totale du fichier sera $(N + 1)(M + 1)$ où M est le nombre d'abscisses. Les couples (x_i, y_j) constituent le maillage (*mesh grid*) du graphique en 3D et les quantités z_{ij} sont les cotes correspondantes.

L'autre format binaire est dit binaire général et est caractérisé par l'utilisation, après l'option *binary*, de l'un des mots-clés suivants : *array*, *record*, *format*, *fletype*.

Même si ces données sont binaires, il faut conceptuellement se les représenter sous la forme de colonnes de valeurs numériques et l'option *using* peut être utilisée comme dans le cas de fichiers texte. Par exemple, une option *using 1 :3* suffira à **gnuplot** pour découper les données séquentielles du fichier binaire en séquences de trois nombres en virgule flottante (*floats*) représentant une ligne de données.

Mot-clé *format* Le mot-clé *format* permet de donner des indications plus précises sur la manière de découper les données. Par défaut, les nombres sont supposés être en virgule flottante sur 4 octets, mais on peut indiquer qu'il en est autrement. Par exemple une option

```
format='%float%int%short'
```

indique qu'il y a trois valeurs par ligne (donc 3 colonnes) et que le flot d'octets doit être découpé en une succession de valeurs de type *float*, puis *int*, puis *short*. Les types reconnus sont rassemblés dans la tableau 4 : certains sont dépendants de la machine pour laquelle **gnuplot** a été compilé, d'autres sont indépendants. On peut faire afficher ces valeurs au moyen de la commande :

```
show datafile binary datasizes
```

Le nom du type à utiliser doit être précédé d'un symbole %, comme dans l'exemple précédent. Une convention supplémentaire consiste à ajouter un astérisque après le symbole % afin que **gnuplot** ignore les valeurs correspondantes. Par exemple, le format

```
format='%float%int%*short'
```

indique que la troisième valeur doit être ignorée.

Mot-clé *array* Une option de la forme *array* = $m \times n$ indique que les données doivent être interprétées comme un tableau de taille $m \times n$, autrement dit comportant m points dans la direction des x et n points dans la direction des y . Dans ce cas-là, en-dehors de précisions supplémentaires, les abscisses et les ordonnées sont déterminées implicitement.

Dépendant de la machine	
<i>Nom</i>	<i>Taille</i>
char, schar, c	1
uchar	1
short	2
ushort	2
int, sint, i, d	4
uint, u	4
long, ld	8
ulong, lu	8
float, f	4
double, lf	8
Indépendant de la machine	
int8, byte	1
uint8, ubyte	1
int16, word	2
uint16, uword	2
int32	4
uint32	4
int64	8
uint64	8
float32	4
float64	8

TABLE 4 – Types de données binaires pour l’option *format*.

Cette forme de spécification du format binaire convient, en particulier, aux graphiques de type *image* ou *rgbimage*. Par exemple :

```
plot 'blutux.rgb' binary array=128x128 format='%uint8' with rgbimage
```

Le fichier *blutux.rgb* comporte 49152 octets : le type *rgbimage* attend des séquences de trois valeurs numériques. Par conséquent les données sont lues comme des triplets d'octets représentant un pixel avec ses couleurs R, G et B. Ils sont organisés sous la forme d'une grille de 128 lignes et 128 colonnes.

Avec un graphique de type *image*, on doit fournir une seule valeur par pixel et la couleur est dérivée de la palette courante. Par exemple :

```
plot 'blutux.rgb' binary array=128x128 format='%uchar%uchar%uchar'
      using ($1+$2+$3)/3 with image
```

Ici, les trois octets sont lus et c'est leur moyenne qui est utilisée pour la correspondance avec la palette de couleurs.

On peut afficher la palette de couleurs au moyen de la commande :

```
set colorbox
```

L'étendue des indices de la palette est contrôlée par l'option *cbrange*.

Les tableaux conviennent aussi aux données destinées à la commande **plot**. Un tableau à une dimension représentera les ordonnées de points régulièrement échantillonnés. Par exemple :

```
plot 'sine.bin' binary array=201 format='%f' using 1 with lines
```

Ici le fichier *sine.bin* comporte 804 octets : les données sont lues comme des *floats* sur 4 octets, ce qui donne 201 valeurs numériques. Les abscisses sont implicites : par défaut, on obtient des abscisses allant de 0 à 200.

Mot-clé record Ce mot-clé fonctionne comme le mot-clé *array* mais la différence est que les coordonnées ne sont pas générées implicitement par **gnuplot**. D'autre part, on spécifie les dimensions des données sous la forme $m_1 : m_2 : \dots$ qui signifie que les enregistrements sont regroupés par paquets de m_1 , m_2, \dots lignes.

Par exemple :

```
splot 'scatter.bin' binary record=30:30:29:26 using 1:2:3
```

Ici le fichier *scatter.bin* comporte 1380 octets : les données sont lues comme trois blocs à trois colonnes, chacun contenant respectivement 30, 30, 29 et 26 enregistrements. Comme les valeurs sont des *floats* sur 4 octets, on obtient bien au total

$$(30 + 30 + 29 + 26) \times 3 \times 4 = 1380.$$

Mot-clé skip Conjointement avec le mot-clé *record*, le mot-clé *skip* permet d'ignorer un nombre spécifié d'octets. S'il y a plusieurs blocs de données et que dans chacun un certain nombre d'octets doivent être ignorés, on utilise un symbole deux-points pour séparer les valeurs. Par exemple :

```
plot 'data.bin' binary skip=1024
plot 'data.bin' binary record=400:200 skip=256:512
```

Mot-clé *endian* Le mot-clé *endian* permet de contrôler la manière dont sont lues les données binaires codées sur plusieurs octets. Les valeurs possibles pour cette option sont *little*, *big*, *default* et *swap*.

Formats binaires prédéfinis

La commande **gnuplot** est capable de lire un certain nombre de formats binaires classiques. La liste des formats supportés peut être obtenue au moyen de la commande suivante :

```
gnuplot> show datafile binary filetype
This version of gnuplot understands the
following binary file types:

avp bin edf ehf gif gpbin jpeg jpg png raw rgb auto
```

4.2 L'option *datafile*

L'option *datafile* permet de contrôler la manière dont sont lues les valeurs depuis un fichier de données.

```
set datafile missing
set datafile separator
set datafile commentschars
set datafile binary
```

La commande *set datafile missing* permet de déclarer le symbole utilisé pour désigner des données manquantes dans un fichier de données. Par exemple :

```
set datafile missing "NaN"
```

La commande *set datafile separator* a été vue à la section 4.1.1 : elle permet de déclarer le séparateur de champs dans un fichier de données au format texte.

La commande *set datafile commentschars* permet de déclarer le symbole utilisé pour les commentaires dans un fichier de données. Par défaut, il s'agit du caractère #.

La commande *set datafile binary* permet de déclarer des réglages par défaut pour la lecture de fichiers binaires. On utilise les mêmes arguments que dans une commande **plot** ou **splot** après le mot-clé *binary*. Par exemple :

```
set datafile binary filetype=png
set datafile binary array=256x256 format="%uint8"
```

4.3 Représentation à partir d'un fichier de données

4.3.1 Transformation des données

Il y a quelques options supplémentaires qui permettent d'influer sur la manière dont **gnuplot** traite les données binaires.

Dans le cas d'un format matriciel (texte ou binaire), les coordonnées *x* et *y* qui permettent de construire la grille de base pour la commande **splot** sont calculées implicitement par **gnuplot**. Si la matrice est de taille $M \times N$, les

M valeurs de x sont espacées régulièrement de 1 à M et de même pour les y . On peut spécifier un autre espacement au moyen des options dx , dy et dz . Par exemple :

```
plot 'sine.bin' binary array=201 dx=0.01 format='%f' with lines
plot 'blutux.rgb' binary array=(128,128) dx=0.7 dy=0.5 \
      format='%uint8' with rgbimage
```

L'effet de ces options est simplement de modifier l'étalonnement des valeurs sur les axes correspondants.

Les options $flipx$, $flipy$ et $flipz$ permettent de renverser une image (ou de manière générale la direction dans laquelle les données binaires sont scannées) le long de la dimension correspondante. Par exemple :

```
plot 'blutux.rgb' binary array=(128,128) flipy format='%uint8' with rgbimage
```

Les pixels des images sont souvent repérés à partir du coin supérieur gauche alors que les coordonnées cartésiennes se repèrent à partir du coin inférieur gauche : ici l'option $flipy$ permet d'accorder les deux.

L'option $origin$ permet de positionner le coin inférieur gauche d'un tableau. Par défaut, **gnuplot** le place au point de coordonnées (0,0). Par exemple :

```
plot 'blutux.rgb' binary array=(128,128) flipy origin=(10,20) \
      format='%uint8' with rgbimage
```

De manière analogue, l'option $center$ positionne le centre d'un tableau. Par exemple :

```
plot 'blutux.rgb' binary array=(128,128) flipy center=(10,20) \
      format='%uint8' with rgbimage
```

L'option $scan$ affecte la manière dont **gnuplot** met en correspondance les données qu'il lit avec les coordonnées. Par exemple, dans le cas d'un tableau destiné à la commande **plot**, **gnuplot** lit en premier les x et, pour une valeur fixée de x , il lit les y . Autrement, il lit les matrices ligne par ligne et considère qu'une ligne représente une valeur de x et que les colonnes représentent les valeurs de y . On peut modifier cet ordre avec l'option $scan$ en écrivant :

```
scan=yx
```

Cela a pour effet de transposer le tableau de données binaires et, par conséquent, d'opérer une symétrie sur l'image.

Dans le cas de la commande **splot**, l'ordre des trois dimensions est xyz par défaut mais peut également être modifié par l'option $scan$. Par exemple :

```
scan=xzy
```

Cette option s'applique aussi au cas des coordonnées cylindriques en utilisant les lettres t , r , z au lieu de x , y , z .

L'option $transpose$ permet d'obtenir une transposition directement. Autrement dit, elle est synonyme de $scan=yxz$ avec **plot** et de $scan=yxz$ avec **splot**.

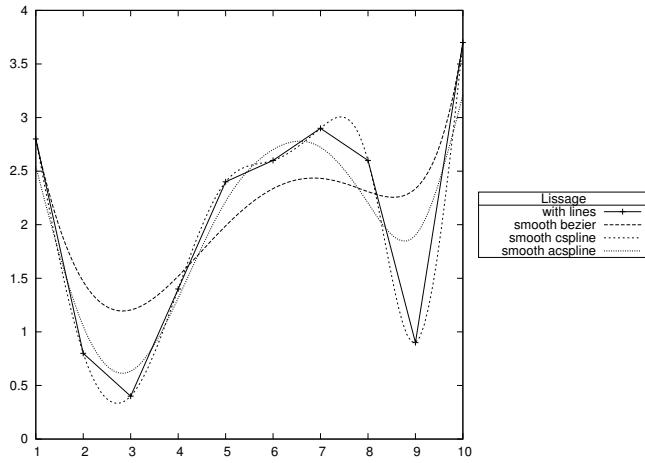


FIGURE 33 – Premier exemple d’option *smooth*.

Le mot-clé *rotate* exécute une rotation d’angle donné autour de l’origine si celle-ci est spécifiée avec l’option *origin* ou autour du centre du tableau. Cette rotation s’applique aux coordonnées *x* et *y*, qu’il s’agisse de la commande **plot** ou de la commande **splot**. Les angles peuvent être exprimés en radians, en multiples de π ou en degrés. Par exemple :

```
rotate=0.785
rotate=0.25pi
rotate=45deg
rotate=45d
```

sont quatre manières différentes d’exprimer l’angle $\frac{\pi}{4}$.

Le mot-clé *perpendicular* permet de spécifier un vecteur normal au plan dans lequel l’image doit être placée. Cette option concerne exclusivement les graphiques exécutés avec la commande **splot**. La valeur est un triplet représentant les coordonnées du vecteur normal. Par exemple :

```
splot 'blutux.rgb' binary array=(128,128) flipy perp=(1,-1,1) \
format='%uint8' with rgbimage
```

On peut le combiner avec l’option *rotate* :

```
splot 'blutux.rgb' binary array=(128,128) flipy perp=(1,-1,1) \
rotate=90d format='%uint8' with rgbimage
```

4.3.2 Lissage des données

L’option *smooth* permet d’appliquer aux données des opérateurs de lissage afin d’éliminer le bruit et les imprécisions. Les techniques de lissage disponibles sont choisies au moyen d’un des mots-clés suivants :

- *unique* : si plusieurs points ont même abscisse, ils sont remplacés par un point unique dont l’ordonnée est la moyenne des ordonnées.
- *frequency* : si plusieurs points ont même abscisse, ils sont remplacés par un point unique dont l’ordonnée est la somme des ordonnées.

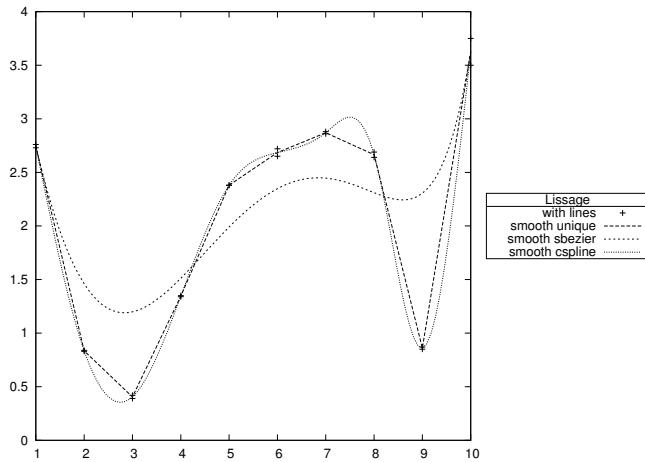


FIGURE 34 – Deuxième exemple d’option *smooth*.

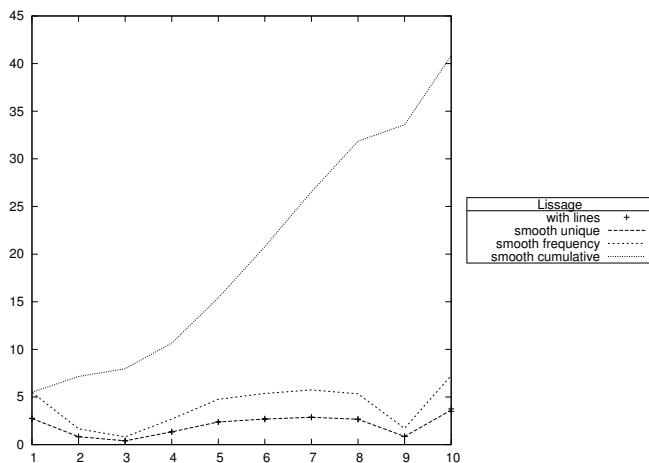


FIGURE 35 – Troisième exemple d’option *smooth*.

- *cumulative* : si plusieurs points ont même abscisse, ils sont remplacés par un point unique dont l'ordonnée est la somme cumulée des ordonnées de tous les points précédents. Cela permet de construire des fonctions de répartition.
- *csplines* : les points sont d'abord remplacés comme avec l'option *unique* puis ils sont lissés au moyen d'une fonction spline cubique.
- *acsplines* : cette méthode requiert une troisième colonne de données indiquant des poids à appliquer pour le calcul des fonctions splines. Ces poids ajoutent des points intermédiaires et rendent l'approximation d'autant meilleure qu'ils sont élevés.
- *bezier* : s'il y a n points, cette méthode utilise une courbe de Bezier de degré n reliant les points extrémaux. En cas de points multiples ayant la même abscisse, il faut plutôt utiliser la méthode *sbezier*.
- *sbezier* : les points sont d'abord remplacés comme avec l'option *unique* puis les points uniques sont lissés par une courbe de Bezier reliant les points extrémaux.
- *kdensity* : cette option calcule une estimation de la densité d'une série d'observations. C'est une implémentation de la méthode dite KDE (*kernel density estimator*) utilisant un noyau gaussien. Il faut fournir deux ou trois colonnes de données : la première est le vecteur d'observations, la seconde est un vecteur de poids affectés à chaque valeur et la troisième permet éventuellement de spécifier les bandes passantes à utiliser pour chaque valeur. Si la troisième colonne n'est pas spécifiée, **gnuplot** calcule lui-même une bande passante appropriée.

