Chapitre 3

Gnuplot

Le langage C ne permet pas directement de dessiner des courbes et de tracer des plots. Il faut pour cela stocker résultats dans des fichier, et, dans un deuxième temps utiliser un autre logiciel qui va lire ces fichiers et tracer les courbes correspondantes. Il existe de nombreux logiciels qui permettent de s'occuper de ces taches (grâce, xmgrace, gnuplot...). Ce chapitre est dédié au dessin scientifique avec Gnuplot.

3.1 Lancer Gnuplot

Gnuplot se lance depuis un terminal avec la commande :
`>gnuplot`
On peut lancer gnuplot en lui demandant d'exécuter un ensemble d'instructions écrite dans un fichier (par exemple `script.gnu`) en utilisant la commande
`>gnuplot script.gnu`
Pour garder gnuplot ouvert après l'exécution de toutes ces instructions, il est préférable d'utiliser la syntaxe :
`>gnuplot script.gnu -persist`

Lorsque gnuplot est ouvert, on a accès à un nouveau prompt `gnuplot>` dans lequel on peut taper des commandes interprétées par gnuplot. On peut par exemple lancer l'exécution des instructions du même fichier en tapant :
`gnuplot>load "script.gnu"`
Sinon, la commande la plus simple pour faire un plot $y(x)$ lorsque x et y ont été enregistrés en deux colonnes dans une fichier `fich.dat` est :

```
gnuplot>plot "fich.dat"
```

Mais Gnuplot de très nombreuses possibilités dont voici un petit résumé...

3.2 Options des graphes

Avant de tracer un graphe, il est possible de régler un certain nombre de de propriétés par défaut. On règle toutes les valeurs des paramètre en utilisant la commande `set` suivie du nom du paramètre et de sa valeur. On peut également revenir aux réglages initiaux grâce à `unset`. Voici une liste des propriétés les plus fréquentes que l'on peut être amené à régler :

`set size ratio 1`	spécifie le ratio x/y pour la taille du graphe (`set size square`)
`set autoscale`	met en marche l'autoscale
`set autoscale fix`	force les échelles à s'ajuster exactement
`set xrange [0:20]`	fixe l'échelle en x et annule l'autoscale (idem pour y et z)
`set key 2,45`	affiche la légende aux coordonnées (x=2,y=45)
`set title "Mon Titre"`	spécifie la légende du graphe
`set notitle`	supprime la légende (équivalent à `unset key`)
`set xlabel "titre x"`	fixe le nom de l'axe x (idem pour y et z)
`set logscale x`	fixe une échelle log pour l'axe x (idem pour y, z, xy, xy, yz, xyz...)
`set logscale x e`	fixe l'échelle log en base "e" (idem pour 10 ou toute autre base)
`set label "texte" at 3,5`	affiche du texte aux coordonnées (x=3,y=5)
`unset label`	efface tous les textes affichés
`set lmargin at screen 0.3`	règle la marge à gauche à 30% de la taille totale du graphe
`set rmargin at screen 0.3`	idem pour la marge à droite
`set tmargin at screen 0.3`	idem pour la marge en haut
`set bmargin at screen 0.3`	idem pour la marge en bas
`set mapping cartesian`	met en coordonnées cartésiennes (idem pour `spherical` et `cylindrical`)

Certaines de ces options peuvent également être spécifiées directement avec la commande `plot` (voir ci-après).

3.3 Les plot 2D : $y = f(x)$

On peut utiliser la fonction `plot` pour tracer des graphes simples 2D, soit à partir d'une fonction définie avec gnuplot, soit à partir d'un fichier de données.

3.3.1 Options générales

La fonction `plot` admet beaucoup d'options donc certaines sont décrites ici :

`title "mon titre"`	spécifie un titre pour la légende
`with dots`	trace avec des points
`with points`	trace avec des symbole (plus, croix...)
`with lines`	trace avec des lignes
`with linespoints`	trace avec des points et des lignes
`with impulses`	trace avec des droites verticales
`with errorbars`	trace avec des points et des barres d'erreur
`linewidth 1.5`	épaissit le trait d'un facteur multiplicatif 1.5
`pointtype 3`	utilise le symbole 3 (entre 0 et 13) quand `points` est specifié
`pointsize 1.6`	grossit le symbole d'un facteur multiplicatif 1.6
`linecolor rgb "couleur"`	utilise une couleur prédéfinie (`show colornames` liste des couleurs prédéfinies)
`linetype 3`	utilise le types prédéfini 3 (entre -1 et 8) qui regroupe plusieurs propriétés

3.3.2 Tracer une fonction

`set isosample 100`	fixe un nombre de points n=100 pour la variable x
`f(x)=sin(x)`	définit une fonction f
`plot f(x)`	trace la fonction f(x)

3.3.3 Tracer des données

La commande de base pour représenter des données contenues dans un fichier est : `plot "fichier"`

Formattage du fichier

De manière générale, les lignes du fichier de données qui commencent par le symbole `#` ne sont pas lues.

