

Aide - mémoire gnuplot 4.0

Nicolas KIELBASIEWICZ

20 juin 2008

L'objet de cet aide-mémoire est de présenter les commandes de base pour faire rapidement de très jolis graphiques et courbes à l'aide du logiciel libre GNUPLOT , dont la page web officielle est :

<http://www.gnuplot.info/>

Pourquoi GNUPLOT ? Les connaisseurs diront que ce n'est la le seul logiciel libre dans le domaine (il y a par exemple XMGR GRACE). Néanmoins, distribué par licence G.N.U., il est probablement le plus courant.

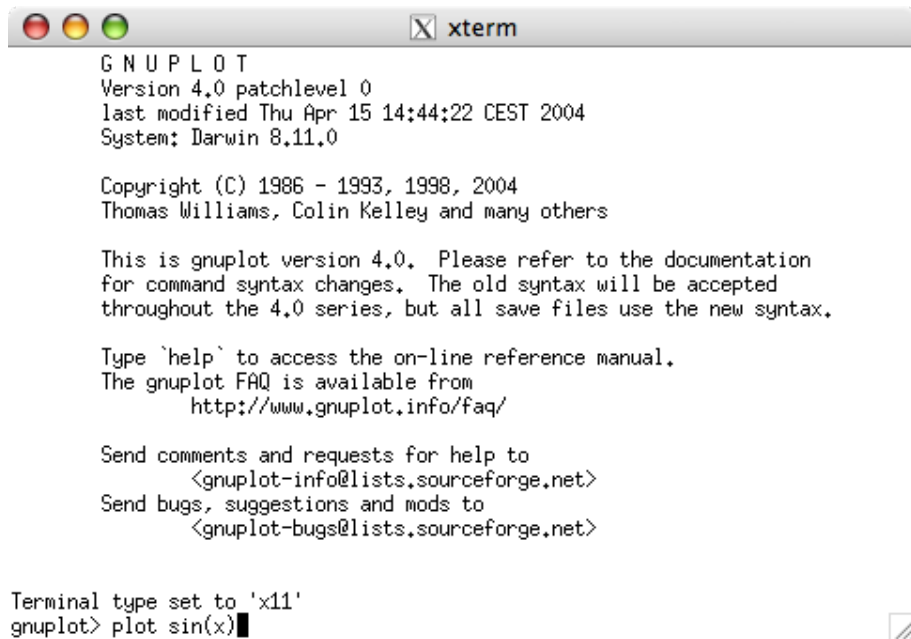
Table des matières

1	Les différents moyens d'utiliser GNUPLOT	2
1.1	A la volée	2
1.2	En ligne de commande	2
1.3	Avec des scripts	2
1.4	Utiliser l'aide	2
2	Courbes 2D : plot	3
2.1	Fonctions et courbes paramétrées	3
2.2	Tracer des données à partir d'un fichier	3
2.2.1	L'attribut using	4
2.2.2	L'attribut every	4
2.3	Modifier l'apparence	5
2.3.1	Spécifier l'aspect du tracé : l'attribut with	5
2.3.2	Définir les axes : xrange , yrange	6
2.3.3	Donner un titre et une légende : title	6
2.3.4	Les commandes set , unset et show	6
3	Courbes 3D : splot	7
3.1	Fonctions et surfaces paramétrées	7
3.2	Tracer des données à partir d'un fichier	8
3.3	Modifier l'apparence	8
3.3.1	Maillages	8
3.3.2	Surfaces de couleur	8
4	Fonctionnalités avancées	9
4.1	Enregistrer des courbes : term	9
4.2	Faire des animations : reread	10
4.3	Simplifier la syntaxe : les abréviations	10
4.4	Plusieurs graphes dans une même fenêtre : multiplot	10