Voici quelques illustrations de ces méthodes de lissage. Le graphe de la figure 33 est obtenu avec le code suivant :

```
plot 'smooth01.dat' with linespoints title "with lines"
replot '' smooth bezier title "smooth bezier"
replot '' smooth cspline title "smooth cspline"
replot '' using 1:2:(5.0) smooth acspline title "smooth acspline"
```

Ici le fichier *smooth01.dat* contient les données suivantes :

1	2.8
2	0.8
3	0.4
4	1.4
5	2.4
6	2.6
7	2.9
8	2.6
9	0.9
10	3.7

Pour les deux exemples qui suivent, on utilise un fichier de données *smooth02.dat* contenant les données suivantes (où chaque abscisse existe avec deux ordonnées différentes) :

1	2.73
1	2.76

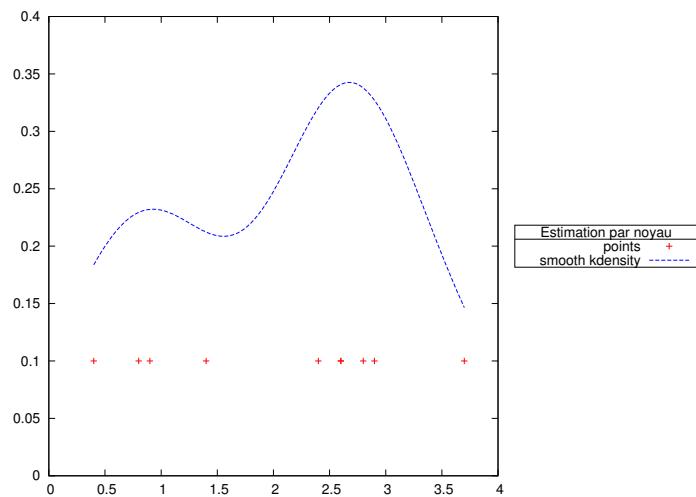


FIGURE 36 – Estimation de densité avec *smooth kdensity*

```

2    0.83
2    0.84
3    0.39
3    0.42
4    1.34
4    1.35
5    2.38
5    2.38
6    2.65
6    2.72
7    2.86
7    2.88
8    2.64
8    2.69
9    0.85
9    0.87
10   3.5
10   3.75

```

Le graphe de la figure 34 est obtenu avec le code suivant :

```

plot 'smooth02.dat' with points title "with lines"
replot '' smooth unique title "smooth unique"
replot '' smooth sbezier title "smooth sbezier"
replot '' smooth cspline title "smooth cspline"

```

Le graphe de la figure 35 est obtenu avec le code suivant :

```

plot 'smooth02.dat' with points title "with lines"
replot '' smooth unique title "smooth unique"
replot '' smooth frequency title "smooth frequency"
replot '' smooth cumulative title "smooth cumulative"

```

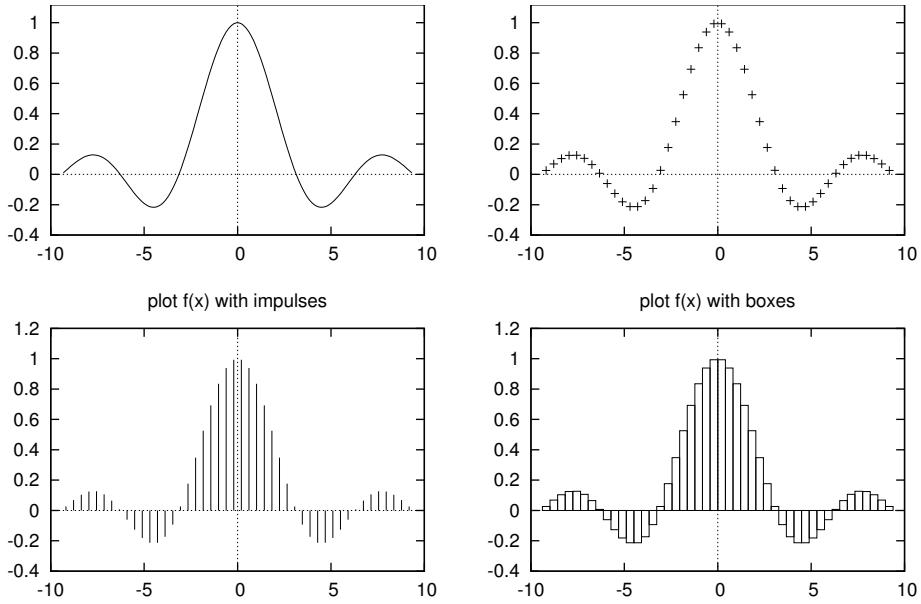


FIGURE 37 – Exemples de styles *points*, *lines*, *impulses*, *boxes*.

L’exemple suivant illustre le calcul d’une estimation de densité par noyau au moyen de l’option *smooth kdensity*. Il utilise le fichier *smooth01.dat* qui comporte 10 valeurs. On fixe un poids identique de 1/10 en chaque point et une bande passante de 0,6. Le graphe est représenté sur la figure 36. Il est obtenu avec les instructions suivantes :

```
set key center vertical rmargin box
set key title "Estimation par noyau"
set yrange [0:0.4]
plot 'smooth01.dat' using 2:(0.1) with points title "points", \
     '' using 2:(0.1):(0.6) smooth kdensity lc rgb "blue" \
     title "smooth kdensity"
```

5 Les types de graphiques

L’option *with* de la commande **plot** permet de spécifier un style de graphique particulier. Le style le plus couramment utilisé est le style *lines* : qu’il s’agisse de tracer un graphique à partir d’une fonction ou à partir d’un fichier stockant des données numériques, **gnuplot** place les points correspondants puis les joint par des segments. Si les points sont suffisamment nombreux et rapprochés les uns des autres on a une illusion de continuité.

D’autres styles que le style *lines* sont disponibles. Par exemple, on peut spécifier le style *points* pour n’avoir que les points, *linespoints* pour avoir à la fois les points et les segments qui les joignent, *impulses* pour avoir des lignes verticales, *boxes* pour avoir des boîtes rectangulaires, etc.

Les styles couramment disponibles sont : *lines*, *points*, *linespoints*, *impulses*, *dots*, *steps*, *fsteps*, *histeps*, *labels*, *boxes*, *histograms*, *filledcurves*, *financebars*, *can-*

dlesticks, vectors, image, rgbimage, pm3d.

Voici quelques exemples avec la fonction $f(x) = \frac{\sin(x)}{x}$ sur l'intervalle $[-3\pi, 3\pi]$. Les graphes de la figure 37 ont été obtenus avec les instructions suivantes successivement :

```
f(x) = x>=-3*pi && x < 3*pi ? sin(x)/x : NaN
set zeroaxis
set yrange [-0.4:1.2]
plot f(x) with lines
plot f(x) with points
plot f(x) with impulses
plot f(x) with boxes
```

Si on ne spécifie pas l'option *with*, le style par défaut utilisé pour les fonctions est *lines* et celui pour les données lues dans un fichier est *points* comme on peut le vérifier avec les commandes suivantes :

```
gnuplot> show style function
      Functions are plotted with lines
gnuplot> show style data
      Data are plotted with points
```

Il est possible de modifier ces valeurs par défaut au moyen d'une commande *set style* comme par exemple :

```
show style function linespoints
show style data impulses
```

5.1 Le style *labels*

Le style *labels* requiert trois colonnes de données : la troisième doit comporter du texte qui sera affiché à l'emplacement déterminé par les deux premières. Supposons qu'on dispose d'un fichier de données *coords.dat* donnant la latitude et la longitude de quelques villes françaises. Voici les premières lignes de ce fichier :

```
8.717 41.917 "Ajaccio"
-0.533 47.483 "Angers"
-1.467 43.500 "Bayonne"
-0.567 44.833 "Bordeaux"
-4.500 48.383 "Brest"
-1.617 49.633 "Cherbourg"
```

On obtient le graphe de la figure 38 au moyen de l'instruction suivante :

```
plot "coords.dat" with labels using 1:(\$2+0.2), "" with points lc 1
```

Cette instruction lit les données dans le fichier *coords.dat* et les place avec l'option *with labels*, ce qui a pour effet d'écrire le nom des villes, puis ensuite lit à nouveau les données du fichier pour placer des points. L'option *using 1 :(\$2-0.2)* signifie que les ordonnées de ces points (deuxième colonne dans le fichier représentée par le symbole *\\$2*) sont décalées de 0.2 vers la bas.

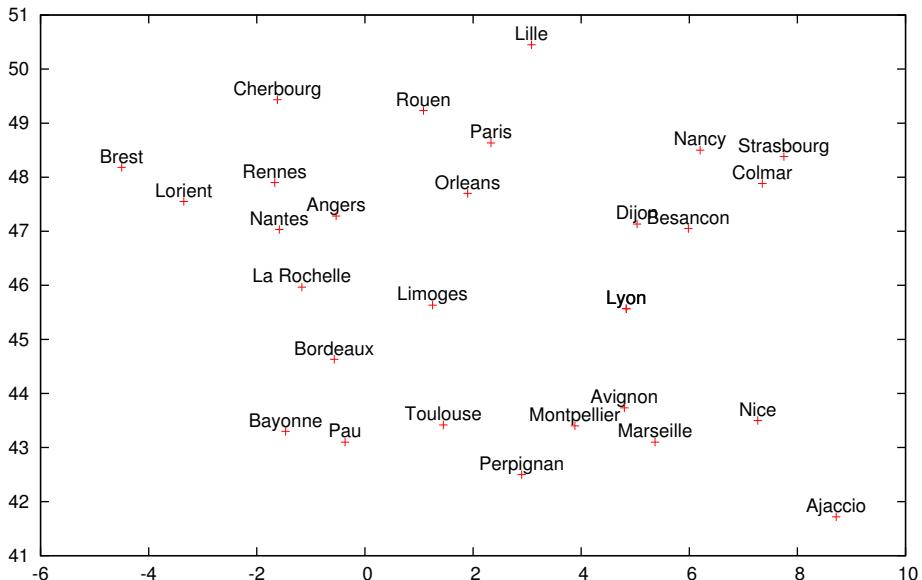


FIGURE 38 – Exemples de style *labels*.

Dans l'instruction qui précède, la chaîne vide "" est une convention de **gnu-plot** qui indique qu'il faut utiliser le même fichier que précédemment.

Remarque : afin d'avoir une représentation qui ne soit pas déformée horizontalement ou verticalement, il faut s'assurer que la même unité de longueur est utilisée sur les deux axes. Comme expliqué à la section 3.6, on obtient ce résultat avec l'instruction suivante qui doit être placée avant la commande **plot** :

```
set size ratio -1
```

5.2 Le style *filledcurves*

Le style *filledcurves* concerne le remplissage de zones délimitées par une courbe ou située entre deux courbes. Dans le cas d'une seule courbe, on peut la spécifier par une fonction ou au moyen de deux colonnes dans un fichier de données correspondant aux abscisses et aux ordonnées. Dans le cas de courbes, il faut spécifier trois colonnes de données : une pour les abscisses et deux pour les ordonnées de chacune des deux courbes.

Il existe quelques options qui indiquent si la zone à remplir est fermée ou délimitée par des droites verticales ou horizontales ou par un point qui sera le sommet d'un secteur angulaire. Ces options peuvent être spécifiées dans la commande **plot** après l'option *with filledcurves* ou préalablement dans une commande *set style data* ou *set style function*. Schématiquement on a les syntaxes suivantes :

```
set style data filledcurves <options>
```

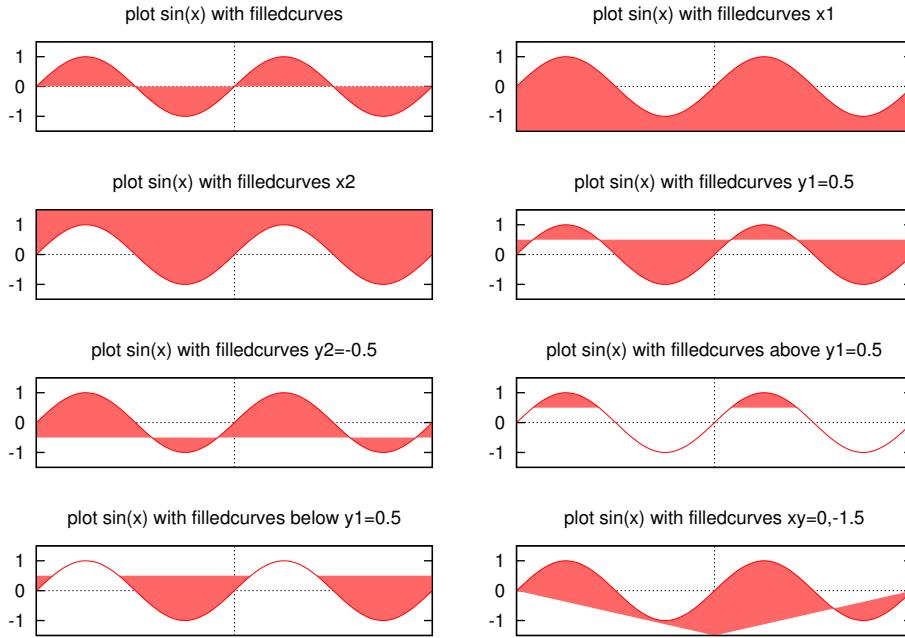


FIGURE 39 – Exemples de style *filledcurves*.

```
set style function filledcurves <options>
plot f(x) with filledcurves <options>
```

Les options possibles sont :

- *closed* pour utiliser la courbe elle-même comme un polygone fermé
- *{above | below} {x1 | x2 | y1 | y2} [=<a>]* pour limiter le remplissage à la zone comprise entre la courbe et une droite horizontale. On désigne cette droite horizontale par les symboles *x1* ou *x2* s'il s'agit de l'un de ces axes ou par une expression *y1=a* ou *y2=a* pour indiquer une droite d'équation $y = a$ par rapport à l'axe *x1* ou *x2*. Les mots-clés *above* et *below* limitent le remplissage à la portion au-dessus ou au-dessous de la droite horizontale (qui peut très bien traverser la courbe elle-même).
- *xy=<x>,<y>* pour définir les coordonnées d'un point qui sera le sommet d'un secteur angulaire limité par la courbe.

Le type de remplissage est spécifié au moyen d'une commande *set style fill*. Cela peut être une couleur unie ou un motif. La syntaxe générale de cette option est :

```
set style fill {empty | solid {<density>} | pattern {<n>}}
               {border {<linetype>} | noborder}
```

Le mot-clé *solid* indique qu'il faut utiliser une couleur unie éventuellement corrigée par un coefficient de densité (valeur entre 0 et 1). La couleur à utiliser est indiquée dans la commande **plot** par l'option *linecolor* habituelle. Le mot-clé *pattern* indique qu'il faut utiliser un motif : la valeur *n* est l'indice du premier

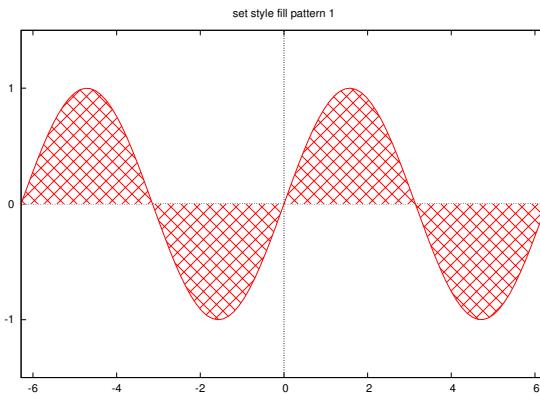


FIGURE 40 – Exemple de style *filledcurves* avec motifs.

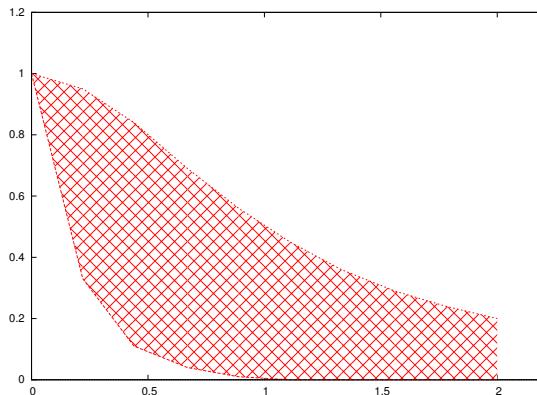


FIGURE 41 – Exemple de style *filledcurves* entre deux courbes.

motif à utiliser. Les motifs disponibles dépendent du type de terminal utilisé. Le mot-clé *border* indique si on veut un contour solide ou pas.

Les exemples de la figure 39 illustrent les diverses combinaisons d'options possibles. Les commandes utilisées sont indiquées au-dessus de chaque graphe. Le mode de remplissage est réglé par l'instruction suivante :

```
set style fill solid 0.6
```

La figure 40 a été obtenue avec le code suivant :

```
set style fill pattern 1
plot sin(x) with filledcurves
```

Attention : tous les terminaux ne sont pas capables d'afficher les motifs. Le code précédent affiche une couleur unie sur certains écrans mais des motifs corrects si le terminal est *postscript*.

La figure 41 a été obtenue avec le code suivant :

```
set style fill pattern 1
plot "courbes.dat" with filledcurves, \
```

```
''' u 1:2 with lines lc 1, \
''' u 1:3 with lines lc 1
```

avec un fichier *courbes.dat* comportant trois colonnes comme ceci :

0	1	1
0.22	0.33	0.95
0.44	0.11	0.84
0.67	0.04	0.69
0.89	0.01	0.56
1.11	0	0.45
1.33	0	0.36
1.56	0	0.29
1.78	0	0.24
2	0	0.2

5.3 Le style *circles*

Ce style dessine un cercle ou un disque de diamètre variable en chaque point. Le fichier de données doit contenir trois colonnes correspondant respectivement aux abscisses, aux ordonnées et aux rayons des cercles. Optionnellement, une quatrième colonne peut contenir des informations de couleur.

La figure 42 reprend le fichier de données de la section 5.2 et illustre quelques possibilités. Elle est obtenue à partir du code suivant :

```
set xrange [-0.5:2.5]
set yrange [-0.2:1.2]
set key off
plot "courbes.dat" u 1:3:(0.4*rand(0)) with circles, \
      "" u 1:3 with linespoints lc rgb "blue" ps 2
```

5.4 Points et lignes

Le style *points* correspond à un graphique dans lequel chaque point est représenté par un petit symbole. La taille des symboles peut être contrôlée par l'option *pointsize*.