D'autre part, pour que gnuplot puisse lire correctement les fichiers de données, ceux-ci doivent être formatés correctement et respecter un certain nombre de règles. La plus importante est que gnuplot ne sait interpréter que des **colonnes**. Les jeux de données doivent donc toujours être rangés en **colonnes** dans les fichiers.

Le fichier le plus simple à lire contient un tableau de n valeurs rangées sous la forme d'une unique colonne. Mais on peut aussi stocker plusieurs tableaux en utilisant plusieurs colonnes, ce qui implique la forme suivante [1] :

```
# x    y     z   ...
x1    y1    z1   ...
x2    y2    z2   ...
...  ...   ...   ...
xn    yn    zn   ...
```

Les fichiers peuvent aussi contenir plusieurs **blocs** de données contenant chacun une ou plusieurs colonnes [2]. Ces blocs doivent être rangés les uns à la suite des autres, et doivent être séparés par **2 lignes blanches** :

```
# bloc 0 :
x01    y01    z01   ...
x02    y02    z02   ...
...    ...    ...   ...
x0n    y0n    z0n   ...


# bloc 1 :
x11    y11    z11   ...
x12    y12    z12   ...
...    ...    ...   ...
x1n    y1n    z1n   ...


# bloc 2 :
x21    y21    z21   ...
x22    y22    z22   ...
...    ...    ...   ...
x2n    y2n    z2n   ...
```

1. Par défaut, et sans option particulière, gnuplot trace uniquement la 2e colonne en fonction de la première (même si d'autres colonnes sont présentes). Si une seule colonne est présente, il trace ce jeux de données en fonction de leur indice.

2. Par défaut, gnuplot superpose les premières courbes (2e colonne en fonction de la première) de tous les blocs et trace donc autant de courbes que de blocs.

Tracer plusieurs courbes

On peut spécifier une colonne à lire en particulier en rajoutant une option à la commande `plot` :

`plot "fichier" using 1:3`	trace la colonne 3 en fonction de la colonne 1
`plot "fichier" using (2*$1):(log($2))`	trace le log de la 2e colonne en fonction de 2x la 1ère
`plot "fichier" using 1:column(2*2)`	trace la 4e colonne en fonction de la première
`plot "fichier1", "fichier2"`	trace les colonnes 2 en fonction de 1 pour 2 fichiers
`plot "fichier" u 1:2, "" u 1:3`	trace les colonnes 2 et 3 en fonction de la 1
`plot "fichier" index 4`	ne trace que les courbes du 5e bloc
`plot "fichier" every :2::10::20`	trace les courbes des blocs 10 à 20 par pas de 2

Lorsque beaucoup de colonnes doivent être lues, il peut devenir judicieux d'utiliser la commande `for`. En voici quelques exemples d'utilisation :
`plot for [i=2:5] "fichier" using 1:i+1` : trace les colonnes 2 à 5 en fonction de la première
`plot for [i=1:10:2] "fichier" u i:i+1` : trace les colonnes 2-11 en fonction de la précédente, par pas de 2
`plot for [i=1:10:2] "fichier" using i:(column(i+1)+3*i)` : idem, mais en décalant à chaque fois les courbes de 3 unités vers le haut, ce qui est joli...

```
liste="fichier1 fichier2 fichier3"
plot for [file in liste] file using 1:2
```

Trace les 2e colonnes en fonction des premières pour tous les fichiers listés

```
filename(n) = spring("file_%d.dat",n)
plot for [i=1:10] filename(i) using 1:2
```

Définit une fonction qui retourne de nom de fichier indexé, et trace les courbes lues dans ces fichiers.

Limiter le nombre de points à tracer

Lorsque l'on a de très gros fichiers, et beaucoup de courbes à tracer, il est parfois intéressant de ne pas tracer tous les points d'une colonne donnée. On peut faire un sélection grâce à l'option `every` :
`plot "fichier.dat" every i:j:k:l:m:n` : trace les points ne correspondant qu'aux lignes de chaque bloc entre les lignes k et la m, par pas de i, et aux blocs de données entre les blocs l et n, par pas de j. Par exemple :

`plot "fichier.dat" every 2`	trace un point sur deux
`plot "fichier.dat" every ::100::200`	ne trace que les points entre 100 et 200
`plot "fichier.dat" every 2::100::200`	ne trace que les points entre 100 et 200 par pas de 2

3.4 Les plots 3D : $z = f(x, y)$

De même que pour les plots 2D, on peut tracer des plots 3D de fonctions définies dans gnuplot ou de données enregistrées dans un fichier. Dans les deux cas, la commande est : `splot`.

