

Table des matières

1	Préambule	2
2	Faire un plot 2D	2
2.1	À partir d'un fichier	2
2.2	Tracer une fonction à partir de la formule	2
2.3	Modifier les données avant de les afficher	3
2.4	Faire une regression à partir de données	3
3	Faire un plot 3D	3
3.1	Surface plot	3
3.1.1	Modifier la palette de couleurs	4
3.1.2	Ajouter des contours à une surface	4
3.1.3	Ajouter un champ de vecteur à une surface	7
3.2	Afficher un champ de vecteur	7
4	Exporter la courbe	8
4.1	Affichage avancé à l'écran	8
4.2	Fichier pdf	8
4.3	fichier svg	8
4.4	fichier png	8
5	Les options	8
5.1	label	8
5.2	échelle logarithmique	9
5.3	grille	9
5.4	Forcer un zoom	9
5.5	Style de courbe	9
5.5.1	Les lignes	9
5.5.2	Les points	10
5.6	Texte avancé	10
6	FAQ	11
6.1	pm3d map : l'image ne remplit pas le graphique	11
6.2	pm3d map : mauvais affichage de la première colonne	12
6.3	Invalid character \	13

1 Préambule

Gnuplot permet de tracer des courbes de manière plus ou moins complexes, en 2D ou en 3D. Ce formulaire est là pour donner des pistes et pouvoir faire rapidement des choses basiques. Il n'a pas la prétention d'être exhaustif, ce qui rendrait l'information plus difficile à retrouver et à séparer de l'essentiel que je m'efforce de mettre en valeur.

Gnuplot se présente sous la forme d'un logiciel en ligne de commande que l'on utilise en faisant :

```
gnuplot
```

Toutes les commandes que je présente peuvent être entrées de manière interactive dans **gnuplot**. Je ne présenterais pas cette manière de faire, dans la suite je présenterais une suite d'instruction que j'enregistre dans un fichier (au nom quelconque, mais à qui je donne l'extension **.gnuplot** pour plus de lisibilité. Afin de s'en servir on fait alors :

```
gnuplot fichier.gnuplot
```

Remarque : La différence principale entre le script ***.gnuplot** et l'affichage interactif, c'est que le plot affiché à l'écran (terminal **x11**) ne reste pas par défaut, il faut ainsi mettre un **pause(-1)** en fin de script pour que le graphique reste, et un appui sur Entrée continue l'exécution du script (et le termine s'il n'y a rien ensuite).

2 Faire un plot 2D

2.1 À partir d'un fichier

Admettons que le fichier **plot.dat** contienne deux colonnes dans lesquelles il y a respectivement les valeurs de x et y pour notre graphique. Afin d'afficher le graphique il suffit de faire :

```
plot 'plot.dat' using 1:2
```

On va alors tracer la deuxième colonne en fonction de la première.

On peut aussi tracer plusieurs courbes :

```
plot 'plot.dat' using 1:2 title 'courbe1', '' using 1:3 title 'courbe2', \
'' using 1:4 title 'courbe3', 'plot2.dat' using 1:2 title 'courbe4'
```

Pour tracer 4 courbes, 3 venant du fichier **plot.dat**, la 2^e, 3^e et 4^e colonne en fonction de la première et une courbe venant de **plot2.dat**.

2.2 Tracer une fonction à partir de la formule

On peut, au lieu de rentrer un tableau de points, tracer directement une fonction connue :

```
plot sin(x)
```

Les fonctions dans gnuplot sont les mêmes que les fonctions dans la librairie mathématique Unix à part que les fonctions acceptent des entiers des réels ou des complexes comme argument.

Remarque : Pour les fonctions qui acceptent ou retournent des angles qui pourraient être donnés en degrés ou en radian ($\sin(x)$, $\cos(x)$, $\tan(x)$, $\arcsin(x)$, $\arccos(x)$, $\arctan(x)$, ...) l'unité peut être sélectionnée par **set angles** (par défaut, ce sont des radians).