Il existe aussi le style *linespoints* qui trace à la fois des points et des segments joignant ces points entre eux. On peut contrôler les points qui seront marqués par un symbole au moyen de l'option *pointinterval* : la valeur de cette option

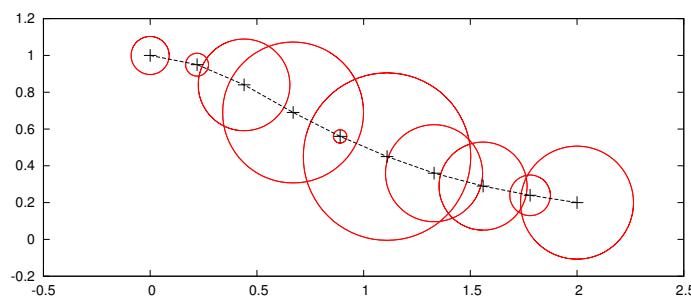


FIGURE 42 – Exemple de style *circles*

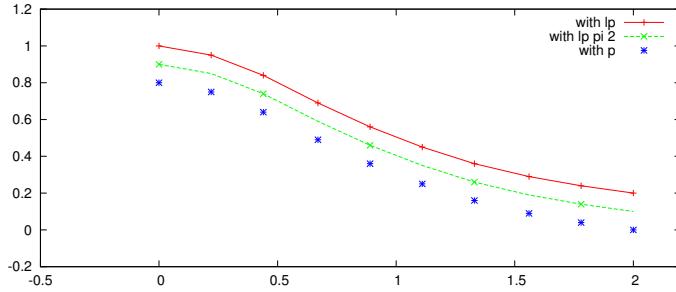


FIGURE 43 – Exemple de style *points* et *linespoints*.

est un nombre entier indiquant la périodicité des points. Par exemple, avec les commandes suivantes (la deuxième est l’abrégué de la première)

```
with linespoints pointinterval 2
w lp pi 2
```

tous les segments sont dessinés mais un point sur deux est marqué.

La figure 43 reprend le fichier de données de la section 5.2 et illustre quelques possibilités. Elle est obtenue à partir du code suivant :

```
set xrange [-0.5:2.2]
set yrange [-0.2:1.2]
plot "courbes.dat" u 1:3 with linespoints title "with lp", \
     "" u 1:(\$3-0.1) with linespoints pi 2 title "with lp pi 2", \
     "" u 1:(\$3-0.2) with points title "with p"
```

5.5 Le style *vectors*

Ce style de représentation permet de dessiner des vecteurs en 2 ou 3 dimensions. Dans le cas d’un graphique en 2D, les vecteurs sont tracés entre les points (x, y) et $(x + \Delta x, y + \Delta y)$: les fichiers de données doivent donc contenir au moins 4 colonnes représentant les valeurs de $x, y, \Delta x$ et Δy .

Dans le cas d’un graphique en 3D, il faut au moins 6 colonnes représentant les valeurs de $x, y, z, \Delta x, \Delta y$ et Δz respectivement.

L’argument *with vectors*, dans les commandes **plot** et **splot**, peut être suivi de spécifications concernant la forme des vecteurs. la syntaxe supportée est la même qu’avec la commande **set arrow**.

La figure 44 reprend le fichier de données de la section 5.2 et dessine en chaque point un vecteur pointant vers l’origine. Elle est obtenue à partir du code suivant :

```
set xrange [-0.2:2.2]
set yrange [-0.2:1.2]
set zeroaxis
plot "courbes.dat" u 1:3:(-0.1*$1/$3):(-0.1) with vectors head filled, \
      "" u 1:3 with linespoints lc rgb "blue" ps 2
```

5.6 Diagrammes de dispersion

Avec le style *dots*, le symbole utilisé est un petit point, ce qui convient bien à la représentation de nuages ou de diagrammes de dispersion (*scatterplot*).

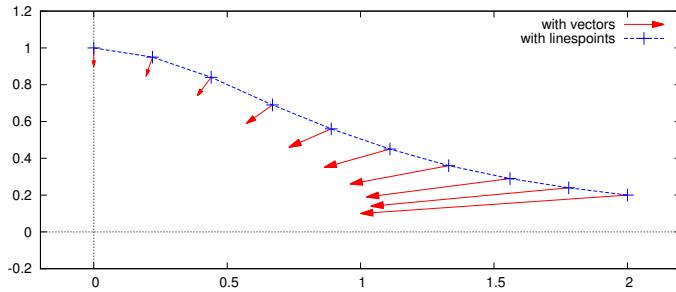


FIGURE 44 – Exemple de style *vectors*.

Le graphe représenté sur la figure 45 est un nuage de 5000 points dont les données sont contenues dans un fichier *nuage.dat* (qui n'est pas reproduit ici). Le code pour obtenir cette figure est :

```
set key off
set title "Nuage de 5000 points"
plot "nuage.dat" w dots
```

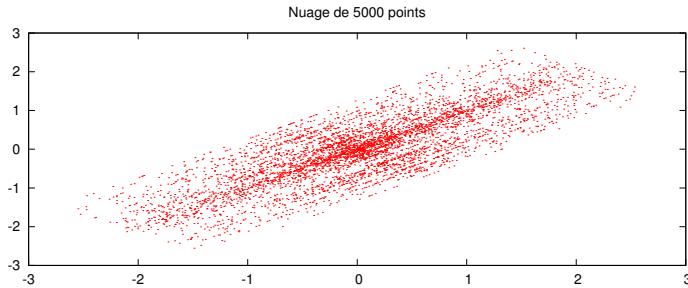


FIGURE 45 – Exemple de style *dots*

5.7 Fonctions en escalier

Les styles suivants représentent des fonctions en escalier passant par les points spécifiés dans un fichier de données ou calculés dans le cas d'une fonction : *steps*, *fsteps*, *histeps*, *histograms*. La différence réside dans la position des points par rapport aux segments horizontaux : dans le cas du style *steps*, chaque point constitue l'extrémité gauche des segments, dans le style *fsteps* ils constituent l'extrémité droite des segments et dans le style *histeps* ils en sont le milieu. Le style *boxes* est similaire au style *histeps* mais dessine en plus des lignes verticales depuis les extrémités des segments horizontaux jusqu'à l'axe des *x*.

Voici quelques exemples avec un fichier de données *steps.dat* contenant les valeurs suivantes :

```
1
2
4
6
```

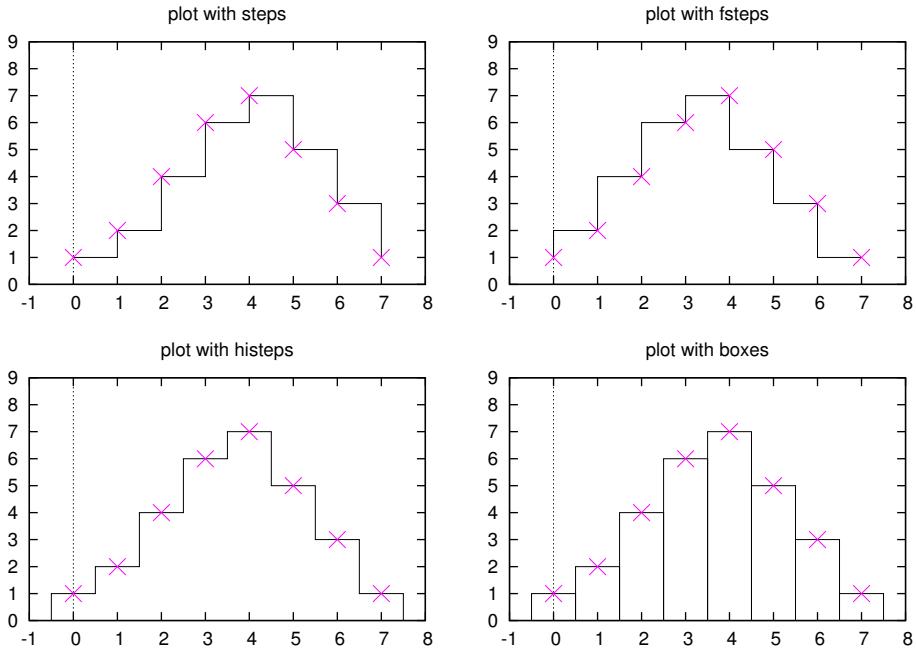


FIGURE 46 – Exemples de styles *points*, *lines*, *impulses*, *boxes*.

```
7
5
3
1
```

S'il n'y a qu'une seule colonne de données, **gnuplot** les prend pour des ordonnées correspondant aux abscisses entières en commençant à 0.

Les graphes de la figure 46 ont été obtenus avec les instructions suivantes respectivement :

```
set xrange [-1:8]
set yrange [0:9]
set style line 15 pt 2 ps 2 lc 4
plot "steps.dat" with steps, "" with points ls 15
plot "steps.dat" with fsteps, "" with points ls 15
plot "steps.dat" with histeps, "" with points ls 15
plot "steps.dat" with boxes, "" with points ls 15
```

Dans le cas du style *boxes*, on peut régler la largeur des bandes verticales avec l'option *boxwidth*. Cela a pour effet d'introduire de l'espace entre les bandes. Par exemple :

```
set boxwidth 0.5
```

5.8 Histogrammes

Le graphe de la figure 47 donne un exemple du style *histograms* :

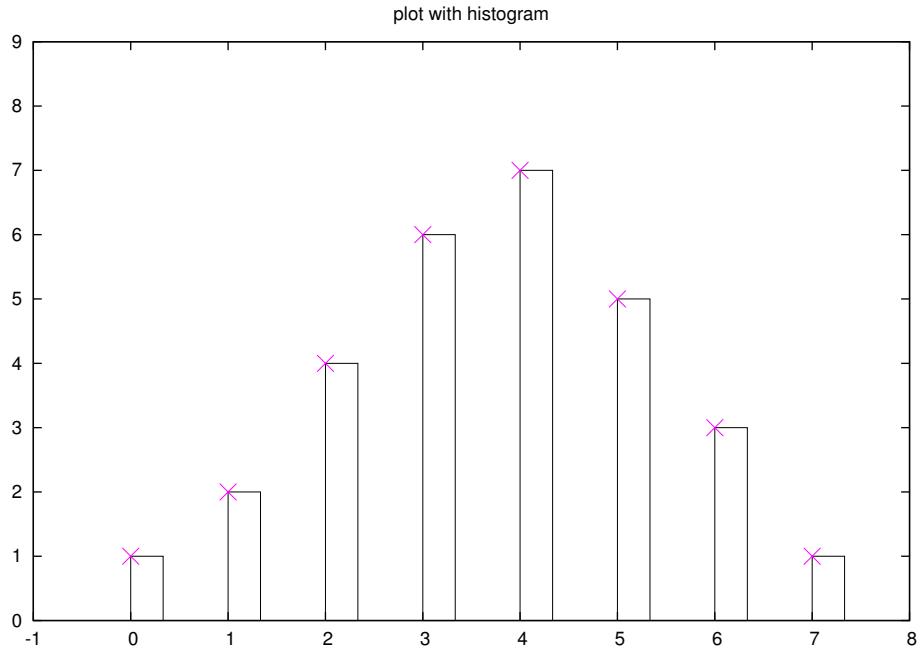


FIGURE 47 – Exemple de style *histograms* simple.

```
plot "steps.dat" with histograms, "" with points ls 15
```

Il correspond aux mêmes données qui ont été utilisées pour les fonctions en escalier à la section 5.7.

Le style *histograms* peut réaliser des histogrammes plus complexes dans lesquels plusieurs séries de données sont disponibles.

Considérons un nouveau fichier de données, avec trois colonnes comme ceci :

```
1    7    3
2    4    5
4    3    6
6    5    2
7    4    1
5    7    5
3    4    3
1    2    4
```

On déclare le style *histograms* au moyen d'une commande *set style data* et on exécute une commande **plot** en spécifiant successivement les colonnes 1, 2 et 3 avec une option *using* comme ceci :

```
set style data histograms
plot "histogram.dat" using 1, "" using 2 lc 3, "" using 3 lc 4
```

Le résultat est le graphe de la figure 48.

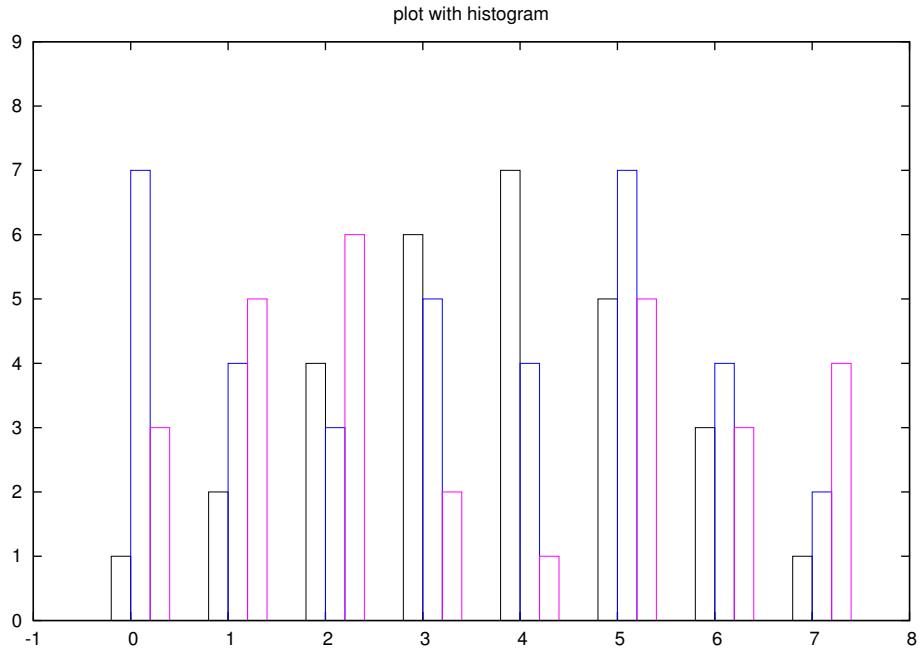


FIGURE 48 – Exemple de style *histograms* avec 3 colonnes.

Pour que les boîtes de l'histogramme soient remplies avec des couleurs différentes, on peut spécifier l'argument *fill* de l'option *style* comme ceci (la valeur 0.5 est la densité) :

```
set style fill solid 0.5
```

Le type d'histogramme vu précédemment s'appelle *clustered*. Il existe deux autres types d'histogrammes dans lesquels les boîtes sont accolées en colonnes verticales ou en rangées horizontales : on les appelle *rowstacked* et *columnstacked* respectivement. On les sélectionne au moyen d'une commande *set style histogram*.

Les graphes de la figure 49 ont été obtenus avec les instructions suivantes respectivement :

```
set style data histograms
set style histogram columnstacked
set style fill pattern 0 border 0
plot "histogram.dat" using 1, "" using 2 lc 3, "" using 3 lc 4

set style histogram rowstacked
set style fill solid 0.5 border -1
plot "histogram.dat" using 1, "" using 2 lc 3, "" using 3 lc 4
```

5.9 Barres d'erreur

Il existe une collection de styles qui permettent de dessiner des intervalles d'erreurs autour des valeurs numériques représentées graphiquement. Ces styles

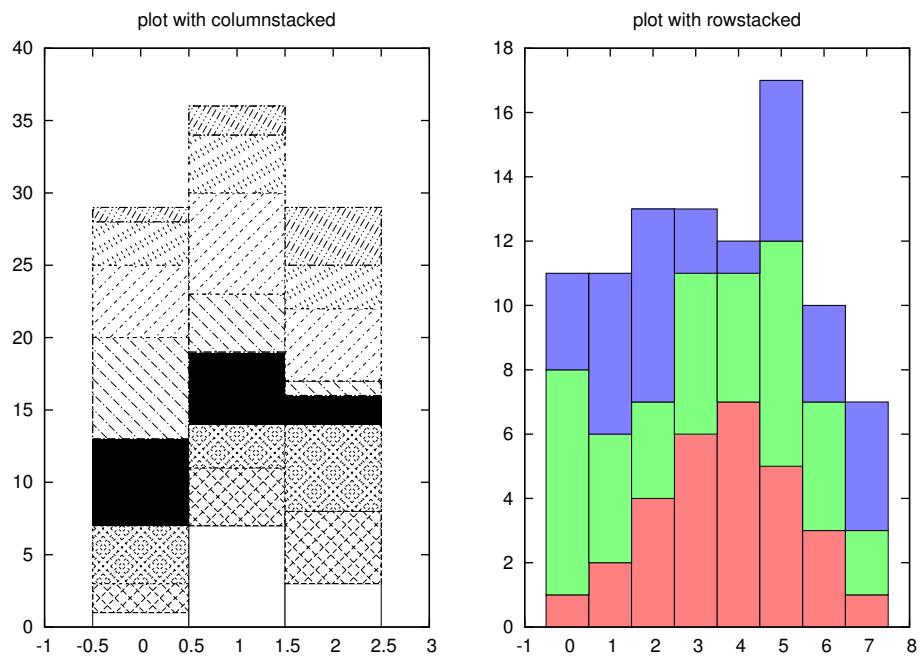


FIGURE 49 – Exemple de style *histograms* empilés.

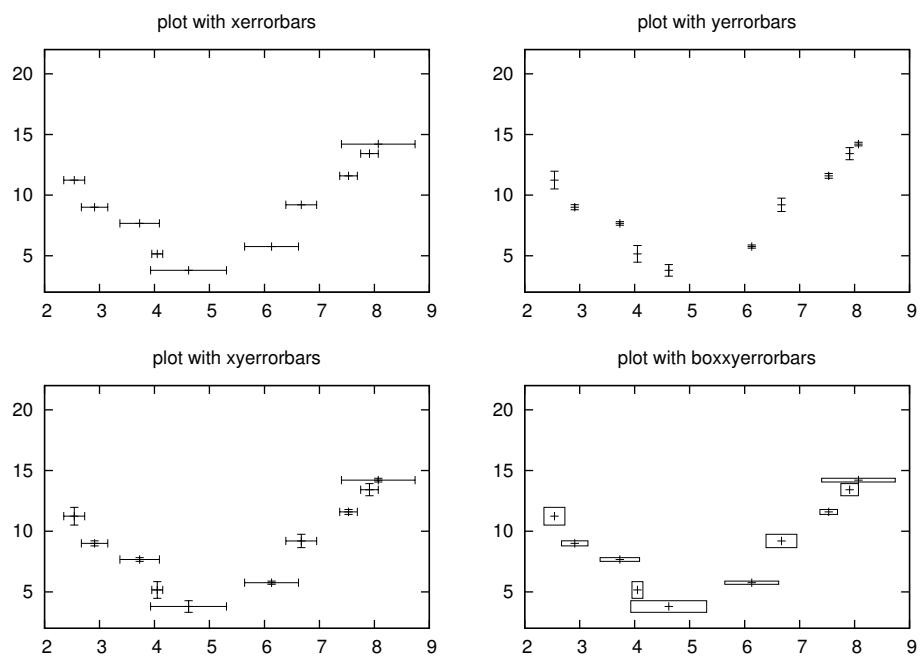


FIGURE 50 – Exemples de styles *errorbars* symétriques.

