Du bon usage de gnuplot

1. Utiliser la version 3.6 de gnuplot :

```
/home3/p02/p02areuh>gnuplot-3.6  # démarrer une session du grapheur (version 3.6)
gnuplot>                         # en réponse
gnuplot>quit (ou exit)           # terminer la session gnuplot et revenir au shell
/home3/p02/p02areuh>             # en réponse
```

2. Avis …

2.1 … aux curieux :

Une aide en ligne est disponible

```
depuis le shell :  /home3/p02/p02areuh>man gnuplot # pour tout savoir sur gnuplot
depuis gnuplot:    gnuplot>help                    # id.
depuis gnuplot:    gnuplot>?plot                   # pour tout savoir sur l'ordre plot
```

La troisième solution permet d'accéder directement à la description d'une commande ou d'une option dont on connaît (ou croît connaître) le nom.

2.2 … aux paresseux :

Dans une session gnuplot, on peut exécuter une commande unix :

```
gnuplot>!emacs slurp.f90  # on édite le fichier slurp.f90 comme si on était dans le shell,
                          # point n'est besoin de fermer la session gnuplot.
```

2.3 … aux très paresseux :

Ce grapheur autorise toute abréviation correspondant à un mot unique de son lexique.
using peut être remplacé par **u**, **with** par **w**, **lines** par **l**, …
xrange ne peut être remplacé par **x** car le lexique contient en particulier **xlabel** (mais **xr** est accepté et correctement interprété).

3. Définition :

Tracé 2D = tracé nécessitant des données $\{(x_i,y_i)\}$, plus d'éventuelles incertitudes.
Tracé 3D = tracé nécessitant des données $\{(x_i,y_i,z_i)\}$.
(corollaire : un tracé de lignes de niveau est un tracé 3D)

4. Fichier de données :

- Bien noter qu'un fichier de données ne doit pas, en général, contenir l'ensemble des points calculés. En effet, pour obtenir une précision donnée on est souvent obligé d'effectuer un grand nombre de calculs. Compte-tenu de la résolution de l'imprimante, il serait ridicule (et coûteux en espace-mémoire) de les imprimer tous !
- Chaque ligne est appelée « enregistrement ». Il s'agit d'un ensemble de N nombres séparés par une ou plusieurs espaces.
- N n'est limité que par la mémoire disponible. On peut sans problème utiliser des enregistrements de quelques dizaines de colonnes, voire énormément plus.
  Ne pas abuser !
- Chaque nombre est écrit naturellement :
  par exemple -1200.6
  ou en utilisant la notation exposant :
  par exemple -1.2006e3
- Toute ligne du fichier commençant par le caractère # est ignorée par gnuplot, c'est une ligne de commentaire.

• Du rôle particulier joué par les lignes blanches.
Tout d'abord, une précaution.

F Un fichier ne doit pas commencer par une ligne blanche !

Puis, un peu de vocabulaire.

F Séquence d'enregistrement
Ensemble d'enregistrements consécutifs., tel que deux séquences consécutives ne soient séparées que par une ligne blanche (*i.e.* par un saut de ligne unique).

F Bloc d'enregistrement
Ensemble d'enregistrements (ou de séquences) tel que deux d'entre eux soient séparés par au moins deux lignes blanches.
Cela permet à gnuplot d'accéder directement à un bloc et d'ignorer le reste du fichier (voir l'option **index** dans un exemple à venir).

Enfin, différencier le rôle de ces sauts de lignes suivant le type de tracés.
On verra par la suite que deux modes de tracé sont possibles : par points ou par lignes.

F Insérer une ou plusieurs lignes blanches n'affecte aucunement le tracé par points des enregistrements contenus dans un fichier.

F L'effet produit sur un tracé par lignes diffère dans les cas 2D ou 3D :

	2D	3D
1 lb	La séquence suivante produira une nouvelle courbe.	Permet le tracé « fil de fer » d'une surface. Chaque séquence d'enregistrements à x (ou à y) fixé doit se terminer par une ligne blanche.
2 lb	Le bloc suivant produira une nouvelle courbe.	Le bloc suivant produira une nouvelle surface.

• Exemple de fichier 2D :

fichier 'humph.res' (x_i, $y_{1,i}$ $\delta y_{1,i}$ $y_{2,i}$ $\delta y_{2,i}$)
(ce qui correspond à deux ensembles de données avec incertitudes ayant des abscisses
communes)
2 séquences d'enregistrements séparées par une double ligne blanche
1ere bloc

```
0.0000000E+00  0.0000000E+00  0.1000000E+00  0.0000000E+00  0.1000000E+00
  1.000000      0.9852516      5.0000001E-02   1.033661      0.1000000
  2.000000       4.185222      0.2000000       4.275208      0.2000000
  3.000000       8.851819      0.4500000       9.763114      0.3000000
  4.000000       16.47338      0.8000000       17.08233      0.4000000
  5.000000       24.61261       1.250000       26.89037      0.5000000
                                                              # ligne blanche
                                                              # ligne blanche
```