On peut alors faire par exemple :

```
plot (sin(x)/x)**2
```

un sinus cardinal au carré.

Si le nombre de points n'est pas assez élevé, on peut alors rajouter

```
set samples 200
```

où 200 est le nombre de points servant à tracer la fonction. Si vous êtes en interactif ou que l'option **samples** est utilisée après le **plot**, il faut aussi utiliser

```
replot
```

2.3 Modifier les données avant de les afficher

Il est possible d'effectuer des opérations mathématiques sur les colonnes d'un fichier de données avant de les afficher.

Par exemple :

```
plot "test.dat" using 1:2          with points,\
    "test.dat" using 1:($2*2)      with points,\
    "test.dat" using 1:(sqrt($2)) with points,\
    "test.dat" using 1:(log($2))  with points
```

La première courbe sera la courbe par défaut, sans retraitement, la colonne 1 étant la colonne des x et la colonne 2 celle des y .

La deuxième courbe, on multiplie les valeurs de la colonne 2 par 2. La troisième courbe, on prend la racine carrée et la 4^e, on prend le logarithme.

Pour faire une opération sur la colonne i , il faut utiliser la variable $\$i$ et utiliser les fonctions mathématiques standard dessus.



Il ne faut pas oublier les parenthèses entourant les opérations mathématiques afin que toute la parenthèse soit considérée comme une nouvelle colonne à utiliser dans le graphique à venir.

2.4 Faire une regression à partir de données

Pour celà, il faut déjà connaître l'expression f reliant les deux colonnes de données. Cette expression peut dépendre de plusieurs paramètres, comme ici :

$$f(x) = T_0 \times x^{-\beta} \quad (2.1)$$

Le code pour faire ça en gnuplot est alors :

```
f(x) = T_0 * x**(-beta)
T_0 = 1015 ; beta = 1.6
fit f(x) 'temperature_profile.dat' using 1:2 via T_0, beta
plot f(x) with lines, 'temperature_profile.dat' with points
pause -1
show variables
```

On peut varier le nombre de paramètres, faire intervenir des fonctions intrinsèques (gnuplot en connaît beaucoup).



On peut se retrouver avec l'erreur suivante lorsqu'on essaie de calculer la régression :

Undefined value during function evaluation

Ça peut être dû au fait que les valeurs d'initialisation des paramètres résultent en une fonction indéfinie dans une partie du domaine des données expérimentales. Ce problème peut d'ailleurs survenir au cours de la convergence.

Mais ça peut être aussi quelque chose de beaucoup plus radical. J'ai parfois eu ce problème quand l'équation que je cherchais à utiliser ne décrivait tout simplement pas les données expérimentales. En rajoutant alors une constante, ça a non seulement convergé, mais le résultat était beaucoup plus satisfaisant.

Il faut donc faire attention à l'équation qu'on utilise, et il vaut mieux savoir à l'avance ce qu'on cherche.

3 Faire un plot 3D

3.1 Surface plot

Pour afficher un graphique sous la forme d'une surface (x, y) dont la couleur dépend de z , il faut d'une part préparer le fichier de données de manière spéciale. Les données doivent être sous la forme :

```
# X      Y      Z
0.0    0.0    2.0
0.0    1.0    2.0
0.0    1.0    3.5
0.0    2.0    3.5

1.0    0.0    2.0
1.0    1.0    2.0
1.0    1.0    3.5
1.0    2.0    3.5

1.0    0.0    1.0
1.0    1.0    1.0
1.0    1.0    3.0
1.0    2.0    3.0

2.0    0.0    1.0
2.0    1.0    1.0
2.0    1.0    3.0
2.0    2.0    3.0
```



Il faut séparer les blocs de $X = c^{te}$ par une ligne blanche sans quoi **gnuplot** ne fonctionnera pas.