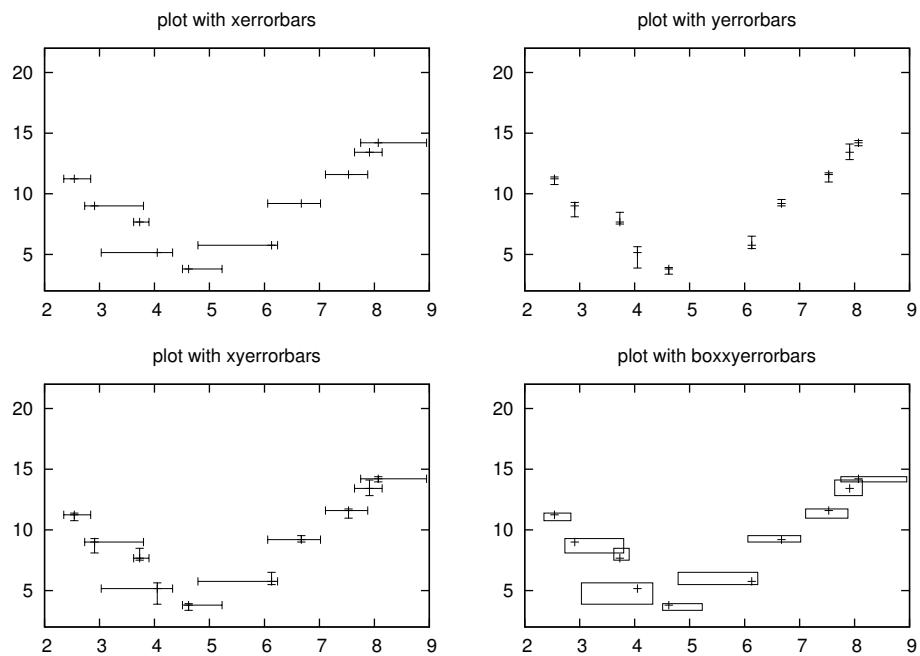


FIGURE 51 – Exemples de styles *errorbars* asymétriques.

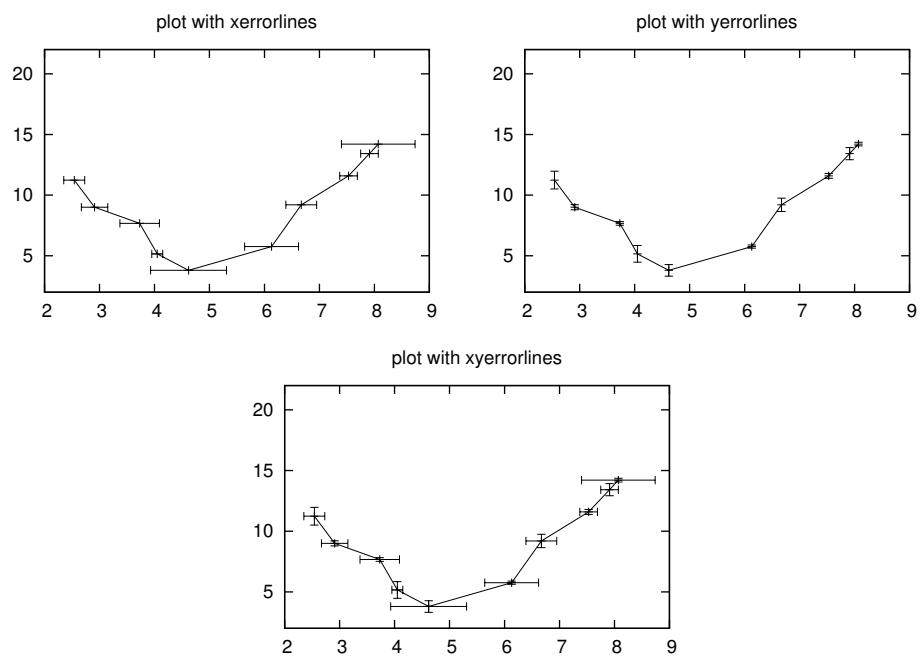


FIGURE 52 – Exemples de styles *errorlines*.

requièrent plus de données que les styles simples vus à la section 5.4 puisqu'il faut spécifier les bornes de l'intervalle dans lequel se trouvent les valeurs représentées. Chaque intervalle est représenté graphiquement par un segment joignant les deux bornes et passant par le point encadré. Ce segment est horizontal dans le cas d'une abscisse x et vertical dans le cas d'une ordonnée y .

Il y a deux manières de caractériser les intervalles. S'ils sont symétriques, il suffit de donner la quantité Δx ou Δy représentant le rayon de l'intervalle. Par exemple, dans le cas d'une abscisse, un intervalle symétrique s'écrit

$$[x - \Delta x, x + \Delta x].$$

Si les intervalles ne sont pas symétriques, on doit spécifier la borne inférieure et la borne supérieure, l'intervalle étant alors $[x_{min}, x_{max}]$.

Les différents styles disponibles sont *xerrorbars* qui dessine des intervalles horizontaux portant sur les abscisses, *yerrorbars* qui dessine des intervalles horizontaux portant sur les ordonnées, *xyerrorbars* qui cumule les deux précédents, et *boxxyerrorbars* qui remplace les segments par des rectangles entourant les points. Les figures 50 et 51 en donnent des exemples dans les cas d'intervalles symétriques et asymétriques respectivement.

Le nombre de colonnes nécessaires dépend du style et du type d'intervalle : les deux premières colonnes représentent les abscisses et les ordonnées. Pour le style *xerrorbars*, par exemple, il faudra une colonne supplémentaire représentant les Δx si on veut des intervalles symétriques, ou deux colonnes supplémentaires représentant les bornes inférieures et supérieures des intervalles.

Si les colonnes du fichier de données sont plus nombreuses ou ne sont pas dans l'ordre attendu, il faut utiliser une option *using*.

Les exemples des figures 50 à 52 utilisent le fichier de données suivant appelé *errorbars.dat* :

2.54	11.24	0.19	0.73	0.19	0.3	0.48	0.15
2.91	9	0.24	0.21	0.18	0.89	0.9	0.29
3.73	7.67	0.36	0.15	0.11	0.17	0.16	0.81
4.05	5.16	0.10	0.69	1.02	0.28	1.28	0.48
4.62	3.8	0.69	0.48	0.11	0.61	0.43	0.12
6.13	5.76	0.49	0.13	1.34	0.11	0.27	0.75
6.67	9.2	0.28	0.55	0.61	0.35	0.2	0.33
7.53	11.59	0.16	0.2	0.42	0.35	0.62	0.14
7.91	13.42	0.16	0.5	0.27	0.23	0.6	0.69
8.07	14.21	0.67	0.15	0.32	0.88	0.25	0.18

dans lequel les colonnes représentent respectivement x , y , Δx , Δy , x_{min} , x_{max} , y_{min} et y_{max} .

Les instructions produisant les graphes de la figure 50 sont :

```
plot "errorbars.dat" using 1:2:3 with xerrorbars
plot "errorbars.dat" using 1:2:4 with yerrorbars
plot "errorbars.dat" using 1:2:3:4 with xyerrorbars
plot "errorbars.dat" using 1:2:3:4 with boxxyerrorbars, "" with points
```

Les instructions produisant les graphes de la figure 51 sont :

```
plot "errorbars.dat" using 1:2:5:6 with xerrorbars
plot "errorbars.dat" using 1:2:7:8 with yerrorbars
```

```
plot "errorbars.dat" using 1:2:5:6:7:8 with xyerrorbars
plot "errorbars.dat" using 1:2:5:6:7:8 with boxxyerrorbars, "" with points
```

Il existe en outre trois styles appelés *xerrorlines*, *yerrorlines*, *xyerrorlines* qui ont la même signification que *xerrorbars*, *yerrorbars*, *xyerrorbars* mais qui joignent en plus les points entre eux comme dans le style *lines*. Des exemples en sont donnés sur la figure 52.

Les instructions produisant les graphes de la figure 52 sont :

```
plot "errorbars.dat" using 1:2:3 with xerrorlines
plot "errorbars.dat" using 1:2:4 with yerrorlines
plot "errorbars.dat" using 1:2:3:4 with xyerrorlines
```

On peut contrôler la taille des terminaisons des barres d'erreur au moyen de l'option *bars*. Les valeurs possibles pour cette option sont *small*, *large*, *fullwidth* ou une dimension : *small* correspond à 0.0 et *large* à 1.0.

La valeur *fullwidth* concerne seulement les histogrammes (*histograms with errorbars*) et indique que les terminaisons des barres d'erreur doivent avoir la même largeur que les blocs de l'histogramme.

5.10 Le style *pm3d*

5.10.1 L'algorithme *pm3d*

Cette section décrit le fonctionnement de l'algorithme de colorisation des surfaces dit *pm3d* afin de mieux comprendre les arguments de l'option du même nom.

Il est basé sur les isocourbes utilisées pour la représentation de graphiques de type *lines*. Il parcourt les points le long de deux isocourbes voisines et considère les quadrillatères formés par deux points successifs et les points homologues sur l'isocourbe suivante.

Par exemple, si les points P_i et P_{i+1} sont deux points successifs d'une isocourbe, l'algorithme leur adjoint les points de même indice Q_i et Q_{i+1} de l'isocourbe voisine. Pour ces quatre points, il calcule les cotes z correspondantes et effectue leur moyenne. La valeur obtenue est comparée aux valeurs minimale et maximale permises pour z puis est ajustée pour donner une valeur équivalente dans l'intervalle $[0, 1]$: cette valeur obtenue entre 0 et 1 fournit le niveau de gris utilisé pour remplir le quadrilatère si la palette courante est en niveaux de gris ou sert à calculer une couleur associée si la palette courante est en couleurs. Les mécanismes de correspondance entre niveaux de gris et couleurs est détaillé à la section 8.3.3.

Pour obtenir les meilleurs résultats, il est souhaitable que les isocourbes aient approximativement le même nombre de points (ce qui n'est pas nécessairement le cas avec des données obtenues depuis un fichier).

Il y a plusieurs variantes possibles de cet algorithme. Par exemple, l'ordre dans lequel les points d'une isocourbe sont parcourus peut être modifié : on peut partir du premier point ou du dernier ou encore du centre de l'isocourbe. D'autre part, la valeur calculée à partir des cotes n'est pas forcément la moyenne des cotes : ce peut être la médiane, la moyenne géométrique, la valeur en l'un seulement des quatre sommets, etc.

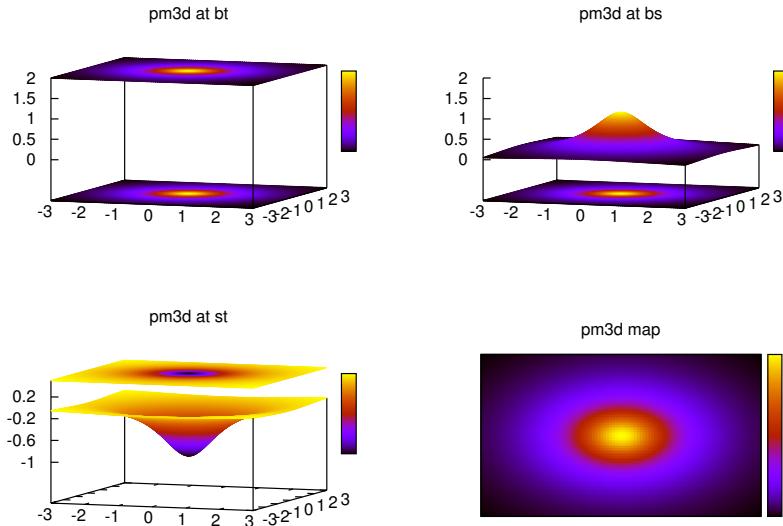


FIGURE 53 – Exemple d’option *pm3d at*.

5.10.2 L’option *pm3d*

Il ne faut pas confondre le style *pm3d* qui est utilisé dans des commandes de la forme *splot...with pm3d* avec l’option *pm3d* utilisée dans des commandes *set pm3d*.

Les arguments de l’option *pm3d* permettent de contrôler le fonctionnement de l’algorithme.

La syntaxe complète est :

```
set pm3d {
  { at <bst combination> }
  { interpolate <steps/points in scan, between scans> }
  { scansautomatic / scansforward / scansbackward / depthorder }
  { flush { begin / center / end } }
  { ftriangles /noftriangles }
  { clip1in / clip4in }
  { corners2color { mean/geomean/median/min/max/c1/c2/c3/c4 } }
  { hidden3d <linestyle> / nohidden3d }
  { implicit / explicit }
  { map }
}
```

L’argument *at* indique les emplacements ou les zones de couleurs sont dessinées. On utilise n’importe quelle combinaison des lettres *b*, *s* et *t* : la lettre *b* veut dire *bottom* et projette les couleurs sur le plan de base, la lettre *s* veut dire *surface* et colorie directement la surface, la lettre *t* veut dire *top* et projette les couleurs sur le plan supérieur du cube contenant la surface.

Le graphe représenté sur la figure 53 illustre l’argument *at*. Le code complet est le suivant :

```
set key off
set xrange [-3:3]
```

```

set yrang [ -3:3]
set size 1,1
set xtics offset 0,-0.5
set ytics offset 0,-0.5
unset cbtics
set view 80,20,1,1
set isosamples 100
f(x,y) = 1/(1+x**2+y**2)
set multiplot layout 2,2 rowsfirst

set zrange [0:2]
set border 255
set pm3d at bt
set title "pm3d at bt"
splot f(x,y) with pm3d

set border 31
set pm3d at bs
set title "pm3d at bs"
splot f(x,y) with pm3d

set pm3d at st
set zrange [-1:0.5]
set ztics -1,0.4,0.4
set title "pm3d at st"
splot -f(x,y) with pm3d
set zrange [0:1]
unset xtics
unset ytics

set pm3d map
set title "pm3d map"
splot f(x,y) with pm3d
unset multiplot

```

L’argument *interpolate* permet d’affiner la trame utilisée pour calculer les points de la surface. Il concerne essentiellement les graphes obtenus à partir de fichiers de données : dans le cas d’un graphe défini par une fonction, il est plus efficace d’augmenter les valeurs des options *samples* et *isosamples* pour obtenir une surface plus régulière.

Les arguments *scansforward* et *scansbackward* permettent de spécifier l’ordre dans lequel les isocourbes sont parcourues tandis qu’avec l’argument *scansautomatic*, **gnuplot** essaie de déterminer lui-même le meilleur choix. Ces trois arguments cohabitent mal avec l’option *hidden3d* qui permet de ne pas dessiner les portions de surface masquées par celles qui sont au premier plan. On peut obtenir de meilleurs résultats si on spécifie l’argument *depthorder* qui ordonne les quadrillatères selon un algorithme de tri en profondeur.

L’argument *flush* suivi d’un des mots-clés *begin*, *center*, *end* permet de spécifier l’ordre dans lequel sont parcourus les points eux-mêmes sur chaque isocourbe. Cet ordre peut avoir une influence sur le rendu de la surface.

Il est difficile de prédire le résultat visuel de toutes ces options et elles sont en général déterminées par tâtonnement.

Un autre paramètre de l’algorithme sur lequel on peut influer est la manière dont est calculée la valeur de la cote *z* à partir de laquelle sera déterminée la teinte du quadrilatère. Par défaut, **gnuplot** effectue la moyenne arithmétique des cotes des quatre sommets de chaque quadrilatère mais d’autres choix sont possibles au moyen de l’argument *corners2color* qui peut être suivi de l’un des

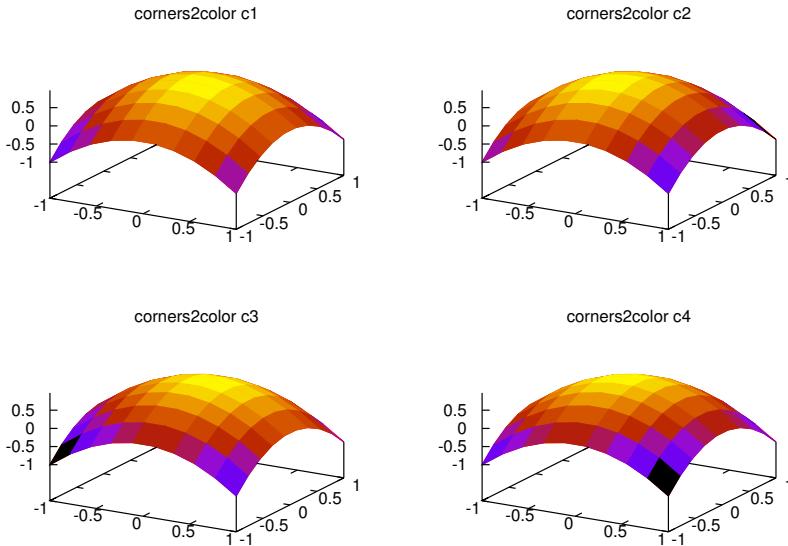


FIGURE 54 – Exemples d’argument *corners2color*

mot-clés suivants : *mean*, *geomean*, *median*, *min*, *max*, *c1*, *c2*, *c3*, *c4*. Les quatre dernières valeurs désignent l’un des quatre coins du quadrilatère. Ici aussi, la détermination du meilleur choix se fait par tâtonnement.