2eme bloc

```
  6.000000       34.52370       1.800000       38.84992      0.6000000
  7.000000       49.98479       2.450000       51.32557      0.7000000
  8.000000       62.72112       3.200000       58.36615      0.8000000
  9.000000       84.30634       4.050000       74.61319      0.9000000
  10.00000       95.40314       5.000000       91.87270       1.000000
  11.00000       121.7117       6.050000       131.5402       1.100000
```

• Exemple de fichier 3D :

fichier 'splash.res'
des résultats donnés dans l'ordre : x_i, y_i, $z_{1,i}$, $z_{2,i}$
(ce qui correspond à deux nuages de points ou à deux surfaces)
un maillage de la portion rectangulaire $[-2\pi, 2\pi]\times[-2\pi, 2\pi]$ du plan (x,y) est utilisé
chaque séquence correspond aux valeurs de z, pour x fixé et pour l'ensemble des valeurs possibles de y.
Une ligne blanche sépare une séquence de la suivante

```
-9.42   -9.42   -9.42    0.00
-9.42   -9.05  -12.76   31.39
-9.42   -8.67  -15.36   55.94
-9.42   -8.29  -16.93   70.73
-9.42   -7.92  -17.33   74.47
.............................
.............................
.............................
```

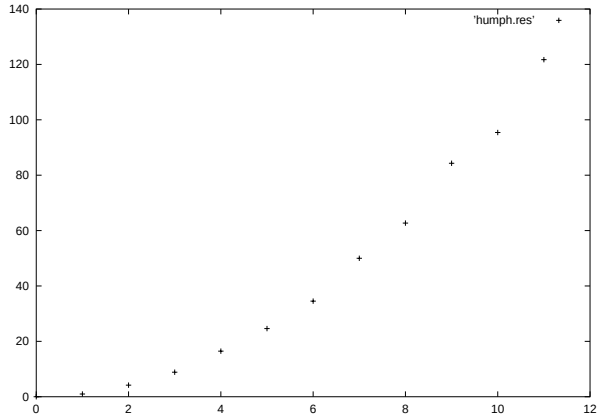
```
-9.42    7.54  -16.60   67.58
-9.42    7.92  -17.33   74.47
-9.42    8.29  -16.93   70.73
-9.42    8.67  -15.36   55.94
-9.42    9.05  -12.76   31.39
                                  # ligne blanche
-9.05   -9.42  -12.52   31.39
-9.05   -9.05  -15.24   56.04
-9.05   -8.67  -16.89   70.99
-9.05   -8.29  -17.33   74.89
-9.05   -7.92  -16.58   68.12
.............................
.............................
.............................
-9.05    7.54  -16.57   68.08
-9.05    7.92  -16.21   64.81
-9.05    8.29  -14.73   51.37
-9.05    8.67  -12.24   28.88
-9.05    9.05   -9.05    0.00
                                  # ligne blanche
.............................
.............................
.............................
                                  # ligne blanche
.............................
.............................
.............................
```

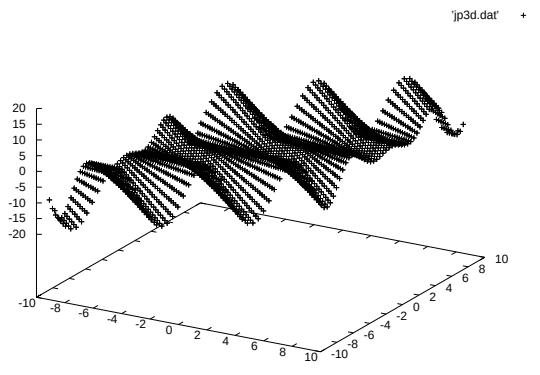
5. Commande de tracé pour un fichier de données

5.1 Les deux tracés de base :

gnuplot>**plot 'humph.res'** # la déclaration minimale pour un tracé 2D :
l'ordre (plot) et le nom du fichier concerné (humph.res).
Les options sont prises par défaut, notamment :
tracé par points de la 2ème colonne en fonction de la 1^{ere},
portées choisies en fonction des valeurs extrêmes,
tous enregistrements pris en compte,
symbole ¨ placé en chaque point,
de taille fixée.

gnuplot>**splot 'splash.res'** # la déclaration minimale pour un tracé 3D :
l'ordre (splot) et le nom du fichier concerné (splash.res),
Les options sont prises par défaut, notamment :
tracé par points de la 3ème colonne en fonction des 2 1^{ere},
portées choisies en fonction des valeurs extrêmes,
tous enregistrements pris en compte,
symbole ¨ placé en chaque point,
de taille fixée.