Ensuite, il faut utiliser les commandes suivantes :

```
set terminal x11 enhanced
set xlabel "semi major axis (AU)"
set ylabel "Planet mass (m_{earth})"
set title "Evolution of the total torque {/Symbol G}"
set pm3d map
splot 'test_total_torque.dat' title ''
```

dans `splot`, le titre vide est là pour ne pas avoir le nom du fichier imprimé au milieu de la courbe. `map` est l'option qui permet d'avoir une carte au lieu d'avoir le graphique 3D. Il est aussi possible d'obtenir la même chose en utilisant l'option qui permet de changer l'angle de visée sous lequel regarder le graphe 3D `set view map; unset surface`

Remarque : Je n'ai pas réussi à imposer, via `set {x|y}range` le zoom du graphique. Dans les bords, en particulier quand l'échantillonnage est faible, ça peut ne pas être très joli.

3.1.1 Modifier la palette de couleurs

Afin de changer la palette de couleurs, et ainsi avoir de meilleurs contrastes avec les couleurs proches, on peut faire

```
set palette rgbformulae 22,13,-31
```

et avoir une sorte d'arc en ciel.

3.1.2 Ajouter des contours à une surface

Ci-dessous voici le contenu d'un script que j'utilise pour afficher un contour sur l'iso-ligne $z = 0$:

```
set contour base; set cntrparam levels discret 0.
unset surface
set table "contour.dat"
set dgrid3d 30,30,10
splot 'test_total_torque.dat'
unset table
# Draw the plot
```

```

reset
set terminal wxt enhanced
set xlabel "semi major axis (AU)"
set ylabel "Planet mass (m_{earth})" center
set title "Evolution of the total torque  $\Gamma_{\text{tot}}/\Gamma_0$  "
set pm3d map
set pm3d explicit
set palette rgbformulae 22,13,-31
splot 'test_total_torque.dat' with pm3d notitle,\
'contour.dat' with line linetype -1 title ' $\Gamma=0$ '
#pause -1 # wait until a carriage return is hit
set terminal pngcairo enhanced
set output 'total_torque.png'
replot # pour générer le fichier d'output
!rm contour.dat

```

On obtient alors [FIGURE 1]

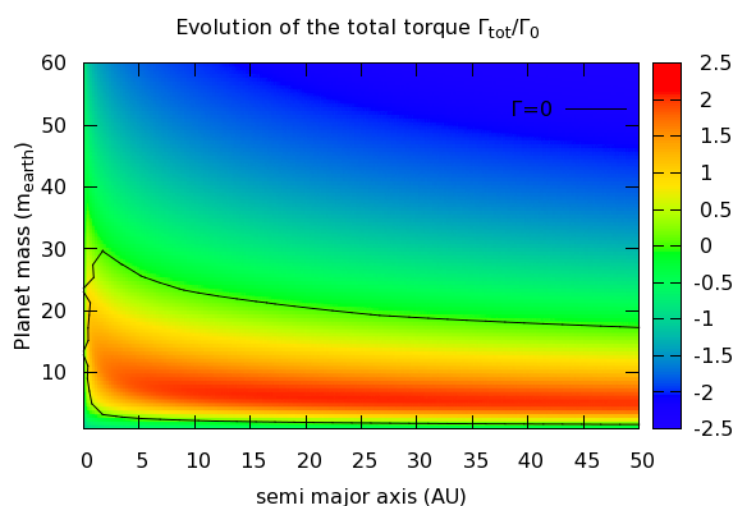


FIGURE 1 – Affichage d'une carte 2D d'un fichier de donnée avec des contours en surimpression

```
set contour base; set cntrparam levels discret 0.
```

On définit les niveaux des contours (ici, uniquement une ligne pour $z = 0$).

```
unset surface
set table "contour.dat"
```

On enlève la surface et on définit un fichier dans lequel on va stocker les données de sortie.

```
set dgrid3d 50, 50, 10
```

La ligne `set dgrid3d` est dans le cas d'un fichier de données (contrairement à une fonction que l'on définirait directement dans gnuplot ($f(x,y) = x * x - y * y$ par exemple)). Comme on veut tracer et faire un contour d'un tableau de donnée, on a besoin de convertir ce tableau de donnée en grille dans un premier temps.