Les graphes représentés sur la figure 54 illustrent quelques-unes de ces méthodes. Ils sont obtenus respectivement avec les instructions suivantes :

```
f(x,y) = 1 - x*x - y*y
set multiplot layout 2,2 rowsfirst
set pm3d at s corners2color c1; splot f(x,y)
set pm3d at s corners2color c2; splot f(x,y)
set pm3d at s corners2color c3; splot f(x,y)
set pm3d at s corners2color c4; splot f(x,y)
unset multiplot
```

Les arguments *clip1in* et *clip4in* concernent le comportement à adopter lorsque l’un des quadrilatères traverse les intervalles délimitant les valeurs des *x* ou des *y*. *clip1in* signifie que les quatre sommets du quadrilatère doivent être définis et qu’au moins l’un d’eux se trouve dans les intervalles. *clip4in* signifie que les quatre sommets doivent être dans les intervalles, autrement le quadrilatère n’est pas dessiné.

L’argument *hidden3d* de l’option *pm3d* (à ne pas confondre avec l’option *hidden3d* utilisée avec la commande *set*) fait en sorte que les lignes soient dessinées selon un style donné tout en prenant en compte les régions masquées par des portions de surface au premier plan. La documentation affirme que c’est une méthode plus efficace que d’utiliser l’option *hidden3d*. La valeur de cet argument est un style qui doit être défini par ailleurs au moyen d’une commande *set style line*.

Les graphes représentés sur la figure 55 illustrent la différence entre les deux méthodes. Ils sont obtenus respectivement avec les instructions suivantes :

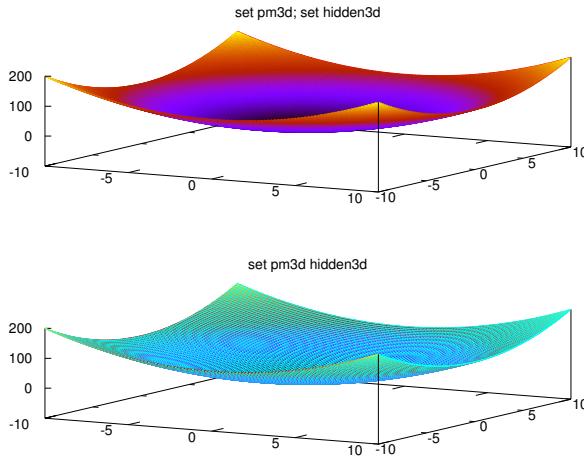


FIGURE 55 – Exemple d’argument *hidden3d*

```

set pm3d at s
set hidden3d
set isosamples 100
splot x*x+y*y

set pm3d at s hidden3d 15
set style line 15 lt 5 lw 0.5
unset hidden3d
unset surface
splot x*x+y*y

```

Finalement l’argument *map* fournit une représentation plane de la projection de la surface sur le plan des (x, y) . La commande *set pm3d map* est équivalente à *set pm3d at b ; set view map*.

5.11 Les images

gnuplot possède plusieurs styles qui permettent l’inclusion d’images dans des graphiques : *image*, *rgbimage* et *rgbalpha*.

Ces styles sont inséparables de la notion de fichier binaire et de la syntaxe utilisée par **gnuplot** pour lire ce type de fichiers. On se reportera aux sections 4.1.2 et 4.3.1 pour des exemples.

6 Ajustement de données

La commande *fit* permet d’exécuter des algorithmes d’ajustement de données pour calculer les paramètres d’un modèle qui est supposé ajuster au mieux des données numériques.

Le principe est le suivant : on dispose de données numériques stockées sous forme de colonnes d’un tableau et on envisage une fonction f dépendant de certains paramètres inconnus qui est censée décrire une relation entre les données.

Le problème est de déterminer des valeurs pour les paramètres inconnus de telle sorte que la fonction f correspondante ajuste au mieux les données recueillies dans le fichier.

La notion de meilleur ajustement est estimée par la somme des carrés des résidus, c'est-à-dire la somme des carrés des différences entre les données observées et celles que fournirait la fonction f .

6.1 Exemple simple

Voici un exemple concret. On suppose qu'on a recueilli des données à la suite de l'observation de deux variables x et y au cours d'une expérience renouvelée 15 fois. Les données sont stockées dans un fichier appelé par exemple *experim.dat* sous forme de 15 lignes comportant chacune les valeurs observées, comme ceci :

2.7	6.05
3.7	9.95
5.7	11.35
9.1	14.85
2	3.9
9	15.2
9.4	15.8
6.6	11.5
6.3	11.75
0.6	2
2.1	5.55
1.8	3.5
6.9	12.15
3.8	8.1
7.7	13.65

On considère que la première colonne représente les valeurs de x et la seconde celles de y . On se pose la question de savoir s'il existe une relation linéaire entre les deux variables, autrement dit une relation de la forme $y = ax + b$. Les paramètres a et b sont les paramètres inconnus.

Ce problème est résolu par **gnuplot** au moyen des deux instructions suivantes :

```

1 f(x)= (a*x)+b
2 fit f(x) "experim.dat" using 1:2 via a,b

```

La première instruction a pour but de déclarer le modèle de fonction recherché. La deuxième instruction utilise la commande **fit** : les deux premiers arguments sont la fonction modèle et le nom du fichier de données. L'argument *using 1 :2* signifie que les données sont les colonnes 1 et 2 du fichier : il faut le préciser car le fichier pourrait contenir d'autres colonnes. Enfin l'argument *via a,b* signifie que ce sont les coefficient a et b de la fonction f qu'il faut calculer.

Le calcul effectué par **gnuplot** repose sur l'algorithme NLLS (*nonlinear least-squares*) de Marquardt-Levenberg. La commande affiche des détails sur le déroulement des calculs et les itérations effectuées. Finalement on obtient :

resultant parameter values

```

a          = 1.50707
b          = 1.91021

```

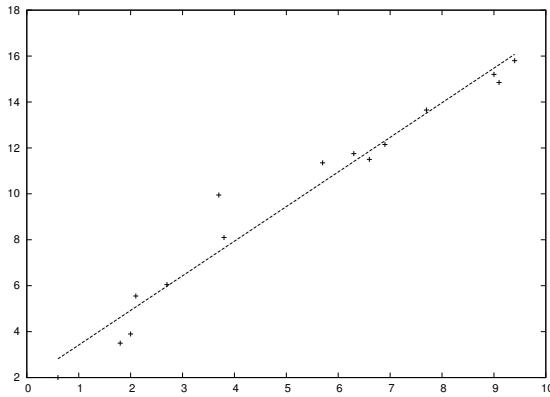


FIGURE 56 – Exemple de régression linéaire.

```

After 4 iterations the fit converged.
final sum of squares of residuals : 11.2512
rel. change during last iteration : -1.08031e-09

degrees of freedom      (FIT_NDF)          : 13
rms of residuals        (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.93031
variance of residuals   (reduced chisquare) = WSSR/ndf   : 0.865477

Final set of parameters            Asymptotic Standard Error
=====                         =====
a                  = 1.50707           +/- 0.08393       (5.569%)
b                  = 1.91021           +/- 0.4952        (25.93%)

```

Les résultats sont stockés dans les variables a et b comme on peut le vérifier directement :

```
gnuplot> print "a=", a, " et b=", b
a=1.50706518204714 et b=1.91021032723335
```

On peut représenter les données graphiquement comme ceci (voir la figure 56) :

```
plot "experim.dat" using 1:2, f(x)
```

On peut maintenant compliquer quelque peu le modèle en cherchant à ajuster les données au moyen d'une fonction quadratique de la forme $y = ax^2 + bx + c$. La commande doit être modifiée comme ceci :

```
fit y=a*x**2+b*x+c "experim.dat" using 1:2 via a,b,c
plot "experim.dat" using 1:2, a*x**2+b*x+c
```

La courbe obtenue est représentée sur la figure 57.

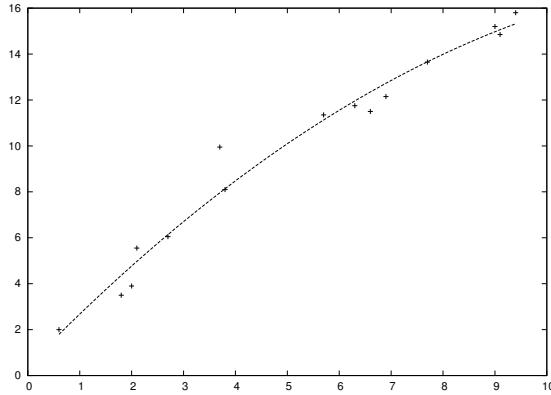


FIGURE 57 – Exemple de régression quadratique.

6.2 La commande `fit`

La syntaxe complète de la commande est :

```
fit {<ranges>} <expression>
'<datafile>', {datafile-modifiers}
via '<parameter file>' / <var1>{,<var2>,...}
```

L’argument *ranges* est optionnel et sert à délimiter des intervalles pour les variables observées. Toute valeur qui ne serait pas dans l’intervalle serait exclue des calculs. On spécifie les intervalles de la même manière qu’avec la commande **plot**.

L’argument *expression* est le nom d’une fonction qui a été déclarée ou une expression algébrique. Par exemple :

```
fit f(x) "experim.dat" using 1:2 via a,b
fit y=a*x**2+b*x+c "experim.dat" using 1:2 via a,b,c
```

L’argument *datafile-modifiers* est optionnel : on peut mettre à cet endroit les mêmes options que celles de la commande **plot** lorsque les données sont lues depuis un fichier (*binary*, *every*, *using*, etc.).

Si le fichier de données contient une unique colonne, **gnuplot** considère que c’est la variable dépendante et que la variable explicative dans ce cas est le numéro de la ligne dans le fichier (autrement les nombres de 1 à *n* si le fichier contient *n* observations).

Si le fichier de données contient deux colonnes, **gnuplot** considère par défaut que la première colonne représente la variable explicative *x* et la deuxième la variable expliquée *y*. Cela correspond à l’option *using 1 :2* qui peut être omise dans ce cas.

Dans un modèle à deux variables de la forme $z = f(x, y)$ le fichier de données doit contenir au moins trois colonnes. Plus précisément, l’option *using* doit être de la forme *x:y:z:s*. Le quatrième terme *s* représente les poids affectés aux observations dans le calcul de la somme des carrés des résidus. Dans le cas le plus simple, on peut considérer que ces poids sont tous égaux à 1. On peut l’indiquer au moyen d’une instruction de la forme *using 1 :2 :3 :(1)*.

Au total, la commande **fit** supporte jusqu'à 5 variables explicatives donc peut calculer des modèles de la forme $z = f(x, y, t, u, v)$. L'option *using* sera dans ce cas de la forme **x:y:t:u:v:z:s**. Par exemple :

```
h(x,y,t,u,v) = a*x + b*y + c*t + d*u + e*v
fit h(x,y,t,u,v) 'experim.dat' using 1:2:3:4:5:6:(1) via a,b,c,d,e
```

Cela suppose que le fichier de données contienne six colonnes représentant respectivement les variables explicatives x, y, t, u, v puis la variable expliquée z .

Si la variable expliquée z était la première colonne du fichier et que x, y, t, u, v soient les suivantes dans cet ordre, l'option *using* devrait s'écrire :

```
using ($2):($3):($4):($5):($6):($1):(1)
```

Variables de contrôle On peut fixer des valeurs pour certaines variables afin d'influencer le déroulement de l'algorithme utilisé par **gnuplot**.

La variable *FIT_LIMIT* fixe une limite pour décréter si une nouvelle itération est nécessaire : si la nouvelle valeur calculée pour la somme des carrés des résidus diffère de la précédente d'une valeur moindre que cette limite alors l'algorithme s'arrête. La valeur par défaut est 1e-5.

La variable *FIT_MAXITER* fixe un nombre maximal d'itérations. Si elle vaut 0 ou n'est pas fixée, il n'y a pas de limite. Pour stopper un calcul qui dure trop longtemps, on peut utiliser la combinaison Ctrl-C.

La variable *FIT_CONVERGED*, à la fin de l'algorithme, vaut 1 si la commande a abouti et 0 si l'algorithme a dû s'interrompre.

```
print FIT_CONVERGED
1
```

Fichier log Chaque fois que la commande **fit** est exécutée, les informations affichées par **gnuplot** sont aussi écrites dans un fichier appelé *fit.log* qui peut être consulté par la suite si nécessaire.

Les nouvelles informations sont ajoutées en fin de fichier sans effacer les précédentes. Donc ce fichier augmente à chaque exécution de la commande **fit**.

On peut modifier le nom du fichier log au moyen de l'instruction *set fit logfile*. Par exemple :

```
set fit logfile "resultat.txt"
```

6.3 L'option *fit*

Il ne faut pas confondre l'option *fit* avec la commande **fit**. L'option *fit* s'utilise, comme toutes les options, au moyen des commandes **show**, **set** et **unset**.

Elle permet en particulier de déclarer le nom du fichier de sortie dans lequel la commande **fit** écrit ce qu'elle produit. Ce fichier s'appelle par défaut *fit.log*. Par exemple :

```
set fit logfile "calculs.txt"
```

La syntaxe complète de cette option est :

```
set fit {logfile "<filename>"}} {{no}errorvariables}
unset fit
show fit
```

Si on spécifie l'option *errorvariables*, on peut obtenir des estimations d'erreur sur les paramètres. Par défaut, **gnuplot** stocke ces estimations dans une variable portant le nom du paramètre avec un suffixe *_err*.

Par exemple, en reprenant l'exemple simple de la section 6.1, pour les paramètres *a* et *b*, on aura donc des variables **a_err** et **b_err** :

```
set fit errorvariables
fit f(x) "experim.dat" using 1:2 via a,b
print a_err
0.0839323871397207
print b_err
0.495243759770381
```

6.4 La commande update

Cette commande permet de sauvegarder les valeurs courantes des paramètres obtenus par la commande **fit** afin de pouvoir les réutiliser par la suite. La syntaxe est la suivante :

```
update 'fichier'
```

Le fichier doit exister préalablement. Par exemple :

```
update "regressionQuad.gp"
```

7 Autres commandes de Gnuplot

Cette section donne des précisions et des exemples concernant certaines commandes de **gnuplot** qui n'ont pas été encore rencontrées dans le reste de ce document ou qui ont été mentionnées rapidement.

7.1 Les commandes cd et pwd

Ces deux commandes ont exactement la même signification et la même syntaxe que les commandes shell Unix de même nom : **pwd** renvoie le répertoire courant et **cd** permet de changer de répertoire. L'argument de la commande **cd** doit être placé entre guillemets (simples ou doubles). Par exemple :

```
cd 'subdir'
cd ".."
```

7.2 La commande clear

La commande *clear* permet d'effacer l'écran graphique ou le périphérique de sortie courant, c'est-à-dire celui qui a été fixé par une commande *set output*.

En la combinant avec les options *origin* et *size*, on peut l'utiliser pour effacer seulement des portions d'un graphique et faire des incrustations. Par exemple :

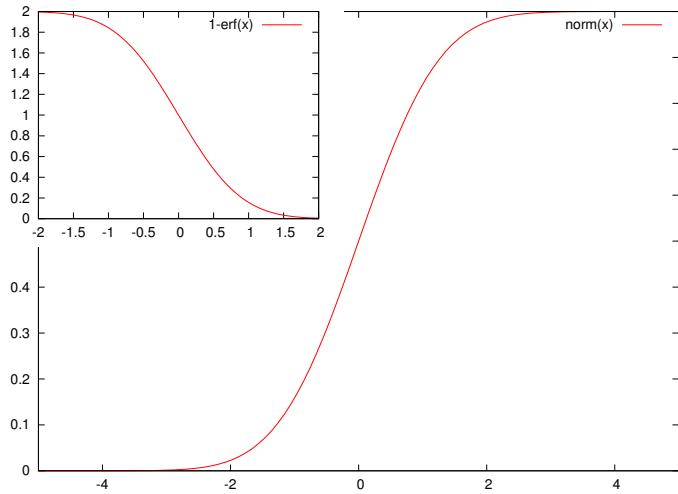


FIGURE 58 – Exemple de commande *clear*.

```
set multiplot
plot [-5:5] norm(x)
set origin 0.0,0.5
set size 0.5,0.5
clear
plot [-2:2] 1-erf(x)
unset multiplot
```

Le graphe est représenté sur la figure 58.

7.3 La commande history

Cette commande est analogue à la commande shell de même nom. Elle permet d'afficher tout ou partie de l'historique des commandes récemment exécutées.

Voici quelques exemples :

```
# Pour afficher l'historique complet
history
# Pour afficher les 5 dernières commandes
history 5
# Pour afficher sans numérotation des commandes
history quiet 5
# Afficher les commandes commençant par "load"
history ?load
# Exécuter la dernière commande commençant par "reread"
hi !reread
```

On peut aussi stocker l'historique dans un fichier afin de le relire par la suite :

```
# Écrire l'historique dans un fichier hist(gp
history "hist(gp"
# Ajouter l'historique au fichier hist(gp
history "hist(gp" append
```

```
# Écrire 5 commandes dans un fichier hist.gp
history 10 "hist.gp"
```

7.4 La commande if

La commande *if* permet de créer des blocs de code conditionnels dans des scripts **gnuplot**. La syntaxe est :

```
if (<condition>) commandes [; else if (<condition>) ...; else ...]
```

Par exemple :

```
x = ceil( rand(0)*100 )
if (x % 2) \
    print sprintf("%d est impair",x); else \
    print sprintf("%d est pair",x);
```

Il est indispensable, dans l'exemple qui précède, de terminer chaque ligne par une contre-oblique pour que **gnuplot** considère qu'il s'agit d'une unique commande logique.

Pour tester l'égalité ou la différence de chaînes de caractères, on utilise les opérateurs **eq** et **ne** plutôt que **==** et **!=**.