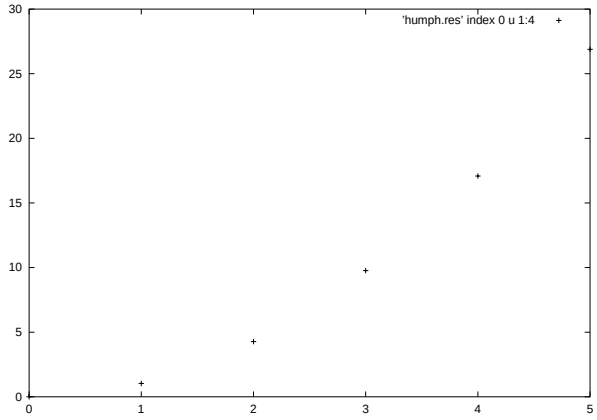


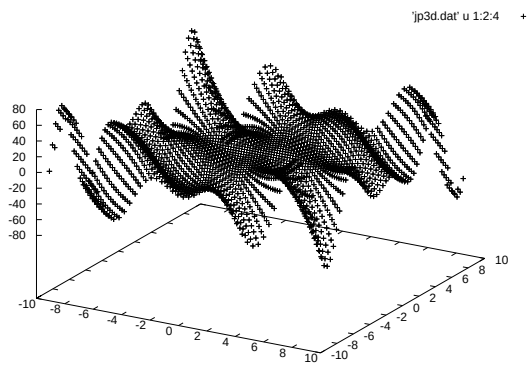
5.2 Des options de tracé sont possibles, voire, en pratique, nécessaires.

Exemples :

gnuplot>**plot [0:5] [0:30] 'humph.res' index 0 using 1:4**
trois options utilisées :
[0:5] : portée des abscisses, [0:30] : portée des ordonnées,
index 0 : tracé des seuls points de la 1e séquence,
(désolé, la 1ere porte le n° 0)
using 1:4 : tracé de la colonne 4 en fonction de la 1.

gnuplot>s**plot 'splash.res' using 1:2:4**
les trois options précédentes sont encore possibles.
on impose simplement le tracé des points correspondants aux colonnes 1,2,4.





6. Attributs graphiques

6.1 Cela concerne

la présentation générale

les tracés

6.2 La présentation générale

F A peu près tout est prévu par défaut. Pour changer un aspect particulier ou en introduire un nouveau, on doit exécuter une commande du type :
gnuplot>**set machin paramêtres_éventuels**
Tous les graphes exécutés par la suite au cours de la même session gnuplot seront parés du nouvel attribut.
Si on souhaite en annuler l'effet ou supprimer l'attribut par défaut, il faut taper :
gnuplot>**set nomachin**

F Quelques exemples utiles.

gnuplot>**set title 'joli graphe'** # place le titre entre apostrophes au-dessus du graphe
gnuplot>**set xlabel 'vla x'** # place le label entre apostrophes (vla x) contre l'axe des *x*
gnuplot>**set ylabel 'vla y'** # place le label entre apostrophes (vla y) contre l'axe des *y*
gnuplot>**set zlabel 'epi z'** # place le label entre apostrophes (epi z) contre l'axe des *z*
gnuplot>**set key 11.3,7.8** # pour faire figurer la légende à la position 11.3,7.8
(une 3eme coordonnée est possible pour un tracé 3D)
gnuplot>**set xrange [-1.5:7]** # impose les valeurs extrêmes de l'abscisse
gnuplot>**set yrange [-pi:pi]** # *id.* pour l'ordonnée
gnuplot>**set zrange [0:5.]** # *id.* pour la côte
gnuplot>**set hidden3d** # masque les faces cachées d'un dessin 3D « fil de fer »

F En pratique :
commencer par un tracé avec les attributs par défaut,
opérer éventuellement des transformations du fichier de données,
introduire avec modération les nouveaux attributs,
en parer le tracé final grâce à l'orde : gnuplot>**replot**

F Retour à la case départ :
On peut revenir aux options par défaut grâce à la commande : gnuplot>**reset**.

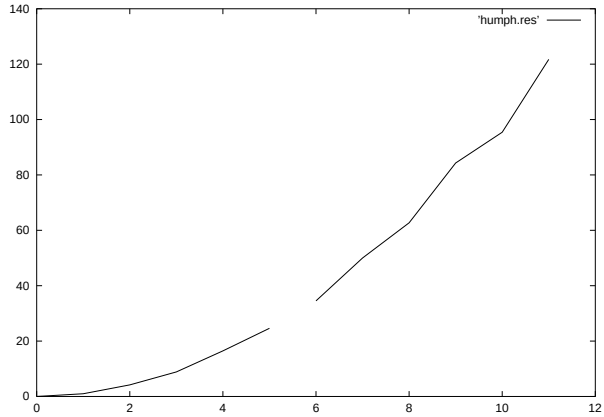
6.3 Les tracés

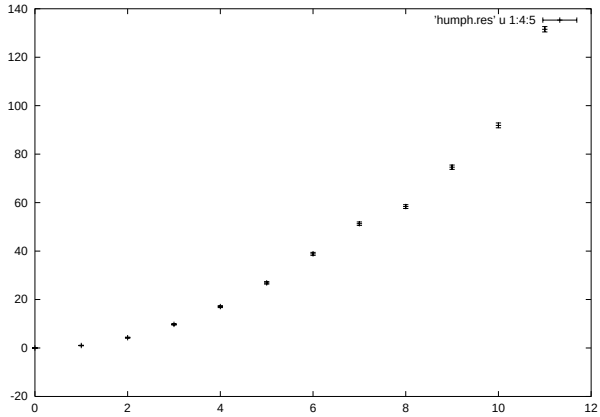
Tout est prévu par défaut. On peut malgré tout souhaiter jouer — modérément — sur le style des différents composants d'un graphe.