L'échantillonnage en x , y et z est crucial pour avoir un rendu correct des contours. Ce maillage est à adapter. J'ai eu des problèmes pendant 2 jours parce que je ne définissais pas du tout de maillage (il en fait un par défaut je suppose) et je pestais parce que je n'avais pas de contours du tout (alors que je savais qu'il existait une ligne $z = 0$).

```
splot 'test_total_torque.dat'
unset table
# Draw the plot
reset
```

On trace le tableau de donnée **test_total_torque.dat**, comme on a défini **unset surface**, le tableau de donnée ne sera pas affiché. Une grille 3D est alors effectuée (**set dgrid3d 50, 50, 10**) et un contour de cette grille pour les valeurs définies est recherché (**set cntrparam levels discret 0.**). Le résultat est alors stocké dans **contour.dat**.

On désactive alors la table, et on double les lignes blanches (apparemment nécessaires à l’affichage correct des contours pour la suite. J’ai testé et ça semble fonctionner même sans ça), ces nouvelles données sont alors stockées dans **contour1.dat**.

Un **reset** est alors effectué pour pouvoir tracer la courbe pour de bon.



Les contours seront ceux du fichier **test_total_torque.dat**, celui ci doit bien évidemment être le même que celui qui sera affiché en même temps que le contour dans la 2^e partie du script, sinon ça n’a pas de sens.

```
set terminal wxt enhanced
set xlabel "semi major axis (AU)"
set ylabel "Planet mass (m_{earth})" center
set title "Evolution of the total torque {/Symbol G}_{tot} "
```

On définit le terminal et les titres. Les labels ne sont bien entendu pas obligatoire.

```
set pm3d map
set pm3d explicit
```

On définit le fait qu’on peut un plot 3D, affiché uniquement en 2D avec la variation en z affichée selon une palette de couleur. On veut que les plots soient explicitement spécifiés comme 3D, ceci nous permet d’afficher les contours simplement en 3D (des lignes noires quoi).

```
set palette rgbformulae 22,13,-31
```

On change la palette de couleur (c’est un arc en ciel, plus pratique pour discerner les couleurs, et donc les valeurs associées).

```
splot 'test_total_torque.dat' with pm3d notitle,\
      'contour.dat' with line lt -1 title '{/Symbol G} = 0'
```

On affiche les données de **test_total_torque.dat** et le contour **contour1.dat** (qui est censé correspondre à **test_total_torque.dat** sinon ça n’a pas de sens). La partie **title ''** permet d’éviter l’affichage du nom du fichier sur la courbe.

```
pause -1 # wait until a carriage return is hit
```

Le graphique restera affiché à l’écran tant qu’on n’a pas appuyé sur la touche Entrée (et ainsi continuer le script)

```
set terminal pngcairo enhanced
set output 'total_torque.png'
replot # pour générer le fichier d’output
```

On stocke le graphique dans un fichier **.png**

```
!rm contour.dat
```

On efface le fichier temporaire qui a été créé pour stocker le contour.

3.1.3 Ajouter un champ de vecteur à une surface

Ci-dessous voici le contenu d'un script que j'utilise pour afficher un contour sur l'iso-ligne $z = 0$:

```
set contour base; set cntrparam levels discret 0.
unset surface
set table "contour.dat"
set dgrid3d 30,30,10
splot 'test_total_torque.dat'
unset table
# Draw the plot
reset
set terminal wxt enhanced
set xlabel "semi major axis (AU)"
set ylabel "Planet mass ( $m_{\text{earth}}$ )" center
set title "Evolution of the total torque  $\{G_{\text{tot}}\}/\{G_0\}$  "
set pm3d map
set pm3d explicit
set palette rgbformulae 22,13,-31
splot 'test_total_torque.dat' with pm3d notitle,\
    'contour.dat' with line linetype -1 title ' $\{G\}=0$ ',\
    'test_vector_total_torque.dat' with vector notitle head filled linestyle -1
#pause -1 # wait until a carriage return is hit
set terminal pngcairo enhanced
set output 'total_torque.png'
replot # pour générer le fichier d'output
!rm contour.dat
```

Pour plus de détails, voir [§ 3.1.2 on page 4].