1 Les différents moyens d'utiliser GNUPLOT

1.1 A la volée

Dans un terminal, GNUPLOT se lance avec la commande `gnuplot`. On peut alors taper les instructions dans la console qui apparaît.



```
GNUPLOT
Version 4.0 patchlevel 0
last modified Thu Apr 15 14:44:22 CEST 2004
System: Darwin 8.11.0

Copyright (C) 1986 - 1993, 1998, 2004
Thomas Williams, Colin Kelley and many others

This is gnuplot version 4.0. Please refer to the documentation
for command syntax changes. The old syntax will be accepted
throughout the 4.0 series, but all save files use the new syntax.

Type `help` to access the on-line reference manual.
The gnuplot FAQ is available from
  http://www.gnuplot.info/faq/

Send comments and requests for help to
  <gnuplot-info@lists.sourceforge.net>
Send bugs, suggestions and mods to
  <gnuplot-bugs@lists.sourceforge.net>

Terminal type set to 'x11'
gnuplot> plot sin(x) █
```

FIG. 1: La console de GNUPLOT

1.2 En ligne de commande

Une autre solution consiste à passer les instructions sur l'entrée standard à l'aide d'une commande du type `echo 'plot sin(x)' | gnuplot`.

1.3 Avec des scripts

Une troisième solution consiste à écrire les instructions dans un fichier qui sera un script GNUPLOT puis à charger ce script. Bien que l'extension n'ait aucune importance, je suggère de donner l'extension `.gnuplot`, comme le fait L^AT_EX quand on utilise le package graphique TikZ / PGF. On exécute alors la commande `gnuplot monscript.gnuplot` dans le terminal ou `load "script.gnuplot"` dans la console de GNUPLOT. Je vous conseille personnellement de lancer la commande `gnuplot -persist monscript.gnuplot`. Cela empêche la fenêtre dans laquelle la courbe sera tracée de disparaître à la fin de l'exécution du script.

1.4 Utiliser l'aide

Il suffit de taper la commande `help`.

2 Courbes 2D : plot

La commande dédiée au tracé de courbes 2D est `plot`. Nous allons voir ici les usages les plus courants.

2.1 Fonctions et courbes paramétrées

Pour tracer une fonction de la forme $y=f(x)$ (la variable muette par défaut est x) :

```
plot sin(x)
```

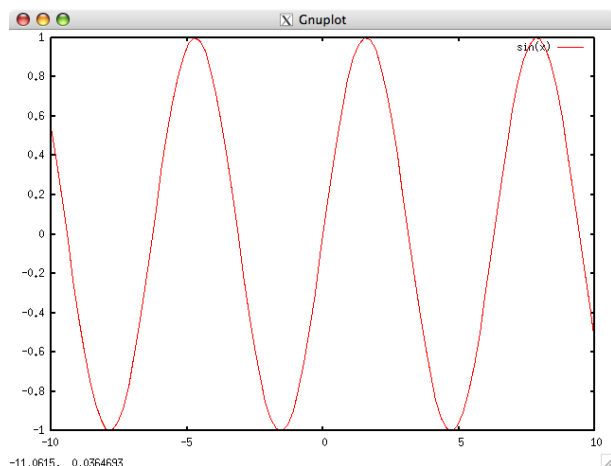


FIG. 2: Tracé d'une fonction avec `plot`

On peut également tracer des courbes paramétrées (la variable muette par défaut est t) :

```
set parametric  
plot 2*t, sin(t)
```

Les fonctions disponibles

- Les fonctions trigonométriques : `cos`, `sin`, `tan`, `acos`, `asin`, `atan`
- Les fonctions hyperboliques : `cosh`, `sinh`, `tanh`, `acosh`, `asinh`, `atanh`
- Les fonctions usuelles : `log`, `exp`, `abs`, `log10`, `sqrt`
- ...

2.2 Tracer des données à partir d'un fichier

On considère un fichier `file.txt` contenant 2 colonnes ou plus de valeurs. Il peut également y avoir des blocs de ce type, chacun des blocs étant séparé par une ligne vierge du bloc précédent.

La commande par défaut pour un fichier contenant 2 colonnes est :

```
plot "file.txt"
```

Le comportement par défaut est comme suit :

- La première colonne servira d'abscisse
- La deuxième colonne servira d'ordonnée

- Si plusieurs blocs sont présents, chaque bloc constituera une courbe et tous les blocs seront tracés avec le même style.