F On a la possibilité, comme pour la présentation générale, d'introduire des ordres du type **set truc**. Consulter à ce sujet l'aide en ligne.

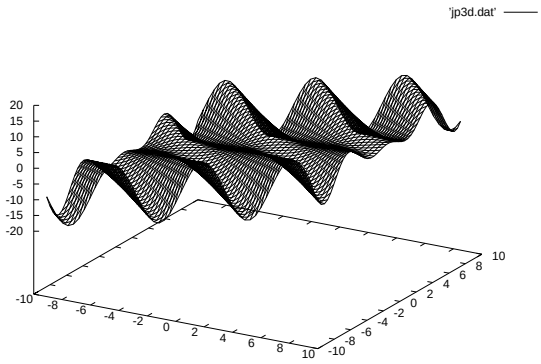
F On peut préciser, dans chaque commande de tracé, un style général. Voici les plus courants :

gnuplot>**plot 'humph.res' with points** # en fait, style par défaut
gnuplot>**plot 'humph.res' with lines** # chaque point relié au suivant par un segment
gnuplot>**plot 'humph.res' u 1:4:5 with errorbars**
colonne 4 fonction de colonne 1
barre d'incertitude (1/2 long. en colonne 5)





gnuplot>**splot 'splash.res' with points** # style par défaut
gnuplot>**splot 'splash.res' with lines** # surface « fil de fer »

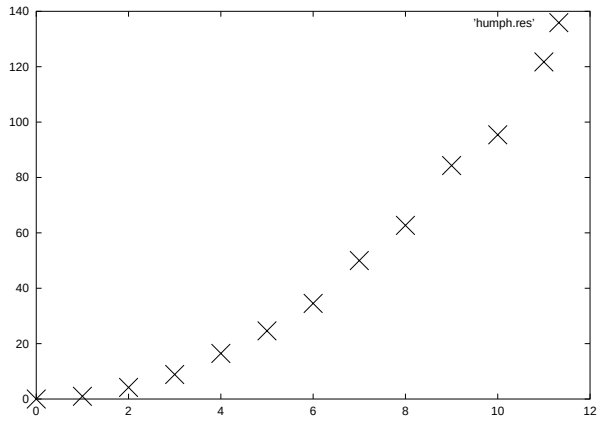


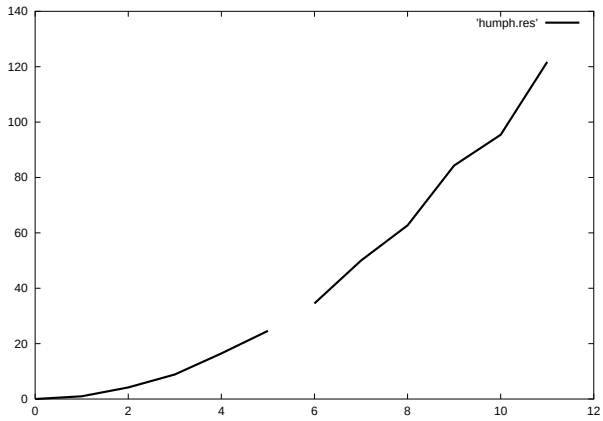
F Pour chaque style, un ou deux attributs peuvent être choisis :

ø Le symbole associé à chaque point peut être choisi parmi 6 motifs (attribut **pt**).
On peut transformer sa taille standard e_0 à partir d'un coefficient a : $e_0 \rightarrow a\ e_0$.
(attribut **ps**)

ø 8 types de lignes sont envisageables (attribut **lt**).
(à chacun est associée une couleur à l'écran, un pointillé particulier à l'impression)
L'épaisseur e_1 par défaut du trait peut être modifiée grâce à un coefficient b : $e_0' \rightarrow b\ e_1$.
(attribut **lw**)

Exemples :
gnuplot>**plot 'humph.res' w p pt 2 ps 3.5** # tracé par points de type 2 et d'épaisseur 3.5 e_0
gnuplot>**plot 'humph.res' w l lw 5.** # tracé par lignes d'épaisseur 5. e_1





7. Fichier de commandes

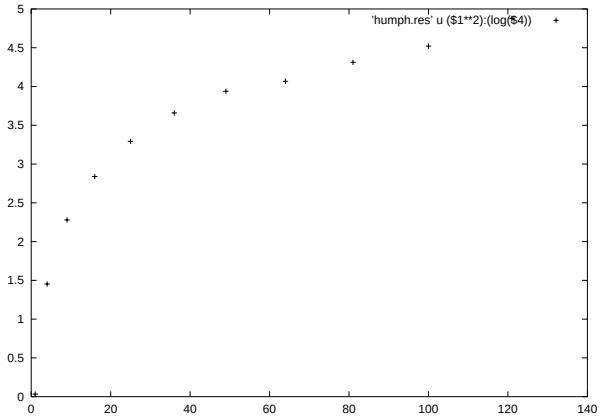
L'introduction de styles et de leurs attributs peut augmenter considérablement le nombre de commandes. Les introduire à chaque session provoque chez certains une saine allergie.
Il existe un remède !
À l'aide de votre éditeur préféré, créez un fichier, par exemple splash.com dans lequel vous introduirez la séquence — une par ligne — de commandes gnuplot correspondant au tracé embelli de la surface correspondant au fichier splash.res.
Quittez l'éditeur…
Activez gnuplot puis lancez l'exécution des commandes contenues dans splash.res :
gnuplot>**load 'splash.com'**