La ligne différente est :

```
'test_vector_total_torque.dat' with vector notitle head filled linestyle -1
```

Pour afficher un champ de vecteur avec la flèche remplie (**head filled**), sans titre dans la légende (**notitle**), et en trait plein noir (**linestyle -1**).

Pour plus de détails sur les champs de vecteurs, voir [§ 3.2].

3.2 Afficher un champ de vecteur

Pour faire un champ de vecteurs, il faut utiliser **with vector**, soit avec **plot** soit avec **splot**.

Avec **plot** il faut quatres colonnes dont la syntaxe est $x, y, \delta x, \delta y$ afin d'afficher des vecteurs dont les coordonnées sont, pour une ligne donnée $x, y, \delta x, \delta y$: $((x, y), (x + \delta x, y + \delta y))$:

```
plot 'vector.dat' with vector
```

ou

```
plot 'vector.dat' using 1:2:3:4 with vector
```

Avec **splot**, il faut six colonnes dont la syntaxe est $x, y, z, \delta x, \delta y, \delta z$ afin d'afficher des vecteurs dont les coordonnées sont, pour une ligne donnée $x, y, z, \delta x, \delta y, \delta z$: $((x, y, z), (x + \delta x, y + \delta y, z + \delta z))$:

```
splot 'vector.dat' with vector
```

ou

```
splot 'vector.dat' using 1:2:3:4:5:6 with vector
```

4 Exporter la courbe

Le principe de gnuplot est de sélectionner un terminal où envoyer les données du graphique. par défaut, ce terminal est **x11**.

Pour définir un terminal il suffit de rentrer

```
set terminal x11
```

Cette dernière commande est la commande par défaut, celle qui permet d'afficher les graphiques à l'écran.

Dans le mode interactif de gnuplot, on accède à la liste des terminaux via

```
help terminal
```

4.1 Affichage avancé à l'écran

Voici un terminal que j'aime bien parce qu'il permet de zoomer sur la courbe notamment

```
set terminal wxt enhanced
```

Remarque : `enhanced` permet une gestion avancée des textes, notamment avec indice et exposant. Pour plus de détails voir [§ 5.6 on page 10].

4.2 Fichier pdf

```
set terminal pdfcairo
set output 'graphique.pdf'
replot
```

Remarque : Le `replot` n'est utile que si la courbe a déjà été affichée (à l'écran par exemple).

4.3 fichier svg

```
set terminal svg rounded size 450,360
set output 'graphique.svg'
replot
```

4.4 fichier png

Pour faire ceci en terminal principal, il faut entrer en préambule les lignes suivantes :

```
set terminal pngcairo
set output 'graphique.png'
```

Pour, au contraire, afficher le graphique, puis l'exporter en png :

```
set terminal pngcairo
set output 'graphique.png'
replot
```

L'option **crop** permet d'éliminer les blancs autour du graphique, mais dans ce cas, la résolution que vous aurez en sortie sera plus petite que la résolution que vous avez demandé. Le graphique aura en fait la même taille dans les deux cas, mais les zones parfaitement blanches autour du graphique seront ôtées.

J'utilise pour ma part la ligne suivante :

```
set terminal pngcairo crop enhanced size 1200, 1000
```

5 Les options

5.1 label

```
set xlabel "axe des abscisses"
set ylabel "axe des ordonnées"
```


5.2 échelle logarithmique

```
set logscale x
set logscale y
```

5.3 grille

```
set grid
```

affiche une grille sur les traits majeurs.