2.2.1 L'attribut `using`

Que se passe-t-il si le fichier comporte plusieurs colonnes ? Comment spécifier les colonnes à tracer ? La réponse est donnée par l'attribut `using` :

```
plot "file.txt" using 2:3
```

Dans cet exemple, la deuxième colonne servira d'abscisse, la 3ème servira d'ordonnée. On peut également effectuer des opérations élémentaires sur les colonnes :

```
plot "file.txt" using ($2+log($1)):3
```

2.2.2 L'attribut `every`

Afin de gagner en rapidité, on peut souhaiter limiter le nombre de points affichés par courbe. De même, on peut également manipuler les blocs. Il faut pour cela utiliser l'attribut `every`, qui ne fonctionne bien évidemment que lorsqu'on manipule des fichiers de données.

Manipuler les points Pour afficher 1 point sur 5 pour chaque courbe :

```
plot "file.txt" using 1:2 every 5
```

Pour afficher un sous intervalle de points (la numérotation commence à 0) :

```
plot "file.txt" using 1:2 every ::3::15
```

Dans cet exemple, on ne spécifie que le 3ème et le 5ème argument, les autres sont laissés à leur valeur par défaut. Bien qu'il y en ait 6 au total, on n'a pas fait apparaître le 6ème et dernier. La raison est simple. Puisqu'on spécifie le 5ème argument, ajouter le 6ème et lui donner sa valeur par défaut est une lourdeur d'écriture inutile. C'est ce que l'on a fait dans le premier exemple où on ne spécifie que le premier argument : les 5 autres prennent la valeur par défaut.

Manipuler les blocs La manipulation des blocs va concerner les arguments 2, 4 et 6 de l'attribut `every`.

Pour afficher un ensemble de blocs :

```
plot "file.txt" using 1:2 every :::0::5
```

Pour afficher un bloc sur 2 :

```
plot "file.txt" using 1:2 every :2
```

On peut bien entendu combiner toutes ces fonctionnalités en spécifiant une valeur pour chacun des 6 arguments, ce qui fait que l'attribut `every` est une fonctionnalité très puissante.

2.3 Modifier l'apparence

2.3.1 Spécifier l'aspect du tracé : l'attribut `with`

On souhaite pouvoir choisir le style de la courbe que l'on souhaite tracer. Cela sousentend de pouvoir choisir de tracer une ligne ou une série de symboles et de pouvoir spécifier la taille ou épaisseur, le symbole et la couleur. On utilise pour cela l'attribut **with**.

```
plot sin(x) with lines , cos(x) with points
```

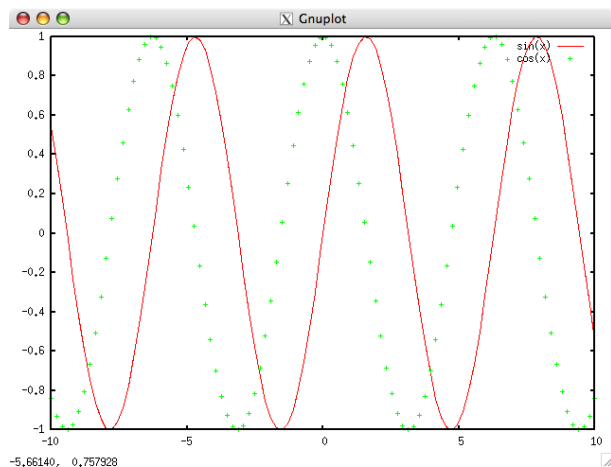


FIG. 3: L'attribut **with**

```
plot sin(x) with lines linetype 2 linewidth 2, \  
cos(x) with points linetype 6 pointtype 5 pointsize 1.5
```