8. Transformation des données

Il s'agit de tracer, à partir d'un fichier de données $\{(x_i, y_i)\}$, $g(y_i)$ en fonction de $f(x_i)$.

```
gnuplot>f(x)=x**2                          # on définit f
gnuplot>g(y)=log(y)                        # on définit g
gnuplot>plot 'humph.res' u (f($1)):(g($4))  # on trace, g(col.4) en fonction de f(col.1)
                                           # $ est obligatoire, les parenthèses aussi
```

Mais on peut condenser en une commande :
gnuplot>**plot 'humph.res' u ($1**2):(log($4))** # sobre, de bon goût

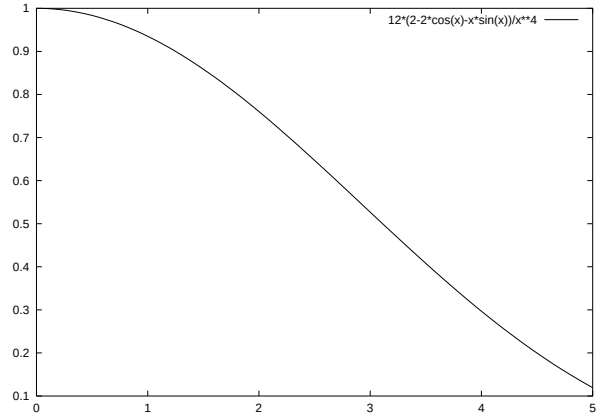


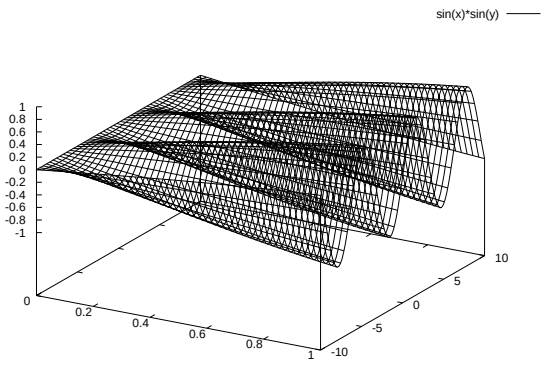
9. Tracé de fonctions

9.1 Représentation cartésienne

```
gnuplot>f(x)=12*(2-2*cos(x)-x*sin(x))/x**4 # on définit une fonction d'une variable
gnuplot>plot [0:5] f(x)                    # on en trace le graphe sur [0,5]

gnuplot>h(x,y)=sin(x)*sin(y)               # on définit une fonction de deux variables
gnuplot>splot [0:1] [-10:10] h(x,y)        # on trace la surface z=h(x,y)
                                           # (sur [0,1]×[-10,10])
```



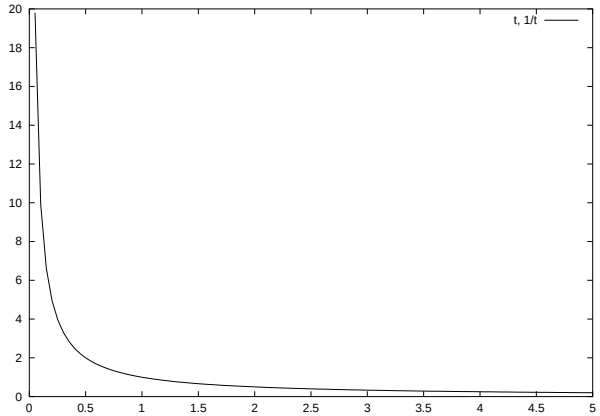


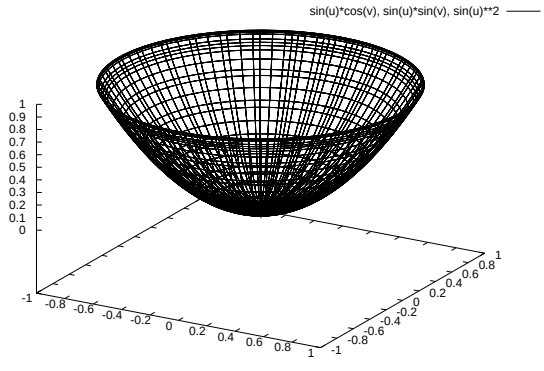
Remarque.
Chacune de ces paires de commandes peut être ramenée à une seule.
Exemple : gnuplot>s**plot [0:1] [-10:10] sin(*x*)∗sin(*y*)**