Pour afficher aussi la grille mineure, il faut utiliser à la place

```
set mxtics 5
set mytics 5
set grid xtics ytics mxtics mytics linetype -1, 0
```

`set mxtics 5` signifie que la grille mineure horizontale va subdiviser un grand intervalle en 5 sous intervalles. Si l'intervalle est de 0 à 10, il y aura donc des traits mineurs pour 2, 4, 6 et 8.

`linetype -1, 0` signifie que la grille majeure sera affichée avec un type de ligne **-1** (ligne noire continue) et la grille mineure avec le type de ligne **0**, c'est à dire une ligne pointillée noire.



La virgule dans `linetype -1, 0` est la seule virgule de la ligne, et sépare les styles de ligne pour grille majeure et mineure. Il n'y a pas d'autre virgule dans la ligne (sauf autre option en nécessitant éventuellement).

5.4 Forcer un zoom

```
set xrange [0:5]
```

5.5 Style de courbe

La commande **test**, donnée à **gnuplot**, permet d'afficher un graphique avec les différentes possibilités comme le montre [FIGURE 2 on the following page]. Je n'ai pas réussi à récupérer le code source de la commande **test** par contre. Cette figure permet de voir les différentes possibilités, en particulier en matière de style de point ou de ligne. L'utilisation des paramètres pour contraindre le style d'affichage est détaillé dans les sous-sections suivantes.

Notez que cette figure a des rendus différents suivant le terminal choisi. Je n'ai pas réussi à avoir le résultat pour un terminal `.pdf` par exemple. Ou plutôt, impossible d'ouvrir le `.pdf` ensuite.

Il y a principalement deux styles d'affichage, la ligne, et les points. La différence majeure est que les points correspondent aux points des données, alors que la ligne (même pointillée) relie entre elle les points et ne se préoccupe pas de la densité de points. S'il y a très peu de points, vous verrez clairement les traits reliant les points mais sinon, vous pourrez avoir une idée très vague de l'échantillonnage, et surtout de la variation de celui-ci le long de la courbe.

5.5.1 Les lignes

Pour afficher une courbe à l'aide d'une ligne :

```
plot 'test_opacity.dat' using 1:2 with lines
```

La version compliquée est :

```
plot 'test_opacity.dat' using 1:2 with lines linetype 0 linewidth 3
```

où **linetype** permet de sélectionner la couleur de la ligne (`linetype 0` est un cas particulier pour avoir une ligne pointillée noire) et **linewidth** permet de spécifier la largeur de la ligne, dont la valeur n'est pas forcément entière (`linewidth 0.4` ou `linewidth 2.5` fonctionnent).

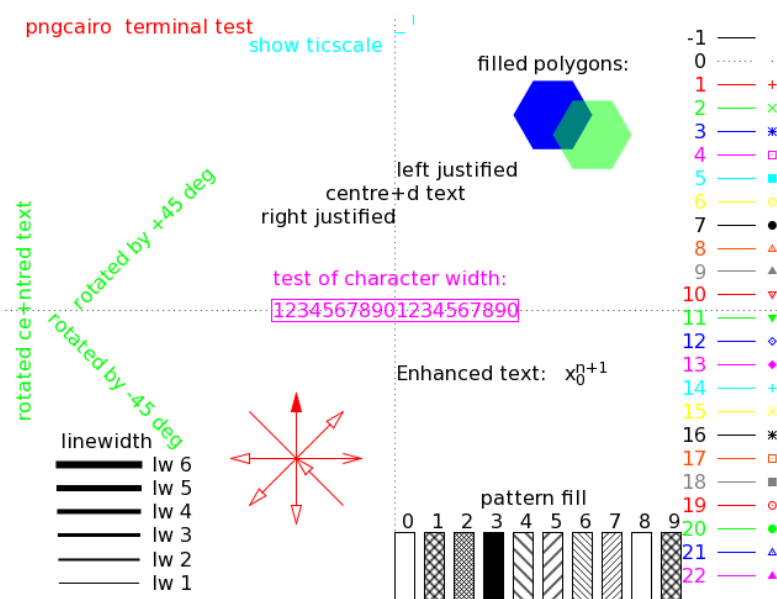


FIGURE 2 – Résultat de la commande `test` dans le cas du terminal **pngcairo** : affichage des différents styles et personnalisations graphiques offertes par gnuplot. Pour les styles de lignes, de points, types etc, le numéro pour avoir le style correspondant est à gauche, mais ce n'est pas la même commande selon qu'on veut un style de ligne (linestyle) ou de points (pointtype)