3.4.1 Options générales

En 3D, on peut tracer 3 types de graphes : les courbes, les contours et les surfaces. On peut choisir une représentation, l'autre, ou en superposer plusieurs.
On peut régler l'orientation du graphe avec :

`set view 10,30`	règle l'angle de visualisation 3D à 10,30
`set view map`	règle l'angle à 0,0 (vue du dessus)

Surfaces

`set surface`	trace la grille définissant la surface
`set pm3d at ss`	trace les valeurs en code de couleurs sur la surface elle-même (`b` pour en bas, `t` pour en haut)
`set pm3d map`	tracer une carte 2D en codes de couleurs (vue du dessus)
`set hidden3d`	n'affiche pas les zones qui sont cachées pour l'angle de vue donné
`set colorbox`	affiche la barre de couleur

Contours

`set contour surface`	trace les contours sur la surface elle-même
`set contour base`	trace les contours en dessous de la surface
`set contour both`	tracer les contours sur et sous la surface
`set cntrparam levels auto 20`	trace 20 niveaux
`set cntrparam levels incremental -5,1,5`	trace les niveaux entre -5 et 5 par pas de 1
`set cntrparam levels discrete -.5,0,5`	trace les niveaux 5,0,5
`set clabel`	affiche les valeurs des contours

Toutes ces options peuvent être désactivées avec `unset`.

3.4.2 Tracer une fonction

`set isosample 100,200`	définit des nombre de points 100 et 200 pour deux variables (x,y)
`f(x,y)=cos(x*x+y*y)`	définit une fonction de 2 variables
`splot f(x,y)`	trace cette fonction

3.4.3 Tracer des données

La commande pour tracer les données est : `splot "fichier.dat"`.
Et on peut sélectionner les colonnes à tracer avec : `splot "fichier" u 1:2:3` qui trace par exemple la colonne 3 en fonctions des colonnes 1 et 2.

Pour les surfaces et les contours, le fichier doit cependant être formaté correctement. Pour un jeux de données qui contient les valeurs d'un résultat $z(x, y)$ pour n_x valeur de la variable x et n_y valeurs de la variable y, le fichier doit être constitué de n_x blocs, contenant chacun 3 colonnes. La première colonne doit contenir la n_y fois la valeur de x pour le bloc considéré. Le deuxième colonne contient le tableau des n_y valeurs de y, et la troisième colonne les valeurs de z. Les blocs ne doivent pas être séparés de plus de **1 ligne blanche**.

3.4.4 Matrices

Lorsqu'aucune opération un peu technique n'est requise, on peut utiliser une manière alternative pour tracer une carte en couleur. Il s'agit de la commande :
`plot "fichier.dat" nonuniform matrix with image`
Il faut alors que le fichier soit formaté de la manière suivante :

```
 n    x01   x02   x03   ...   x0n
y01   F11   F12   F13   ...   F1n
y02   F21   F22   F23   ...   F2n
y03   F31   F32   F33   ...   F3n
...   ...   ...   ...   ...   ...
y0m   Fm1   Fm2   Fm3   ...   Fnm
```

où la première ligne contient les valeurs de x et la première colonne les valeurs de y.

3.4.5 Palettes de couleur

Dans ces cas, il peut être intéressant de choisir la palette de couleurs utiliser. Il existe plusieurs moyen de définir des palettes de couleurs. En voici quelques uns :
`set palette rgbformulae 33,13,10` (essayer aussi 22,13,-31 ou 22,0,-22 par exemple)
`set palette defined (-.4 "blue",0 "light-grey", 1. "red")`

3.5 Sauvegarder des images

Pour enregistrer une image, il faut exécuter deux étapes : choisir le type d'image à générer (eps, gif...) et choisir le nom du fichier (si les drivers ne sont pas installés, ces terminaux ne sont pas accessibles).
`set term` affiche tous les types de sorties possibles. Parmi eux :
`set term postscript eps enhanced color` dirige les sorties vers un fichier postscript avec plusieurs options
`set term tgif` dirige les sorties vers un fichier gif
`set term gif animate` dirige les sorties vers un fichier gif animé (version 4.6 uniquement)
`set term png` dirige les sorties vers un fichier png (version 4.6 uniquement)
`set term x11` redirige les sorties vers une fenêtre de l'écran
`set output "fichier"` nomme le fichier de sortie

3.6 Animations

Il est également possible de réaliser des animation (à l'écran ou dans un fichier). Le fichier de données doit alors être formaté pour contenir un bloc de données correspondant à chaque instant de l'animation. Il faut faire quelques réglage et appeler un script :

`set xrange [0:10]`	supprime l'autoscale
`imax=100`	spécifie le nombre de pas de temps de l'animation
`i=0`	initialise l'indice de boucle
`load "script.gnu"`	appelle un script qui va se charge de la boucle

Ce script doit contenir les instructions suivantes [3] :

`plot "fichier" index i`	trace l'étape i (contenue dans le bloc i)
`pause 0.2`	ralentit un peu l'exécution
`i=i+1`	incrémente le pas de temps
`if (i<=imax) reread`	retourne au début du script tant qu'on a pas fini

Si les données ne sont pas enregistrées en blocs mais en colonnes, on peut faire pareil, en adaptant la syntaxe de l'instruction `plot`.

Pour sauvegarder une animation avec les versions récentes de gnuplot, on peut utiliser le terminal "gif animate". Sinon, il faut stocker les images dans des fichiers différents, et les ouvrir avec un logiciel adapté.

3. si on met 'pause -1', il faut taper return pour continuer.