9.2 Représentation paramétrique

gnuplot>**set parametric** # on entre dans le mode « parametric »
gnuplot>**plot [0:5] *t,1/t*** # on trace le graphe sur [0,5] de $x(t)=t, y(t)=1/t$
t est le nom imposé du paramètre

gnuplot> **set parametric** # on entre dans le mode « parametric »
gnuplot>**splot sin(*u*)∗cos(*v*),sin(*u*)∗sin(*v*),sin(*u*)∗∗2**
on trace le graphe du paraboloïde :
$x=\sin(u)*\cos(v), y=\sin(u)*\sin(v), z=\sin(u)**2$
u et *v* sont les noms imposés des paramètres

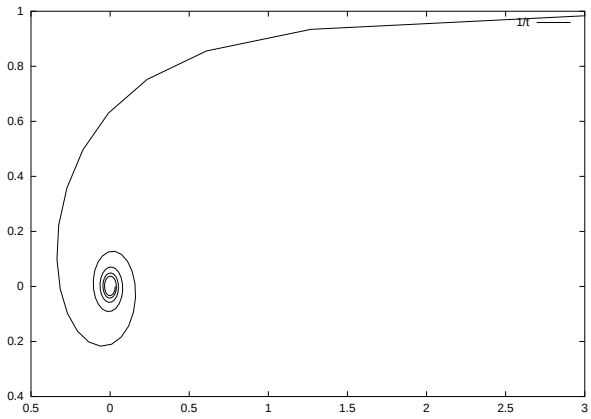




Remarques.
Des portées peuvent être imposées, mais elles concernent cette fois le(s) paramètre(s).
Tout tracé ultérieur d'une fonction en représentation cartésienne doit être précédé par une sortie du mode parametric (**set noparametric**).

9.3 Représentation polaire

gnuplot>**set polar** # on entre dans le mode « polar »
gnuplot>**plot [0:10∗pi] 1/*t*** # on trace le graphe sur $[0,10\pi]$ de $r(t)=1/t$
t est le nom imposé du paramètre



Remarques
Des portées peuvent être imposées, elles concernent l'angle *t*, *x* et *y*.
Tout tracé ultérieur d'une fonction en représentation cartésienne doit être précédé par une sortie du mode polar (**set nopolar**)

9.4 Remarques générales

F Les fonctions peuvent dépendre de constantes. Simplement, elles doivent être précisées avant (ou à) la commande de tracé. Est autorisé : gnuplot>**plot *a**x**2 , *a*=.45**

F Par défaut, une courbe est tracée avec 100 points, une surface avec un treillis de 10×10 isolignes (*i.e.* lignes de *x* ou de *y* donné)de 100 points. Dans l'un et l'autre cas, les points sont liés 2 à 2 par des segments. (pour modifier, voir **set samples**)

10 Superposition de plusieurs tracés

Lorsqu'on souhaite superposer sur une même fenêtre plusieurs courbes, on peut utiliser une des trois méthodes ci-dessous.

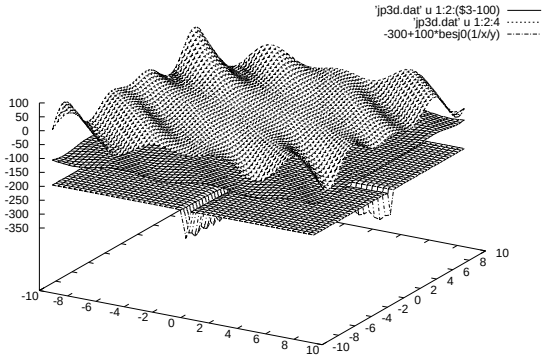
F 1ere méthode
On peut utiliser un fichier de données comprenant plusieurs séquences d'enregistrements séparées d'une ligne blanche.
gnuplot>**plot 'humph.res' w l**
À chaque séquence est associée un tracé.

F 2eme méthode
En une seule commande, on peut superposer les tracés correspondants à plusieurs fichiers et à des fonctions :

```
gnuplot>splot 'splash.res' u 1:2:($3-100) w l\  # \ signale que la commande se poursuit à
, 'splash.res' u 1:2:4 w l\                     # la ligne suivante
,-300+100*besj0(1/x/y)                          # 0r donc, 1 commande sur 3 lignes !
```

F 3eme méthode
On use de la commande **multiplot** :

```
gnuplot>set multiplot                           # ouvre la session de tracés multiples
multiplot>splot 'splash.res' u 1:2:($3-100) w l # 1 fenêtre, 1 tracé
multiplot>splot 'splash.res' u 1:2:4 w l        # 1 fenêtre, 2 tracés
multiplot>splot -300+100*besj0(1/x/y)           # 1 fenêtre, 3 tracés
multiplot>splot …                               # 1 fenêtre, … tracés
multiplot>…                                     # …
multiplot>set nomultiplot                       # pour que les tracés suivants soient simples
gnuplot>…
```