5.5.2 Les points

Pour afficher les points correspondant à une liste de données (sans les relier entre elles) :

```
plot 'test_opacity.dat' using 1:2 with points
```

Une version un peu plus complexe est :

```
plot 'test_opacity.dat' using 1:2 with points linestyle -1 pointtype 2 linewidth 3
```

linestyle permet de choisir la couleur (ici noir), **pointtype** permet de choisir le type de point (ici une croix) et **linewidth** la largeur des points.



Le type de point **pointtype -1** est un peu particulier puisque c'est un point, mais non affecté par la taille. On ne peut pas le grossir avec **linewidth**.

5.6 Texte avancé

Dans le terminal¹ courant, lors de sa définition, il faut rajouter l'option **enhanced**

```
set terminal x11 enhanced
```



Tous les terminaux ne supportent l'affichage de texte avancé (et donc l'option **enhanced**)

Ceci permet de faire du texte en indice, en exposant et d'accéder aux lettres grecques par exemple. On peut alors écrire dans le graphique $x_1 + y^2$ qui apparaîtra comme $x_1 + y^2$.

Remarque : Pas besoin de mettre en mode mathématique, il suffit de l'écrire comme ça.


Une différence importante a lieu pour l'affichage des lettres grecques :

1. Pour plus de détails sur la définition du terminal, voir [§ 4 on page 8]

ALPHABET	SYMBOL	ALPHABET	SYMBOL	alphabet	symbol	alphabet	symbol
A	A	N	N	a	α	n	ν
B	B	O	O	b	β	o	o
C	X	P	Π	c	χ	p	π
D	Δ	Q	Θ	d	δ	q	θ
E	E	R	P	e	ϵ	r	ρ
F	Φ	S	Σ	f	ϕ	s	σ
G	Γ	T	T	g	γ	t	τ
H	H	U	Y	h	η	u	v
I	I	V	ς	i	ι	v	ϖ
J	ϑ	W	Ω	j	φ	w	ω
K	K	X	Ξ	k	κ	x	ξ
L	Λ	Y	Ψ	l	λ	y	ψ
M	M	Z	Z	m	μ	z	ζ

Ainsi, pour faire la lettre minuscule α il faut faire :

`{/Symbol a}`

 Ce n'est pas un anti-slash comme les habitudes L^AT_EX pourraient nous le faire croire.

6 FAQ

6.1 pm3d map : l'image ne remplit pas le graphique

[FIGURE 3] montre une courbe où la carte ne remplit pas le graphique. J'ai mis du temps à trouver mais j'ai pu constater que c'était dû aux valeurs extrémales de mes données.

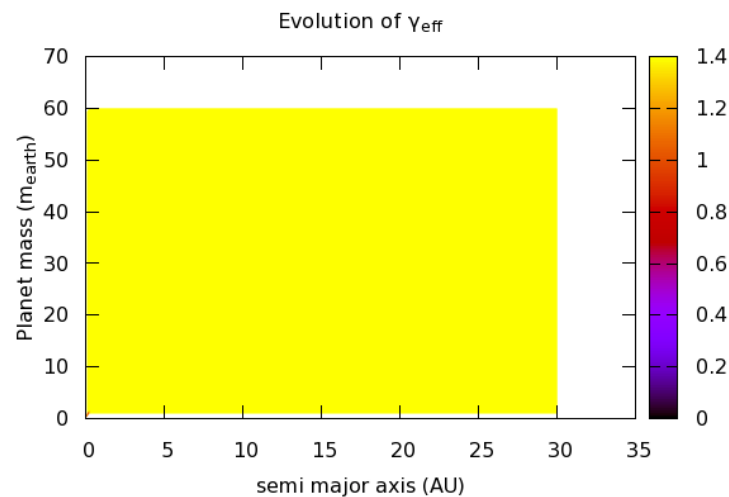


FIGURE 3 – Exemple d’affichage incorrect de données avec `pm3d map`. Il y a des blancs sur les bords du graphique que l’on ne souhaite pas.

```
0.161206    60.000000000000007    -0.65698873843863126
```