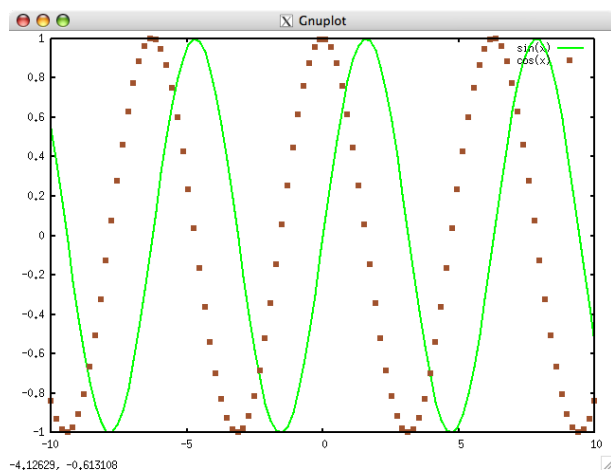


FIG. 4: L'attribut **with**

En ce qui concerne **linewidth** et **pointsize**, la valeur donnée correspond au facteur multiplicatif par rapport à la taille standard. Dans l'exemple précédent, on traçait donc une ligne 2 fois plus épaisse et des symboles 1.5 fois plus grands que la normale.

Quelle est la liste des valeurs possibles pour les attributs **linetype** et **pointtype** ? On peut connaître cette liste en exécutant la commande `test` dans la console de GNUPLOT , qui donne tout un tas d'informations sur les possibilités offertes. Voici ce qui concerne **linetype** et **pointtype** :

valeur	couleur	valeur	couleur
-1	ligne noire épaisse	0	ligne pointillée
1	rouge	2	vert
3	bleu	4	magenta
5	cyan	6	marron
7	orange	8	rose

TAB. 1: Les valeurs de **linetype**

valeur	symbole	valeur	symbole
0	.	1	+
2	×	3	*
4	◻	5	■
6	⊙	7	●
8	△	9	▲
10	▽	11	▼
12	◇	13	

TAB. 2: Les valeurs de **pointtype**

L'attribut **with** offre bien davantage de possibilités de tracés, dont voici quelques unes : **linespoints**, **dots**, **errorbars**, **impulses**. Se reporter à l'aide pour davantage d'informations.

2.3.2 Définir les axes : **xrange**, **yrange**

```
set xrange [0:6]
set yrange [-2:2]
plot sin(x)
```

2.3.3 Donner un titre et une légende : **title**

```
set title "titre"; plot sin(x) title "legende"
```

2.3.4 Les commandes **set**, **unset** et **show**

La commande **set** permet de spécifier les diverses options du tracé à venir. Si l'on veut désactiver une option, on utilisera **unset**. Si l'on veut connaître la valeur d'une option, on utilisera la commande **show**.

Si par exemple on ne veut pas afficher la légende, on utilisera la commande **set key off** et **unset key**. On peut également la positionner avec par exemple **set key left**.

3 Courbes 3D : splot

La commande dédiée au tracé de surfaces est `splot`. Nous allons voir ici les quelques différences qu'il y a par rapport aux courbes 2D.

3.1 Fonctions et surfaces paramétrées

Les variables muettes par défaut en mode standard sont x et y . Le tracé d'une surface définie par une fonction $z=f(x,y)$ se fait de la manière suivante :

```
splot cos(0.25*x)*sin(0.5*y)
```

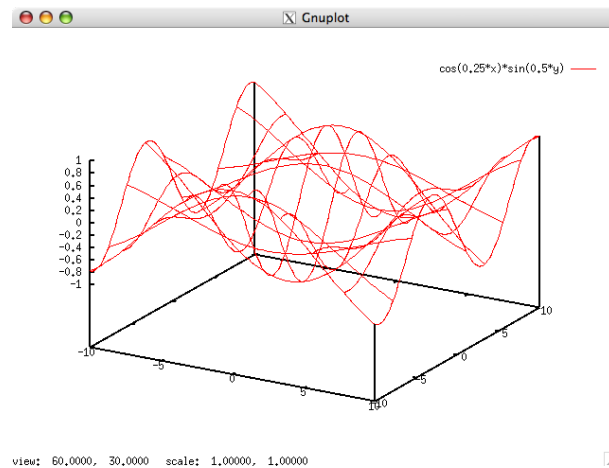


FIG. 5: Tracé de surface avec `splot`

Si l'on veut modifier le nombre de points du maillage, il faut utiliser `samples` et/ou `isosample` :

```
set isosample 101, 101  
splot cos(0.25*x)*sin(0.5*y) with lines linetype 3
```