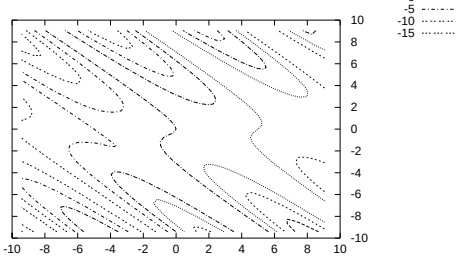


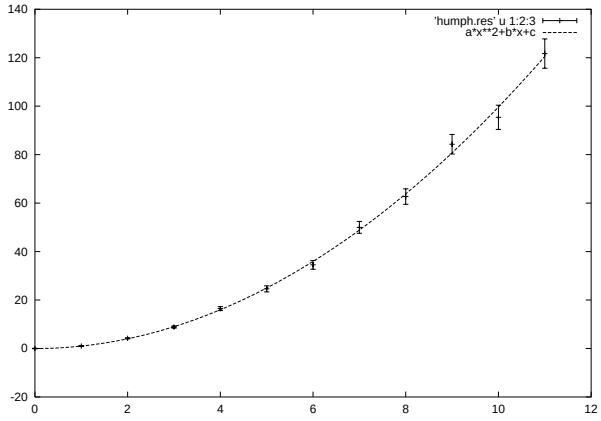
11 Tracé de lignes de niveau

gnuplot>**set contour** # pour faire apparaître des lignes de niveau
dans le plan (x,y) (par défaut, il y en a 5)
gnuplot>**set cntrparam levels 15** # impose 15 l. de n. (équiréparties entre z_{min} et z_{max})
gnuplot>**set nosurface** # la surface disparaît : il ne reste que les l. de n.
gnuplot>**set view 0.,0.** # donne une orientation habituelle aux axes…
gnuplot>**splot 'splash.res' w l** # la récompense !

12 Réaliser un « fit » de données

gnuplot>**f(x)=a*x**2+b*x+c** # définition de la fonction fit
gnuplot>**fit f(x) 'humph.res' u 1:4:5 via *a,b,c***
« fit » des données {x_i (en colonne n° 1),
y_i (en colonne n° 2) affectée de l'incertitude δy_i
(en colonne n° 3)} par la fonction $f(x)$.
La déclaration des paramètres (via a,b,c) est obligatoire.





Remarques :

- Lorsqu'on ne se réfère qu'à deux colonnes (ex. u 1:4) on réalise un fit non pondéré.
- On peut concaténer les lignes de commande en une seule :
gnuplot>**fit *a*x**2+*b*x+c* 'humph.res' u 1:4:5 via *a,b,c***
- Le fit peut être non linéaire (par ex. (a+b*x**2)*exp(c*x**2))
- On peut tracer les données et le fit obtenu en une seule commande :
gnuplot>**plot *a*x**2+b*x+c*, 'humph.res' u 1:4:5 w errorbars**

13 Mettre de l'animation

Supposons qu'une grandeur y soit fonction de deux autres x et a. Il est souvent intéressant de représenter, pour a fixé, les variations de y en fonction de x mais aussi de suivre l'évolution du graphe associé lorsque a varie. Par exemple on peut souhaiter suivre

l'évolution au cours du temps ($a \equiv t$) de la répartition spatiale d'un paquet d'onde unidimensionnel ($x \equiv x,\ y \equiv \psi$).

1ere cas
On dispose d'un fichier schmol.dat contenant des enregistrements de 11 colonnes. La première colonne représente l'ensemble des valeurs de *x*, les 10 autres, les valeurs correspondantes de *y* pour 10 valeurs du paramètre *a*. On a créé un fichier de commande, disons schmol.com, capable d'afficher le graphe de la colonne *i* en fonction de la colonne 1 lorsque *i* varie entre 2 et 11.

```
gnuplot>set yrange [0.:20.]  # obligatoire. On doit imposer une portée commune.
gnuplot>i=2                  # pour que le premier tracé soit col. 2 en fonction de col. 1
gnuplot>imax=11              # pour que le dernier tracé soit col. 11 en fonction de col. 1
gnuplot>load 'schmol.com'    # pour lancer l'exécution des commandes gnuplot contenues
                             # dans le fichier schmol.com
```

contenu du fichier de commande :

```
#schmol.com                 # une ligne de commentaires
plot 'schmol.dat' u 1:i w l  # noter que le numéro de la 2eme colonne est i
i=i+1                        # incrémente le n° de colonne
if (i<=imax) reread          # si i≤11, relance l'exécution des commandes de ce fichier
                             # depuis la 1ere
```

2eme cas
Si l'ensemble des valeurs de *x* change à chaque graphe, on ne peut plus utiliser la méthode précédente. Le plus simple consiste à créer un fichier, disons dugudu.dat, contenant 10 blocs d'enregistrements (séparés par une double ligne blanche). À chaque bloc devra être associée une courbe.

séquence gnuplot

```
gnuplot>set xrange [0.:5.]
gnuplot>set yrange [0.:20.]
gnuplot>i=0                  # pour que le premier tracé corresponde au 1er index (0)
gnuplot>imax=10              # pour que le dernier tracé corresponde au dernier index (10)
gnuplot>load 'dugudu.com'    # pour lancer l'exécution des commandes gnuplot contenues
          # dans le fichier dugudu.com
```

fichier de commande

```
#dugudu.com                 # une ligne de commentaires
plot 'dugudu.dat' index i w l # l'index i désigne le bloc i (le 1er étant numérotée 0)
i=i+1                        # incrémente l'index
if (i<=imax) reread          # si i≤10, relance l'exécution des commandes de ce fichier
                             # depuis la 1ere
```

14 Quelques plus

14.1 Afficher heure et date

```
gnuplot>set time             # affiche heure et date
                             # dans le coin inférieur gauche de la fenêtre graphique
```

14.2 Défendre les petits nombres ?

Dans gnuplot, par défaut, les nombres compris entre -1.e-8 et +1.e-8 sont pris égaux à zéro. Si un fichier de données ne contient que des abscisses (et/ou des ordonnées, et/ou des côtes) comprises dans cet intervalle, elles seront traitées comme étant toutes nulles. La mise à l'échelle automatique ne peut plus fonctionner et la portée de l'axe correspondant sera celle par défaut !