7.5 Les commandes save, load et call

La commande **save** permet de sauvegarder les variables et les fonctions définies par l'utilisateur, de même que la plupart des options qui ont été fixées, l'état du terminal courant et la dernière commande **plot** ou **splot** utilisée. On peut aussi ne sauvegarder que certains de ces éléments en précisant l'une des options suivantes : *functions*, *variables*, *terminal* ou *set*.

La syntaxe de la commande est :

```
save [option] "fichier"
```

Un fichier ainsi sauvegardé est un fichier texte et peut ensuite être sourcé au moyen d'une commande *load*. Cette commande *load* permet en fait de sourcer n'importe quel script contenant des instructions **gnuplot**.

Voici quelques exemples :

```
save 'foobar.gp'
save functions 'funcs.dat'
save var 'vars.dat'
save set 'opts.dat'
save term 'term_options.gp'
```

Si on spécifie un tiret, la commande *save* écrit sur la sortie standard :

```
save '-'
```

On peut aussi utiliser un symbole de tube (*pipe*) pour que la sortie soit filtrée à travers d'autres programmes. Par exemple :

```
set title 'Fonction logarithme' tc rgb 'red'  
save '/grep title >t.gp'
```

La commande **call** est analogue à **load** mais permet en plus de passer des arguments à la manière des scripts shell. Les arguments sont désignés dans le fichier script par les symboles $\$0$, $\$1$, $\$2$, ..., $\$9$ ce qui permet de passer jusqu'à 10 arguments. La syntaxe de la commande a la forme suivante :

```
call 'fichier' param0 param1 param2 ... param9
```

Comme avec les scripts shell la variable $\$\#$ désigne le nombre d'arguments passés effectivement dans la commande.

Remarque La notation $\$1$, $\$2$, ... est en conflit avec l'utilisation des mêmes variables pour désigner les colonnes d'un fichier de données dans une commande de dessin comportant l'option *using*. La solution est de désigner les colonnes de données sous la forme $\$\1 , $\$\2 avec un double signe dollar ou d'utiliser la fonction *column* et de les noter *column(1)*, *column(2)* etc.

7.6 La commande pause

La commande **pause** permet de suspendre l'exécution d'un script pendant un temps donné. On lui passe en argument une valeur entière : si c'est un nombre positif, il désigne le nombre de secondes pendant lesquelles l'exécution est suspendue, si c'est 0 il n'y a pas de pause et si c'est -1, l'exécution ne reprendra que lorsque l'utilisateur aura pressé la touche RETOUR.

Un second argument permet de faire afficher un message. Par exemple :

```
pause -1 "Presser Retour pour continuer"
```

Une autre forme de la commande attend un clic de la souris. Par exemple :

```
pause mouse "Cliquer pour continuer"
```

7.7 Les commandes exit et quit

Ces commandes permettent de quitter le programme lorsqu'il est exécuté interactivement. Il y a une petite différence dans la façon dont elles fonctionnent. La commande **exit** quitte le programme inconditionnellement tandis que la commande **quit** ne quitte que le flux courant et, en cas de flux imbriqués (via des commandes **system** ou **shell**), repasse la main au flux parent.

La commande **exit** peut donc laisser certains fichiers dans un état incomplet s'ils n'ont pas été proprement refermés auparavant.

7.8 La commande reread

La commande **reread** permet de réexécuter un script depuis son commencement. Elle n'a de sens que si elle fait elle-même partie du script et n'a aucun effet si on l'invoque depuis la ligne de commandes de **gnuplot**.

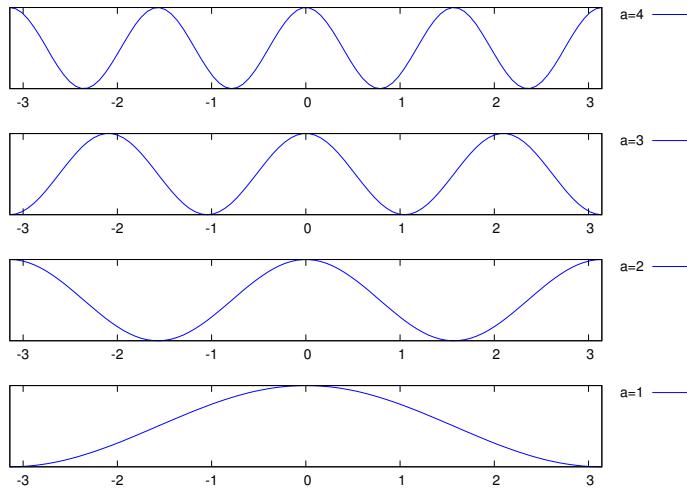


FIGURE 59 – Exemple de commande *reread*.

Elle permet d’implémenter des boucles et de pallier l’absence de commande de type *for* ou *while*. Par exemple, le script suivant trace quatre fonctions cosinus avec des arguments différents sur un même graphique :

```
a = a+1
set origin 0.0,0.25*(a-1)
set size 1,0.25
tt = sprintf("a=%d",a)
plot [-pi:pi] cos(a*x) lt rgb "blue" title tt
if(a<5) reread
```

Supposons que ce code soit stocké dans un fichier appelé *QuatreCos.gp*. On peut l’invoquer depuis **gnuplot** en exécutant les commandes suivantes :

```
set multiplot
a = 0
load 'QuatreCos.gp'
```

On obtient le graphique de la figure 59.

7.9 Les commandes **shell** et **system**

La commande **shell** crée un sous-shell depuis la ligne de commande de **gnuplot**. Une fois qu’on a fini d’utiliser ce sous-shell, on peut retourner à la ligne de commande de **gnuplot** au moyen de la commande *exit* ou avec le symbole de fin de fichier (ctrl-D).

La commande **system** est analogue : elle prend en argument une commande shell et l’exécute dans un sous-processus, puis elle le quitte immédiatement pour repasser le contrôle à **gnuplot**. Cette commande peut être abrégée en un simple point d’exclamation et, dans ce cas, la commande shell ne doit pas être placée entre guillemets. Par exemple, les deux commandes suivantes sont équivalentes :

```
gnuplot> system "uname -m"
gnuplot> ! uname -m
```

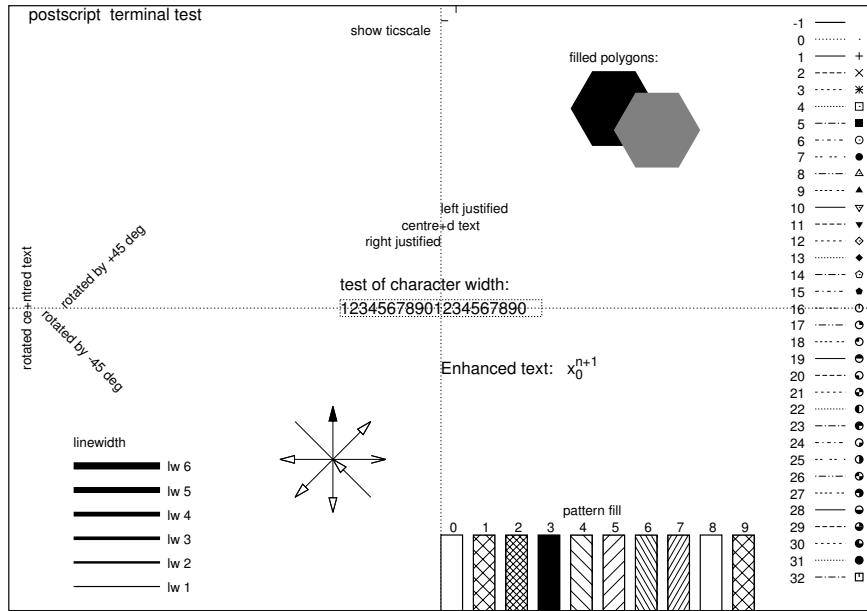


FIGURE 60 – Exemple de commande *test terminal*.

7.10 La commande *test*

Cette commande a deux formes différentes :

```
test terminal
test palette [rgb]
```

La première forme permet d'afficher une référence visuelle rapide comportant les styles et autres éléments utiles disponible pour le terminal courant.

On obtient le graphique de la figure 60.

La deuxième forme de la commande imprime les profils de couleur pour la palette courante. La figure 61 montre le graphique obtenu pour une palette couleur.

De même la figure 62 montre une palette de gris.

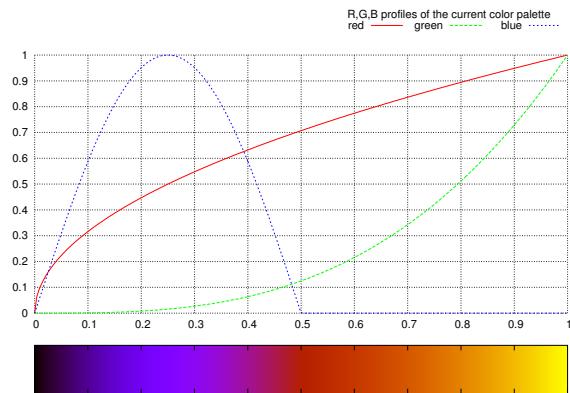


FIGURE 61 – La palette de couleurs.

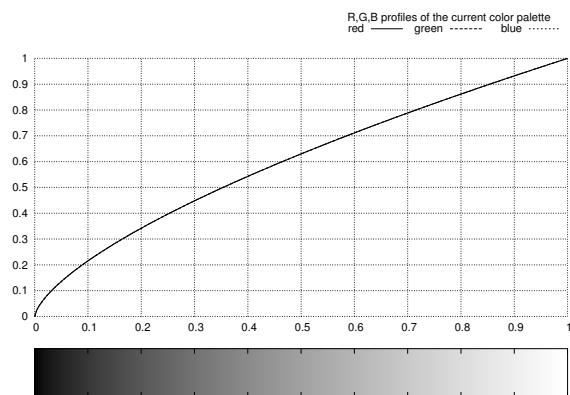


FIGURE 62 – La palette de gris.

8 Annexes

Cette section donne des précisions sur certains types de données manipulés **gnuplot** tels que les dates et les couleurs.

8.1 Les dates

gnuplot définit quelques fonctions utilitaires permettant de manipuler les heures et les dates. La date de référence est le 1er janvier 2000 à 0 heure. Les fonctions suivantes prennent comme argument un temps mesuré en secondes par rapport à cette date et renvoient différents éléments :

<i>tm_hour</i>	heures
<i>tm_min</i>	minutes
<i>tm_sec</i>	secondes
<i>tm_wday</i>	jour de la semaine (dimanche=0)
<i>tm_mday</i>	jour du mois (entre 1 et 31)
<i>tm_mon</i>	mois de l'année (entre 0 et 11)
<i>tm_year</i>	année
<i>tm_yday</i>	jour de l'année (1er janvier=0)

Par exemple, sachant que la valeur 352126777 correspond au dimanche 27 février 2011 à 12 :59 :37 UTC, on peut vérifier les valeurs fournies par les différentes fonctions comme ceci :

```
gnuplot> x = 352126777
gnuplot> print tm_hour(x)
      12.0
gnuplot> print tm_min(x)
      59.0
gnuplot> print tm_sec(x)
      37.0
gnuplot> print tm_wday(x)
      0.0
gnuplot> print tm_mday(x)
      27.0
gnuplot> print tm_mon(x)
      1.0
gnuplot> print tm_year(x)
      2011.0
gnuplot> print tm_yday(x)
      57.0
```

Il faut noter que les mois sont numérotés à partir de 0 : la valeur 1 renvoyée par la fonction *tm_mon* correspond au mois de février. De même, pour les jours de l'année, la valeur 57 renvoyée par la fonction *tm_yday* correspond au 58ème jour de l'année. Pour les jours de la semaine, les valeurs vont de 0 à 6, le 0 correspondant au dimanche.

Les valeurs renvoyées sont en système UTC (*Coordinated Universal Time*) car **gnuplot** ne tient pas compte des heures d'été ou des zones.

La fonction *strftime* applique un format de date à un temps spécifié en secondes depuis le 1er janvier 2000. Par exemple :

```
gnuplot> print strftime("%Y-%m-%d",352126777)
2011-02-27
```

La fonction *strptime* effectue l'opération inverse. Elle analyse une date selon un format et fournit la valeur correspondante en nombre de secondes. Par exemple :

```
gnuplot> print strptime("%Y-%m-%d", "2011-02-27")
352080000.0
```

En combinant cette fonction avec la fonction *system*, on peut obtenir le nombre de secondes depuis le 1er janvier 2000 pour la date du jour comme ceci :

```
gnuplot> auj = system("date '+%Y-%m-%d'")
gnuplot> print auj
2011-02-28
gnuplot> print sprintf("%d", strptime("%Y-%m-%d", auj))
352166400
```

8.2 Les chaînes de caractères

gnuplot définit plusieurs fonctions utilitaires permettant de manipuler les chaînes de caractères. Elles sont inspirées par les fonctions de même nom de la bibliothèque du langage C.

La fonction *strlen* renvoie la longueur d'une chaîne. Par exemple :

```
gnuplot> print strlen("abracadabra")
11
```

La fonction *substr* extrait une sous-chaîne. Par exemple :

```
gnuplot> print substr("abracadabra",5,8)
cada
```

La fonction *strstrt* renvoie l'indice de la première occurrence d'une sous-chaîne dans une chaîne. Par exemple :

```
gnuplot> print strstr("abracadabra", "cada")
5
```

La fonction *system* permet d'exécuter des commandes shell et renvoie le résultat. Par exemple :

```
gnuplot> print system("date")
Mon Feb 28 08:58:02 CET 2011
```

La fonction *word* renvoie le *n*-ième mot d'une chaîne. Par exemple :

```
gnuplot> print word("abra cada bra", 3)
bra
```

La fonction *words* renvoie le nombre de mots d'une chaîne. Par exemple :

```
gnuplot> print words("abra cada bra")
3
```

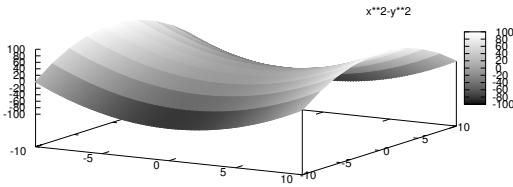


FIGURE 63 – Exemple d’option *palette gray*.

La fonction *sprintf* fonctionne comme la fonction *printf* du langage C et utilise la même syntaxe pour les spécificateurs de format. Par exemple :

```
gnuplot> x=1; y=-4
gnuplot> print sprintf("Point M de coord (%d,%d).",x,y)
Point M de coord (1,-4).
```

La commande **gnuplot** possède aussi des spécificateurs de format qui lui sont propres et ne sont pas ceux de la fonction C *printf*. La documentation sur ces formats peuvent être obtenue par la commande *help format specifiers*. On a en particulier les spécificateurs suivants :

%t	<i>mantissa to base 10</i>
%l	<i>mantissa to base of current logscale</i>
%s	<i>mantissa to base of current logscale; scientific power</i>
%T	<i>power to base 10</i>
%L	<i>power to base of current logscale</i>
%S	<i>scientific power</i>
%c	<i>character replacement for scientific power</i>
%P	<i>multiple of pi</i>

On les utilise par l’intermédiaire de la fonction *gprintf* ou avec l’option *format*. Par exemple :

```
gnuplot> print gprintf("%.P pi",6.283185)
2 pi
gnuplot> print gprintf("%S",1000000)
6
gnuplot> print gprintf("%s+789^{%S}",123456)
123.456000+789^{3}
```

8.3 Les couleurs

L’option *palette* permet de fixer les paramètres relatifs aux nuances de gris ou de couleurs : modèle utilisé, type de palette (gris ou couleur), coefficient gamma, valeurs numériques, etc.

8.3.1 Exemples

La palette est utilisée par les commandes qui remplissent des contours ou des polygones et par le style *pm3d*.