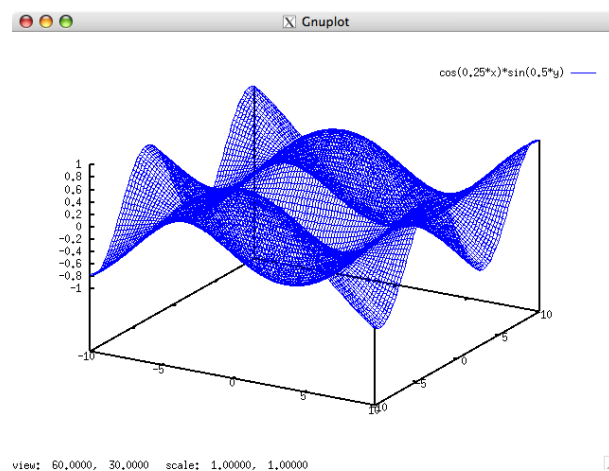


FIG. 6: Modifier le nombre de points du maillage

En mode paramétrique, les variables muettes par défaut sont u et v . Dans ce cas, la commande `splot` prendra 3 arguments.

3.2 Tracer des données à partir d'un fichier

Tout comme en 2D, on peut tracer une surface définie à partir de données issues d'un fichier `file.txt` correspondant aux valeurs d'une fonction à 2 variables sur une grille. Néanmoins, la structure de ces données, toujours fondamentale sur le résultat, est un peu plus complexe. En effet, un tel fichier comprendra au moins trois colonnes de la forme $x_i \ y_j \ z_{ij}$, mais il nous faut ajouter une précaution supplémentaire : une fois terminée la boucle sur j pour un i donné, il faudra passer une ligne. Ainsi, chaque ligne constituera un bloc (au sens de `GNUPLOT`).

La commande utilisée sera :

```
splot "file.txt" with lines
```

A l'affichage, chaque point sera relié à ses voisins dans la grille. Si les blocs ne sont pas séparés, il s'agira de points qui, s'ils sont reliés moyennant l'option adéquate, constitueront les éléments d'une courbe paramétrée 3D.

3.3 Modifier l'apparence

3.3.1 Maillages

Il s'agit du comportement par défaut. Si la surface provient d'une fonction à deux variables, il s'agit du maillage déformé. Si la surface provient de données issues d'un fichier, seuls les points sont tracés.

Les attributs vus dans le cas des courbes 2D restent valables en 3D.

3.3.2 Surfaces de couleur

L'attribut **with** nous offre une nouvelle possibilité pour tracer des surfaces de couleurs, à savoir l'option **pm3d** :

```
set pm3d
splot cos(0.25*x)*sin(0.5*y) with pm3d at s
```

"pm3d at s" signifie que la surface de couleur est sur le tracé du maillage. On peut également choisir de la tracer en bas ("at b") ou en haut ("at t") du graphe. On peut également le définir par défaut en écrivant "set pm3d at s".

Le choix de la palette de couleur est ici par défaut. On a la possibilité de le changer, par exemple pour retrouver la palette *hot* (le choix de Matlab par défaut). Deuxième remarque concernant la légende : dans ce type d'affichage (avec "with pm3d"), elle n'a plus de raison d'être, donc autant l'enlever.

```
unset key
set pm3d at s
set palette rgbformulae 33,13,10
splot cos(0.25*x)*sin(0.5*y) with pm3d
```