F Un remède est souvent envisageable :
Ayant correctement formulé le problème à traiter, seules des variables sans dimension interviennent (!) dont la grandeur caractéristique est bien souvent de l'ordre de l'unité…

F Vous tenez absolument à pactiser avec des nombres très petits, vous appliquez la potion magique :

```
gnuplot>set zero 1.e-15      # dans cet exemple, seuls les nombres de valeur absolue <
                             # 1e-15 sont pris égaux à 0
```

14.3 Obtenir des échelles (semi) logarithmiques

```
gnuplot>set logscale yz e    # pour obtenir des échelles logarithmiques en base e sur
                             # l'axe des y et celui des z dans le cas 3D
```

Remarques :

F *yz* peut être remplacé par toute combinaison de *x*, *y* et *z*, *e* par tout nombre (positif). C'est, bien sur et en premier lieu, utilisable pour un tracé 2D.

F Attention, on ne peut user d'une échelle logarithmique sur un axe, disons des *x*, qu'à la condition que les extrémités de la portée soient strictement positifs (logarithme oblige !)

14.4 Des segments et des flèches

Il peut être utile de faire figurer sur un tracé des flèches et/ou des segments de droite. Un ordre permet de le réaliser qui utilise les coordonnées du tracé. Curieusement, la ligne — on devrait plutôt dire le segment — n'est prévue qu'en terme de cas particulier de la flèche.

```
gnuplot>plot [-0.1:0.1] 'smurf.res' u 2:1 w l          # un tracé sans histoire
gnuplot>set arrow 12 from .1,1.12 to -0.034,1. nohead  # on veut ajouter une ligne
gnuplot>replot                                         # tracé + ligne
gnuplot>set arrow 25 from 0.05,1. to 0.05,1.           # on veut introduire une flèche
gnuplot>replot                                         # tracé + ligne + flèche
gnuplot>plot [0:100.] 'gloups.don'                     # nouveau tracé + ligne + flèche
gnuplot>set noarrow 12                                 # pour supprimer la ligne 12
gnuplot>replot                                         # nouveau tracé + flèche
```

Dans cet exemple, la 2eme commande prépare l'ajout aux tracés qui vont suivre, d'un segment (car option nohead) joignant les points de coordonnées (0.1,1.12) et (-0.034,1.). (Pour gnuplot cette « flèche » porte le numéro (optionnel !) 12.
La septième commande permet de supprimer dans les tracés ultérieurs la ligne n°12.

14.5 Vive la pause

Il peut être intéressant de contrôler, lors d'une séquence de tracés, le temps d'affichage de chaque image. Reprenons l'exemple donné en 13 en modifiant légèrement le contenu du fichier de commande :

```
#schmol.com
plot 'schmol.dat' u 1:i w l
pause -1                  # le tracé suivant n'apparaît qu'après un « return »
                          # si -1 est remplacé par 5, chaque tracé reste affiché 5 secondes
i=i+1
if (i<=imax) reread
```

14.6 Imposer les dimensions de la fenêtre d'affichage

```
gnuplot>set size 1.5,0.5  # Donne une surface d'impression de côtés L' = 1,5 L
                          # (pour les abscisses), l' = 0,5 l (pour les ordonnées),
                          # où L et l sont les dim. par défaut de la zone d'impression.
gnuplot>set size .721,1.  # Donne une surface d'impression carrée
                          # (ce n'est qu'approximatif à l'écran !)
                          # Par ex., permet d'obtenir des axes orthonormés
                          # (en cas de portée identique pour les x et les y)
```

15 Faire bonne impression

gnuplot est capable de générer un fichier postscript, langage que comprend l'imprimante du service d'enseignement. D'où :

```
gnuplot>set term postscript      # demande à gnuplot de passer en mode « postscript »,
                                 # plus rien ne s'affiche à l'écran qui ne connaît pas ce
                                 # langage !
gnuplot>set output 'smurf.ps'    # pour envoyer la description postscript du tracé à venir
                                 # vers le fichier smurf.ps
gnuplot>plot 'smurf.dat'         # justement, on lance le tracé lié au fichier smurf.dat…
gnuplot>set term x11             # exige de gnuplot de revenir au mode « console X »
                                 # toute nouvelle commande de tracé se matérialisera de
                                 # nouveau à l'écran
gnuplot>!lpr -P impmag smurf.ps  # pour immortaliser votre œuvre sur velin
```