Le graphe représenté sur la figure 63 est obtenu avec les instructions suivantes :

```
set palette gray
splot x**2-y**2 with pm3d
```

On peut spécifier les couleurs avec des commandes de style telles que *linecolor* (en abrégé *lc*), *textcolor* (en abrégé *tc*) ou *linetype* (en abrégé *lt*). La couleur peut être désignée par un nombre ou par une spécification de la forme :

```
rgb "nom_de_couleur"
rgb "#RRGGBB"
```

Par exemple : On utilise ces spécifications directement dans une commande telle que **plot** :

```
plot sin(x), sin(2*x) linecolor 1, sin(3*x) lc rgb 'turquoise'
plot [-4:4] cos(4*x) lt rgb 'navy'
plot [-4:4] sin(4*x) lt rgb '#0080ff'
```

ou par l'intermédiaire d'une définition de style :

```
set style line 10 lt rgb "magenta" lw 2 pt 4
plot [-4:4] cos(x**2) with linespoints ls 10
```

8.3.2 Réglages de la palette

L'instruction *show palette* affiche les principaux réglages mais on peut obtenir des informations plus précises en ajoutant les mots-clés suivants : *palette*, *gradient*, *fit2rgbformulae*, *rgbformulae*, *colornames*. Par exemple, la commande *show palette colornames* fournit les noms des couleurs prédéfinies ainsi que leurs valeurs numériques dans le système RGB. Voici les premières lignes :

```
gnuplot> show palette colornames
      There are 112 predefined color names:
white          #ffffff = 255 255 255
black          #000000 = 0 0 0
dark-grey      #a0a0a0 = 160 160 160
red            #ff0000 = 255 0 0
web-green      #00c000 = 0 192 0
web-blue       #0080ff = 0 128 255
dark-magenta   #c000ff = 192 0 255
etc.
```

La commande *show palette palette* permet d'afficher les valeurs des nuances de gris et des couleurs associées pour constituer une palette de *n* couleurs en utilisant les réglages courants de l'option *palette*. Par exemple :

```
gnuplot> show palette palette 6
Color palette with 6 discrete colors.
0. gray=0.0000, (r,g,b)=(0.0000,0.0000,0.0000), #000000 = 0 0 0
1. gray=0.2000, (r,g,b)=(0.4472,0.0080,0.9511), #7202f3 = 114 2 243
2. gray=0.4000, (r,g,b)=(0.6325,0.0640,0.5878), #a11096 = 161 16 150
3. gray=0.6000, (r,g,b)=(0.7746,0.2160,0.0000), #c63700 = 198 55 0
4. gray=0.8000, (r,g,b)=(0.8944,0.5120,0.0000), #e48300 = 228 131 0
5. gray=1.0000, (r,g,b)=(1.0000,1.0000,0.0000), #ffff00 = 255 255 0
```

En ajoutant les mots-clés *float* ou *int*, on peut afficher seulement les valeurs RGB sous forme de nombres en virgule flottante entre 0 et 1 ou d'entiers entre 0 et 255. Par exemple :

```
gnuplot> show palette palette 6 float
Color palette with 6 discrete colors.
0.0000  0.0000  0.0000
0.4472  0.0080  0.9511
0.6325  0.0640  0.5878
0.7746  0.2160  0.0000
0.8944  0.5120  0.0000
1.0000  1.0000  0.0000

gnuplot> show palette palette 6 int
Color palette with 6 discrete colors.
0      0      0
114    2      243
161    16     150
198    55     0
228    131    0
255    255    0
```

Cette commande peut être utilisée conjointement avec la commande *set print* pour écrire les valeurs dans un fichier comme ceci :

```
set print "color.tbl"
show palette palette 256
exit
```

Pour changer de type de palette, on utilise l'une des valeurs *gray* ou *color* :

```
set palette gray
set palette color
```

Pour changer de modèle (espace colorimétrique), on utilise la sous-commande *model* dont les valeurs possibles sont : RGB pour le système additif Red-Green-Blue, HSV pour le système Hue-Saturation-Value, CMY pour le système sous-traitif Cyan-Magenta-Yellow, YIQ pour le système Color Television Broadcast-ing, XYZ pour le système de la CIE (Commission Internationale de l'Éclairage). Dans chacun de ces systèmes, les couleurs sont représentées par un triplet de valeurs (a, b, c) . Par exemple, les commandes suivantes permettent de basculer entre les systèmes CMY et RGB et de comparer les valeurs de quatre couleurs :

```
gnuplot> set palette model CMY
gnuplot> show palette palette 4 int
Color palette with 4 discrete colors.
255  255  255
108  246  34
47   179  255
0    0    255

gnuplot> set palette model RGB
gnuplot> show palette palette 4 int
Color palette with 4 discrete colors.
```

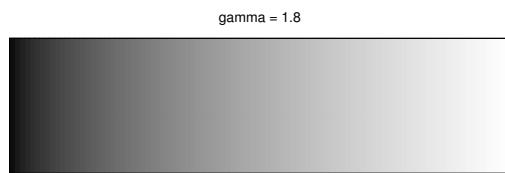
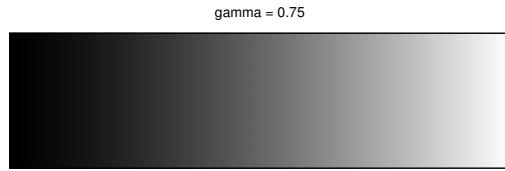


FIGURE 64 – Exemple d’option *palette gamma*

```

0      0      0
147    9     221
208   76      0
255  255      0

```

Le coefficient de correction *gamma* peut être fixé au moyen de l’argument *gamma*. Il concerne seulement les palettes de gris, pas les palettes de couleurs. La valeur par défaut est 1,5. Par exemple :

```
set palette gamma 0.75
```

Plus le coefficient *gamma* est faible et plus le nuancier comporte de teintes sombres comme on peut le voir sur la figure 64 obtenue avec les instruction suivantes :

```

set palette gray
set pm3d map
unset colorbox
unset tics
set multiplot layout 2,1
set palette gamma 0.75
set title "gamma = 0.75"
splot x
set palette gamma 1.8
set title "gamma = 1.8"
splot x
unset multiplot

```

8.3.3 Correspondance gris-couleurs

Les nuanciers de gris ou de couleurs sont conceptuellement des espaces à une dimension. Par exemple, le nuancier de gris peut être vu comme une séquence de valeurs croissantes dans l’intervalle $[0, 1]$.

Les algorithmes de couleur de **gnuplot** calculent en général une valeur de paramètre *t* entre 0 et 1 pour déterminer une teinte. Si la palette de gris est en

vigueur, cette valeur t peut servir directement à déterminer le niveau de gris. Si c'est la palette de couleurs qui est sélectionnée, il faut associer une couleur à la valeur t . C'est pourquoi on parle de correspondance gris-couleurs.

Le nuancier de couleurs peut être vu comme un courbe dans l'espace colorimétrique : **gnuplot** utilise une représentation paramétrique de cette courbe, c'est-à-dire calcule les trois coordonnées d'un point de l'espace colorimétrique en fonction d'un seul paramètre t . Les formules permettant de calculer ces coordonnées paramétriques sont désignées sous le nom de *rgbformulae*.

Il existe un certain nombre de formules *rgbformulae* prédéfinies. On peut les afficher au moyen de la commande suivante :

```
gnuplot> show palette rgbformulae
* there are 37 available rgb color mapping formulae:
  0: 0           1: 0.5          2: 1
  3: x           4: x^2          5: x^3
  6: x^4          7: sqrt(x)      8: sqrt(sqrt(x))
  9: sin(90x)     10: cos(90x)    11: |x-0.5|
 12: (2x-1)^2    13: sin(180x)    14: |cos(180x)|
 15: sin(360x)    16: cos(360x)   17: |sin(360x)|
 18: |cos(360x)| 19: |sin(720x)| 20: |cos(720x)|
 21: 3x          22: 3x-1         23: 3x-2
 24: |3x-1|        25: |3x-2|        26: (3x-1)/2
 27: (3x-2)/2    28: |(3x-1)/2|   29: |(3x-2)/2|
 30: x/0.32-0.78125 31: 2*x-0.84 32: 2*x - 1
 33: |2*x - 0.5| 34: 2*x          35: 2*x - 0.5
 36: 4x;1;-2x+1.84;x/0.08-11.5
* negative numbers mean inverted=negative colour component
* thus the ranges in 'set pm3d rgbformulae' are -36..36
```

Chacune de ces formules porte un numéro. En choisissant trois formules, on établit ainsi la correspondance entre le paramètre et les trois coordonnées de l'espace colorimétrique. La commande suivante permet de connaître le réglage courant :

```
gnuplot> show palette fit2rgbformulae
Current palette is
  set palette rgbformulae 7,5,15
```

Sur cet exemple, on voit que les *rgbformulae* portant les numéros 7, 5 et 15 dans la liste précédente sont utilisées pour calculer les trois coordonnées dans l'espace courant, autrement les valeurs de rouge, de vert et de bleu si le système en vigueur est le système RGB. On peut spécifier des valeurs négatives pour faire en sorte que les couleurs soient inversées, ce qui signifie que les formules seront appliquées à la valeur $1 - t$ plutôt qu'à t , si t est le paramètre de gris qui est passé.

Le mécanisme ci-dessus n'est pas le seul possible pour établir la correspondance entre une valeur de gris et une couleur. En fait, quatre méthodes sont possibles et peuvent être spécifiées au moyen de l'option *palette* :

rgbformulae : c'est la méthode qui vient d'être décrite ;

functions : cette méthode est analogue à la précédente mais permet de définir ses propres formules pour calculer les trois coordonnées de l'espace colorimétrique. Il faut fournir trois expressions en fonction d'une variable appelée symboliquement *gray* supposée entre 0 et 1. La valeur produite doit être dans l'intervalle [0, 1]. Par exemple :

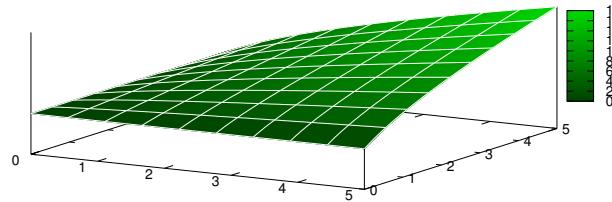


FIGURE 65 – Exemple d'option *set palette function*

```
set palette model XYZ functions gray**0.35, gray**0.5, gray**0.8
```

Voici un exemple qui établit un nuancier de vert en mettant les composantes de rouge et de bleu à 0 et en définissant une fonction linéaire du paramètre *gray* qui représente la nuance *t* de gris à convertir :

```
set palette functions 0, gray*0.6+0.25, 0
set pm3d at s
set xrange [0:5]
set yrange [0:5]
set style lines 10 lc rgb "white"
splot x*log(1+y) + y*log(1+x) ls 10
```

Le graphe est représenté sur la figure 65.

defined : cette méthode fournit une liste explicite de correspondances gris-couleur. Cette liste est utilisée ensuite par interpolation. Chaque élément de la liste comporte une valeur de gris et une valeur de couleur : cette dernière peut être exprimée sous forme d'un triplet de coordonnées ou par son nom ou par sa valeur hexadécimale. Les éléments de la liste sont séparés par des virgules. Voici quelques exemples :

```
set palette defined ( 0 "blue", 1 "yellow", 2 "red" )
set palette defined ( 0 0 0 1, 1 1 1 0, 2 1 0 0 )
set palette defined ( 0 "#0000ff", 1 "#ffff00", 2 "#ff0000" )
set palette model HSV
set palette defined ( 0 0 1 0, 1 0 1 1, 6 0.8333 1 1, 7 0.8333 0 1)
```

Si les valeurs de gris fournies ne sont pas entre 0 et 1, **gnuplot** se charge de les ajuster.

file : cette méthode est analogue en principe à la précédente mais lit les valeurs à partir d'un fichier. Le fichier doit contenir trois ou quatre colonnes. S'il y a quatre colonnes, elles représentent le gris et les coordonnées des couleurs. S'il y a trois colonnes, ce sont les couleurs et le gris sera le rang de chaque ligne. L'argument *using* peut être utilisé pour préciser les colonnes à utiliser. Par exemple, en supposant que les colonnes sont des nombres entiers entre 0 et 255 :

```
set palette model RGB
set palette file 'color.table' using ($1/255):($2/255):($3/255)
```

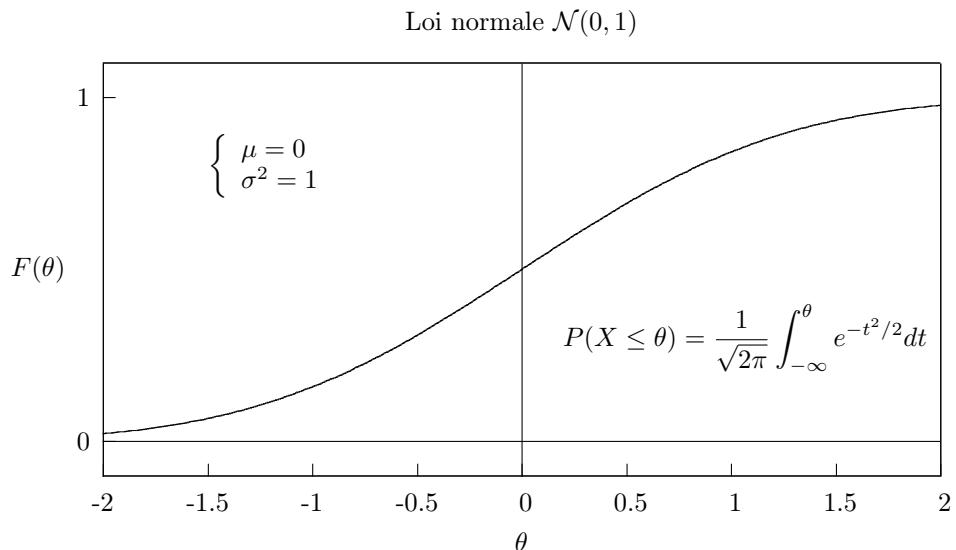


FIGURE 66 – Exemple d'utilisation du terminal L^AT_EX

8.4 Gnuplot et L^AT_EX

Il est possible de créer des figures comportant des commandes et des formules en L^AT_EX à condition d'utiliser le périphérique de sortie *latex* de **gnuplot**. Ce périphérique génère du code qui peut être placé dans un environnement *figure* à l'intérieur d'un document L^AT_EX.

Voici un exemple d'instructions **gnuplot** permettant de créer la figure 66.

```
set terminal latex
set output 'figLatex.tex'

set xlabel '$\theta$'
set ylabel '$F(\theta)$'
set label '$\left\{ \begin{array}{l} \mu=0 \\ \sigma^2=1 \end{array} \right.$' \
at -1.5,0.8
set label '$P(X \leq \theta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\theta} e^{-t^2/2} dt$' \
at 0.2,0.3
set title 'Loi normale $\mathcal{N}(0,1)$'

set xrange [-2:2]
set zeroaxis

plot norm(x) notitle
```

L'instruction *set terminal* sélectionne le périphérique de sortie *latex* et l'instruction *set output* indique le nom du fichier dans lequel le graphique sera généré. Ce fichier comportera le code L^AT_EX nécessaire pour créer ce graphique.

Les commandes *set label* et *set title* peuvent comporter des formules dans le mode mathématique de L^AT_EX. La figure peut ensuite être insérée dans un document L^AT_EX avec les macros suivantes :

```
\begin{figure}
\input{figLatex.tex}
\caption{Légende de la figure}
\end{figure}
```

Remarque : le texte pour les options *label*, *title*, etc. doit être placé entre des guillemets simples, sinon il faut utiliser des contre-obliques doubles pour que les macros L^AT_EX soient transmises correctement.

Index

- == (opérateur), 90
- != (opérateur), 90
- above (mot-clé), 68
- above (option key), 32
- abs (fonction mathématique), 8, 9
- acos (fonction mathématique), 9
- acosh (fonction mathématique), 9
- acsplines (option smooth), 63
- add (option xtics), 23
- all (mot-clé), 16
- angles (option), 7, 14
- append (option print), 19
- arc (option circle), 50
- arg (fonction mathématique), 8, 9
- array (mot-clé), 56, 58
- arrow
 - arrowstyle, 40
 - as, 40
 - back, 40
 - backhead, 40
 - empty, 40
 - filled, 40
 - front, 40
 - head, 40
 - heads, 40
 - linestyle, 40
 - linetype, 40
 - linewidth, 40
 - nofilled, 40
 - nohead, 40
 - size, 40
- arrow (option), 39, 40
- arrowstyle (option arrow), 40
- as (option arrow), 40
- ASCII, 54
- asin (fonction mathématique), 9
- asinh (fonction mathématique), 9
- aspect ratio, 34
- at (option circle), 50
- at (option key), 32
- at (option pm3d), 80
- atan (fonction mathématique), 9
- atan2 (fonction mathématique), 9
- atanh (fonction mathématique), 9
- autoscale
 - fix, 20
- fixmax, 20
- fixmin, 20
- max, 20
- min, 20
- autoscale (option), 20
- axes (option), 7
- axis (mot-clé), 49
- axis (option tics), 22
- back (option arrow), 40
- back (option border), 39
- back (option object), 48
- backhead (option arrow), 40
- bars
 - fullwidth, 79
 - large, 79
 - small, 79
- bars (option), 79
- base (option contour), 43
- begin (argument *flush*), 81
- behind (option object), 48
- below (mot-clé), 68
- below (option key), 32
- besj0 (fonction mathématique), 9
- besj1 (fonction mathématique), 9
- besy0 (fonction mathématique), 9
- besy1 (fonction mathématique), 9
- bezier (option smooth), 63
- big (option endian), 59
- binary
 - center, 60
 - dx, 55, 60
 - dy, 55, 60
 - dz, 60
 - flipx, 60
 - flipy, 60
 - flipz, 60
 - origin, 60, 61
 - perpendicular, 61
 - rotate, 61
 - scan, 60
 - transpose, 60
- binary (mot-clé), 55, 56, 59
- binary (option plot), 86
- bmargin (option), 34
- border
 - back, 39

front, 39
 linestyle, 39
 linetype, 39
 linewidth, 39
 border (mot-clé), 69
 border (option colorbox), 52
 border (option tics), 22
 border (option), 12, 38, 39
 both (option contour), 43
 bottom (option key), 32
 bottom (option timestamp), 15
 box (option dgrid3d), 48
 box (option key), 32
 boxerrorbars (style), 35
 boxes (style), 35, 65, 72, 73
 boxwidth (option), 35, 73
 boxxyerrorbars (style), 78
 by (option tics), 22

 c1 (argument *corners2color*), 82
 c2 (argument *corners2color*), 82
 c3 (argument *corners2color*), 82
 c4 (argument *corners2color*), 82
 call (commande), 16, 90, 91
 candlesticks (style), 35, 65
 canevas, 34
 cartesian (option mapping), 27
 cauchy (option dgrid3d), 48
 cb, 20, 21
 cblabel (option), 53
 cbrange (option), 53, 58
 cbtics (option), 52
 cd (commande), 17, 88
 ceil (fonction mathématique), 9
 center (argument *flush*), 81
 center (option binary), 60
 center (option circle), 50
 center (option key), 32
 cercles, 49
 character (mot-clé), 8, 28, 29, 39
 circle
 arc, 50
 at, 50
 center, 50
 radius, 50
 size, 50
 circles (style), 70
 clabel (option), 30
 clear (commande), 88
 clear (fonction), 88

 clip
 one, 46
 points, 46
 two, 48
 clip (option), 46
 clip1in (option pm3d), 82
 clip4in (option pm3d), 82
 closed (mot-clé), 68
 clustered, 75
 CMY (modèle de couleurs), 99
 cntrparam
 levels, 43
 cntrparam (option), 43
 color (option palette), 99
 colorbox
 border, 52
 default, 52
 horizontal, 52
 origin, 52
 size, 52
 user, 52
 vertical, 52
 colorbox (option), 52
 colormames (option palette), 98
 column (fonction), 91
 columnsfirst (option multiplot), 41
 columnstacked, 75
 contour
 base, 43
 both, 43
 surface, 43
 contour (option), 30, 42, 43
 corners2color
 c1, 82
 c2, 82
 c3, 82
 c4, 82
 geomean, 82
 max, 82
 mean, 82
 median, 82
 min, 82
 corners2color (option pm3d), 81
 cos (fonction mathématique), 9
 cosh (fonction mathématique), 9
 cp1250 (mot-clé), 14
 cp437 (mot-clé), 14
 cp850 (mot-clé), 14
 cp852 (mot-clé), 14
 csplines (option smooth), 63