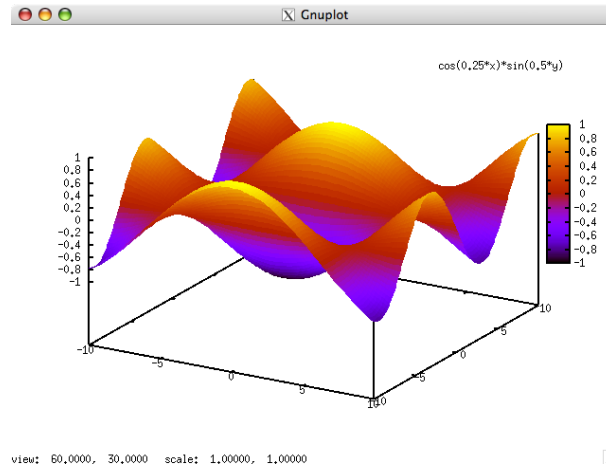



FIG. 7: Surfaces de couleur avec **pm3d**

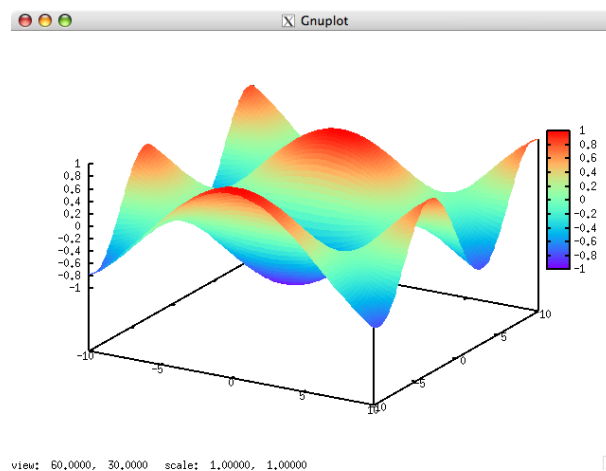


FIG. 8: Choisir la palette de couleur : ici, le style "hot"

4 Fonctionnalités avancées

4.1 Enregistrer des courbes : term

Si l'on tape la commande **show term**, il nous dira que ce paramètre vaut 'x11'. Cela signifie que la sortie standard de GNUPLOT est le terminal X. C'est la raison pour laquelle les courbes s'affichent dans des fenêtres. Pour pouvoir sauvegarder les courbes dans des fichiers, il faut donc changer la sortie standard. Les possibilités offertes dépendent de votre serveur X. Pour les connaître, il suffit de taper la commande **set terminal**. Je parlerai ici de la plus commune, à savoir la sortie au format postscript encapsulé couleur. L'exemple suivant montre comment sauvegarder notre surface dans un fichier 'image.eps' :

```
set terminal postscript eps enhanced color
set output "image.eps"
unset key
set pm3d at s
set palette rgbformulae 33,13,10
splot cos(0.25*x)*sin(0.5*y) with pm3d
```

L'exécution de ce script va donc générer le fichier `image.eps` contenant le tracé de notre courbe. Néanmoins, on ne verra rien s'afficher à l'écran.

Une manière de procéder est la suivante :

```
unset key
set pm3d at s
set palette rgbformulae 33,13,10
splot cos(0.25*x)*sin(0.5*y) with pm3d
set terminal postscript eps enhanced color
set output "image.eps"
replot
```

4.2 Faire des animations : `reread`

GNUPLOT offre la possibilité de définir des variables. Si on ajoute à cela la commande **reread** qui exécute de nouveau le script, alors on voit très vite comment procéder pour effectuer une animation, ce qui peut s'avérer très utile lorsqu'on exécute un code qui simule un problème d'évolution, comme une équation de la chaleur ou une équation des ondes et qu'on veut tracer le profil de la solution calculée au cours du temps. La solution que je préconise est de sauvegarder les fichiers en utilisant la structure de blocs pour séparer les courbes d'un instant à l'autre. Imaginons ici que l'on résout l'équation de la chaleur 1D.