J’ai en effet `60.000000000000007` au lieu de `60`, c’est peu mais suffisant pour que gnuplot rajoute la partie `60 :70` sur le graphique.

Si vous utilisez

```
set yrange [0:60]
```

les données ayant $y = 60.000000000000007$ ne seront pas affichées. La solution que j'ai utilisé est de rajouter dans mon code l'écriture du **yrange** avec les valeurs du code.

J'obtiens alors :

```
set xrange [ 9.99999977648258209E-003 : 50.000000000000000 ]
set yrange [ 1.0000000000000000 : 60.000000000000007 ]
```

L'affichage est alors correct.

6.2 pm3d map : mauvais affichage de la première colonne

J'avais un affichage étrange de la première colonne de donnée, comme le montre [FIGURE 4a]. Ceci est dû aux données que je cherche à afficher.

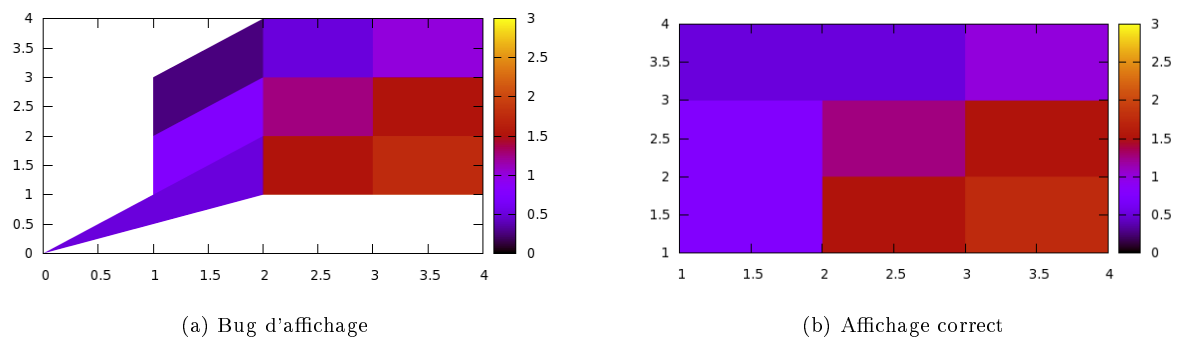


FIGURE 4 – Exemple d'affichage incorrect de la première colonne avec **pm3d map**

Il faut que les lignes qui contiennent des commentaires ou légendes commencent par le caractère #

Pour un mauvais affichage, j'ai

```
x y z
1 1 0
1 2 1
1 3 0
1 4 2

2 1 0
2 2 2
2 3 0
2 4 0

3 1 3
3 2 1
3 3 2
3 4 0

4 1 2
4 2 1
4 3 2
4 4 0
```

Pour un affichage correct il faut :

```
#x y z
1 1 0
```

```
1 2 1
1 3 0
1 4 2
```

```
2 1 0
2 2 2
2 3 0
2 4 0
```

```
3 1 3
3 2 1
3 3 2
3 4 0
```

```
4 1 2
4 2 1
4 3 2
4 4 0
```

Remarque : Les données étant affichées via :

```
set pm3d map
splot 'test.dat'
pause -1 # wait until a carriage return is hit
```

6.3 Invalid character \

```
splot 'test_total_torque.dat' with pm3d notitle, \
^
"torque_parameter_space.gnuplot", line 12: invalid character \
```

Quand j'ai eu ce problème, c'était dû à un espace en trop, juste après le caractère « \ ».