csv, 54
 cumulative (option smooth), 63
 cylindrical (option mapping), 27
 datafile
 separator, 54
 datafile (option), 54, 59
 decimalsign (option), 14, 15
 default (mot-clé), 14, 36
 default (option colorbox), 52
 default (option endian), 59
 defined (option palette), 102
 degrees (mot-clé), 14
 depthorder (option pm3d), 81
 dgrid3d
 box, 48
 cauchy, 48
 exp, 48
 gauss, 48
 hann, 48
 qnorm, 48
 splines, 48
 dgrid3d (option), 48
 dots (style), 65, 71
 downwards (option multiplot), 42
 dummy (option), 25
 dx (option binary), 55, 60
 dy (option binary), 55, 60
 dz (option binary), 60
 ellipses, 50
 empty (option arrow), 40
 encoding (option), 12, 14
 end (argument *flush*), 81
 endian
 big, 59
 default, 59
 little, 59
 swap, 59
 endian (mot-clé), 59
 enhanced (option term postscript), 18
 eps (option term postscript), 18
 eq (opérateur), 90
 equal, 25
 erf (fonction mathématique), 9
 erfc (fonction mathématique), 9
 errorvariables (option fit), 88
 every (option plot), 54, 86
 exit, 92
 exit (commande), 4, 91
 exp (fonction mathématique), 9
 exp (option dgrid3d), 48
 file (option palette), 102
 filetype (mot-clé), 56
 fill (option style), 75
 filled (option arrow), 40
 filledcurves (style), 65, 67
 financebars (style), 65
 first (mot-clé), 8, 28, 39
 fit
 errorvariables, 88
 using, 86, 87
 fit (commande), 84, 86–88
 fit (fonction), 83
 fit (option), 87
 fit.log, 87
 fit2rgbformulae (option palette), 98
 FIT_CONVERGED (variable), 87
 FIT_LIMIT (variable), 87
 FIT_MAXITER (variable), 87
 fix (option autoscale), 20
 fixmax (option autoscale), 20
 fixmin (option autoscale), 20
 flipx (option binary), 60
 flipy (option binary), 60
 flipz (option binary), 60
 float, 56
 float (mot-clé), 99
 floats, 55
 floor (fonction mathématique), 9
 flush
 begin, 81
 center, 81
 end, 81
 flush (option pm3d), 81
 font (option timestamp), 15
 font (option xlabel), 29
 fontpath (option), 13, 16
 format (mot-clé), 56, 57
 format (option), 23, 97
 frequency (option smooth), 61
 from (option rectangle), 49
 front (option arrow), 40
 front (option border), 39
 front (option object), 48
 fsteps (style), 65, 72
 fullwidth (option bars), 79
 functions (option palette), 101
 functions (option save), 90

functions (option), 16
 gamma (fonction mathématique), 9
 gamma (option palette), 100
 gauss (option dgrid3d), 48
 geomean (argument *corners2color*), 82
 gprintf (fonction), 97
 GPVAL_ (préfixe), 10
 gradient (option palette), 98
 graph (mot-clé), 8, 28, 39, 49
 gray (option palette), 99
 gray (variable), 101
 grid
 linecolor, 37
 linestyle, 37
 linetype, 37
 linewidth, 37
 mx2tics, 37
 mxtics, 37
 my2tics, 37
 mytics, 37
 mztics, 37
 nomx2tics, 37
 nomxtics, 37
 nomy2tics, 37
 nomytics, 37
 nomztics, 37
 noxtics, 36
 noytics, 36
 noztics, 36
 xtics, 36
 ytics, 36
 ztics, 36
 grid (option), 36
 hann (option dgrid3d), 48
 head (option arrow), 40
 heads (option arrow), 40
 help (commande), 7, 18
 hidden3d
 trianglepattern, 46
 hidden3d (option pm3d), 82
 hidden3d (option), 45, 46, 81, 82
 histeps (style), 65, 72
 histograms (style), 35, 65, 72–74
 history (commande), 89
 historysize (option), 14
 horizontal (option colorbox), 52
 horizontal (option key), 32
 HSV (modèle de couleurs), 99
 ibeta (fonction mathématique), 9
 if (commande), 90
 if (fonction), 90
 igamma (fonction mathématique), 9
 imag (fonction mathématique), 9
 image (style), 58, 66, 83
 impulses (style), 65
 in (option tics), 22
 index (option plot), 54
 inside (option key), 32
 int, 55, 56
 int (fonction mathématique), 9
 int (mot-clé), 99
 interpolate (option pm3d), 81
 inverf (fonction mathématique), 9
 invnorm (fonction mathématique), 9
 iso_8859_1 (mot-clé), 14
 iso_8859_15 (mot-clé), 14
 iso_8859_2 (mot-clé), 14
 isocourbes, 44, 79
 isosamples (option), 44, 45, 81
 kdensity (option smooth), 63
 key
 above, 32
 at, 32
 below, 32
 bottom, 32
 box, 32
 center, 32
 horizontal, 32
 inside, 32
 left, 32
 linestyle, 32
 linetype, 32
 linewidth, 32
 outside, 32
 over, 32
 reverse, 32
 right, 32
 title, 32, 33
 top, 32
 under, 32
 vertical, 32
 key (option), 31
 koi8r (mot-clé), 14
 koi8u (mot-clé), 14
 label (option), 29, 30, 104
 labels (style), 65, 66

lambertw (fonction mathématique), 9
 LANG, 15
 large (option bars), 79
 latex (terminal), 103
 layout (option multiplot), 41, 42
 lc (option), 5, 35, 98
 LC_ALL, 15
 LC_TIME, 15
 left (option key), 32
 levels (option cntrparam), 43
 lgamma (fonction mathématique), 9
 linecolor (option grid), 37
 linecolor (option plot), 68
 linecolor (option), 5, 35, 98
 lines (style), 33, 44, 54, 65, 66, 79
 linespoints (style), 65, 70
 linestyle (option arrow), 40
 linestyle (option border), 39
 linestyle (option grid), 37
 linestyle (option key), 32
 linestyle (option zeroaxis), 21
 linestyle (option), 35
 linetype (option arrow), 40
 linetype (option border), 39
 linetype (option grid), 37
 linetype (option key), 32
 linetype (option zeroaxis), 21
 linetype (option), 5, 35, 98
 linewidth (option arrow), 40
 linewidth (option border), 39
 linewidth (option grid), 37
 linewidth (option key), 32
 linewidth (option zeroaxis), 21
 linewidth (option), 5, 35
 little (option endian), 59
 lmargin (option), 34
 load (commande), 16, 90, 91
 load (fonction), 90
 loadpath (option), 13, 16
 locale (mot-clé), 14
 locale (option), 15
 log (fonction mathématique), 9
 log10 (fonction mathématique), 9
 logscale (option), 21
 long (mot-clé), 16
 ls (option), 35
 lt (option), 5, 35, 98
 lw (option), 5, 35
 macro, 11
 macros (option), 14
 map, 25
 map (option pm3d), 83
 mapping
 cartesian, 27
 cylindrical, 27
 spherical, 27
 mapping (option), 27
 margin (option), 34
 matrix (mot-clé), 55
 max (argument *corners2color*), 82
 max (option autoscale), 20
 mcbtics (option), 53
 mean (argument *corners2color*), 82
 median (argument *corners2color*), 82
 min (argument *corners2color*), 82
 min (option autoscale), 20
 mirror (option tics), 22
 model (option palette), 99
 mouse (option), 14
 multiplot, 11
 columnsfirst, 41
 downwards, 42
 layout, 41, 42
 offset, 42
 rowsfirst, 41
 scale, 42
 title, 42
 upwards, 42
 multiplot (option), 41
 mx2tics (option grid), 37
 mx2tics (option), 22
 mxtics (option grid), 37
 mxtics (option), 22
 my2tics (option grid), 37
 my2tics (option), 22
 mytics (option grid), 37
 mytics (option), 22
 mz2tics (option grid), 37
 mz2tics (option), 22
 NaN, 10
 ne (opérateur), 90
 nofilled (option arrow), 40
 nohead (option arrow), 40
 nomirror (option tics), 22
 nomx2tics (option grid), 37
 nomxtics (option grid), 37
 nomy2tics (option grid), 37
 nomytics (option grid), 37

nomztics (option grid), 37
 noratio (mot-clé), 34
 norm (fonction mathématique), 9
 norotate (option tics), 22
 nosquare (mot-clé), 34
 notitle (option plot), 33
 noxtics (option grid), 36
 noytics (option grid), 36
 noztics (option grid), 36

 object
 back, 48
 behind, 48
 front, 48
 object (option), 48
 offset (option multiplot), 42
 offset (option tics), 22
 offset (option timestamp), 15
 offset (option title), 28
 offset (option xlabel), 29
 offsets (option), 28
 one (option clip), 46
 origin (option binary), 60, 61
 origin (option colorbox), 52
 origin (option), 19, 88
 out (option tics), 22
 output (option), 13, 17
 outside (option key), 32
 over (option key), 32

 palette
 color, 99
 colormames, 98
 defined, 102
 file, 102
 fit2rgbformulae, 98
 functions, 101
 gamma, 100
 gradient, 98
 gray, 99
 model, 99
 palette, 98
 rgbformulae, 98, 101
 using, 102
 palette (option palette), 98
 palette (option), 52, 97, 98, 101
 parametric (option), 26, 27
 pattern (mot-clé), 68
 pause (commande), 91
 perpendicular (option binary), 61

 pi (variable), 10
 plot
 binary, 86
 every, 54, 86
 index, 54
 linecolor, 68
 notitle, 33
 smooth, 61
 title, 33
 using, 56, 74, 78, 86
 with, 65, 66
 plot (commande), 4, 5, 7, 11, 16, 19,
 20, 33, 36, 41, 55, 58–61, 65,
 67, 68, 71, 74, 86, 90, 98
 plot (fonction), 54
 plot (option), 16
 pm3d
 at, 80
 clip1in, 82
 clip4in, 82
 corners2color, 81
 depthorder, 81
 flush, 81
 hidden3d, 82
 interpolate, 81
 map, 83
 scansautomatic, 81
 scansbackward, 81
 scansforward, 81
 pm3d (option), 80, 82
 pm3d (style), 52, 66, 79, 80, 97
 png (terminal), 16, 18
 pointinterval (option), 70
 points (option clip), 46
 points (style), 65, 66, 70
 pointsize (option), 35, 70
 pointtype (option), 35
 polar (option), 26
 polygons, 51
 postscript
 enhanced, 18
 eps, 18
 postscript (terminal), 16–18, 69
 print
 append, 19
 print (commande), 8, 18, 19
 print (option), 18
 printf, 23
 projection orthogonale, 24
 ps (option), 35

pt (option), 35
 pwd (commande), 17, 88

 qnorm (option dgrid3d), 48
 quit (commande), 4, 91

 radians (mot-clé), 14
 radius (option circle), 50
 rand (fonction mathématique), 9
 ratio (mot-clé), 34, 35
 real (fonction mathématique), 9
 record (mot-clé), 56, 58
 rectangle
 from, 49
 rto, 49
 to, 49
 rectangles, 48
 replot (commande), 4, 5
 reread (commande), 91
 reset (commande), 13
 reverse (option key), 32
 RGB (modèle de couleurs), 99
 rgba (style), 83
 rgbfomulae (mot-clé), 101
 rgbfomulae (option palette), 98, 101
 rgbiimage (style), 58, 66, 83
 right (option key), 32
 rmargin (option), 34
 rotate (mot-clé), 61
 rotate (option binary), 61
 rotate (option tics), 22
 rotate (option timestamp), 15
 rotate (option xlabel), 29
 rowsfirst (option multiplot), 41
 rowstacked, 75
 rrange (option), 28
 rto (option rectangle), 49

 samples (option), 44, 45, 81
 save
 functions, 90
 set, 90
 terminal, 90
 variables, 90
 save (commande), 90
 save (fonction), 90
 sbezier (option smooth), 63
 scale (option multiplot), 42
 scan (option binary), 60
 scansautomatic (option pm3d), 81

 scansbackward (option pm3d), 81
 scansforward (option pm3d), 81
 scatterplot, 71
 screen (mot-clé), 8, 19, 28, 39, 41, 49
 second (mot-clé), 8, 28, 39
 separator (option datafile), 54
 set (commande), 5, 12, 13, 82, 87
 set (option save), 90
 sgn (fonction mathématique), 9
 shell (commande), 91, 92
 short, 56
 show (commande), 12, 16, 87
 sin (fonction mathématique), 9
 sinh (fonction mathématique), 9
 size (option arrow), 40
 size (option circle), 50
 size (option colorbox), 52
 size (option), 34, 35, 88
 skip (mot-clé), 58
 small (option bars), 79
 smooth
 acsplines, 63
 bezier, 63
 csplines, 63
 cumulative, 63
 frequency, 61
 kdensity, 63
 sbezier, 63
 unique, 61, 63
 smooth (option plot), 61
 solid (mot-clé), 68
 spherical (option mapping), 27
 splines (option dgrid3d), 48
 splot (commande), 4, 5, 11, 16, 19, 20,
 38, 43, 55, 59–61, 71, 90
 sprintf (fonction), 97
 sqrt (fonction mathématique), 9
 square (mot-clé), 34
 stderr, 18
 stdout, 18
 steps (style), 65, 72
 strftime (fonction), 95
 strlen (fonction), 96
 strptime (fonction), 96
 strstr (fonction), 96
 style
 fill, 75
 style (option), 75
 substr (fonction), 96
 surface (option contour), 43

surface (option), 43
 swap (option endian), 59
 system (commande), 91, 92
 system (fonction), 96

 table (option), 19
 tag, 39
 tan (fonction mathématique), 9
 tanh (fonction mathématique), 9
 tc (option), 98
 Terminal
 latex, 103
 png, 16, 18
 postscript, 16–18, 69
 x11, 17
 terminal (option save), 90
 terminal (option), 13, 17
 termoption (option), 18
 test (commande), 36, 93
 textcolor (option xlabel), 29
 textcolor (option), 98
 tics
 axis, 22
 border, 22
 by, 22
 in, 22
 mirror, 22
 nomirror, 22
 norotate, 22
 offset, 22
 out, 22
 rotate, 22
 tics (option), 21, 22
 timefmt (option), 14, 24
 timestamp
 bottom, 15
 font, 15
 offset, 15
 rotate, 15
 top, 15
 timestamp (option), 15
 title
 offset, 28
 title (option key), 32, 33
 title (option multiplot), 42
 title (option plot), 33
 title (option), 28, 29, 33, 104
 tm_hour (fonction), 95
 tm_mday (fonction), 95
 tm_min (fonction), 95

 tm_mon (fonction), 95
 tm_sec (fonction), 95
 tm_wday (fonction), 95
 tm_yday (fonction), 95
 tm_year (fonction), 95
 tmargin (option), 34
 to (option rectangle), 49
 top (option key), 32
 top (option timestamp), 15
 trange (option), 28
 transpose (option binary), 60
 trianglepattern (option hidden3d), 46
 tsv, 54
 two (option clip), 48

 under (option key), 32
 unique (option smooth), 61, 63
 unset (commande), 12
 unset (option), 12
 unset. (commande), 87
 update (commande), 88
 upwards (option multiplot), 42
 urange (option), 28
 user (option colorbox), 52
 using (option fit), 86, 87
 using (option palette), 102
 using (option plot), 56, 74, 78, 86
 using (option), 5, 6, 91

 variables (option save), 90
 variables (option), 16
 vectors (style), 66, 71
 version (option), 16
 vertical (option colorbox), 52
 vertical (option key), 32
 view (option), 24
 vrangle (option), 28

 with (option plot), 65, 66
 word (fonction), 96
 words (fonction), 96

 x, 20, 21, 24
 x (mot-clé), 7
 x11 (terminal), 17
 x2, 20, 21, 24
 x2 (mot-clé), 7
 x2data (option), 24
 x2dtics (option), 22
 x2label (option), 29

x2mtics (option), 22
x2range (option), 28
x2tics (option), 21, 23
x2zeroaxis (option), 21
xdata (option), 24
xdtics (option), 22
xerrorbars (style), 78, 79
xerrorlines (style), 79
 xlabel
 font, 29
 offset, 29
 rotate, 29
 textcolor, 29
 xlabel (option), 29
 xmtics (option), 22
 xrange (option), 28
 xtics
 add, 23
 xtics (option grid), 36
 xtics (option), 15, 21, 22, 52
 xy, 20
 xyerrorbars (style), 78, 79
 xyerrorlines (style), 79
 xyplane (option), 20
 XYZ (modèle de couleurs), 99
 xzeroaxis (option), 21

 y, 20, 21, 24
 y (mot-clé), 7
 y2, 20, 21, 24
 y2 (mot-clé), 7
 y2data (option), 24
 y2dtics (option), 22
 y2label (option), 29
 y2mtics (option), 22
 y2range (option), 28
 y2tics (option), 21, 23
 y2zeroaxis (option), 21
 ydata (option), 24
 ydtics (option), 22
 yerrorbars (style), 78, 79
 yerrorlines (style), 79
 YIQ (modèle de couleurs), 99
 ylabel (option), 29
 ymtics (option), 22
 yrangle (option), 28, 40
 ytics (option grid), 36
 ytics (option), 15, 21, 22
 yzeroaxis (option), 21