Le script `trace.gnuplot` qui va tracer une courbe va s'écrire comme suit :

```
set xrange [0:1]; set yrange [0:1]
plot "heat1d.txt" using 1:2 every :::a::a with lines linetype 2 linewidth 2
pause 0.01
a=a+1
if(a<b) reread
```

A présent, deux solutions s'offrent à nous, ou bien écrire un second script ou bien taper les instructions qui suivent dans la console de GNUPLOT :

```
a=0
b=100
load "trace.gnuplot"
```

On obtient ainsi une animation à 100 images, dont on peut piloter la vitesse avec la commande **pause**.

4.3 Simplifier la syntaxe : les abréviations

Dans tout ce qui a été vu précédemment, un certain nombre de mots clé, parce qu'il ne peut y avoir d'ambiguïté, peuvent être écrits de manière abrégée.

4.4 Plusieurs graphes dans une même fenêtre : **multiplot**

Voilà le dernier élément que je souhaite vous présenter : la possibilité de décomposer la fenêtre graphique en plusieurs graphes. On va utiliser pour cela l'environnement **multiplot**. L'exemple qui va suivre montre l'utilisation des variables **size** et **origin** pour positionner les différents graphes, en revisitant au passage les divers éléments abordés dans cet aide-mémoire.

attribut	abréviation	attribut	abréviation
using	u	linespoints	lp
with	w	linetype	lt
lines	l	linewidth	lw
points	p	pointtype	pt
title	t	pointsize	ps

TAB. 3: Les abréviations usuelles

```

set multiplot
set size 0.5, 0.5
set origin 0.0, 0.0
plot cos(x) t "cos" w l lt 3
set size 0.5, 0.5
set origin 0.5, 0.0
set xrange [-1:1]; set yrange [-1:1]
unset key
set pm3d at s
set palette rgbformulae 33, 13, 10
set isosample 101, 101
set title "gaussienne"
splot exp(-5*(x*x+y*y)) w pm3d
set size 1.0, 0.5
set origin 0.0, 0.5
unset title
set key left
set xrange [-10:10]; set yrange [-2:2]
plot atan(x) w p pt 4 ps 2 lt 6, sin(x)/x w lp lt 2 lw 2 pt 6
unset multiplot

```

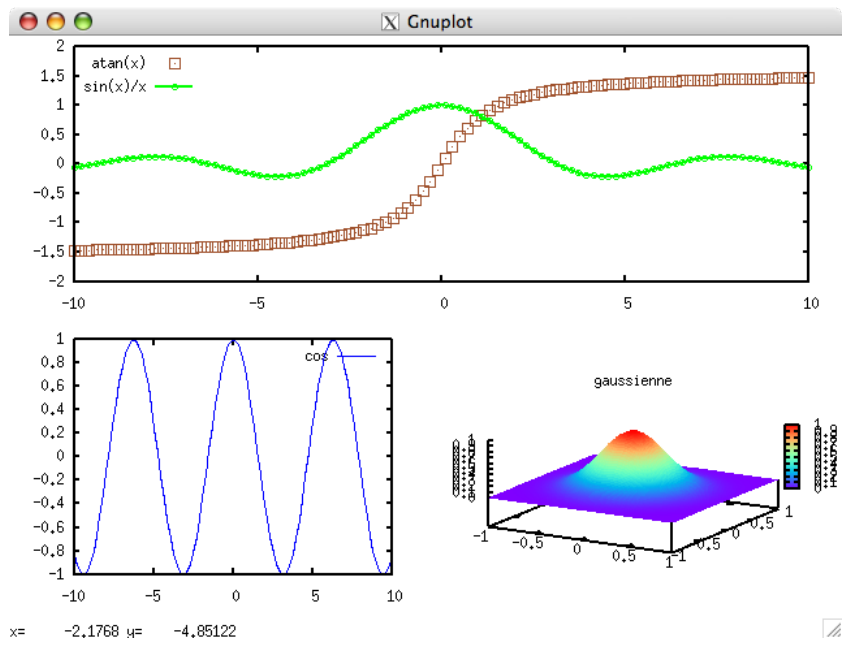


FIG. 9: Exemple de synthèse avec **multiplot**