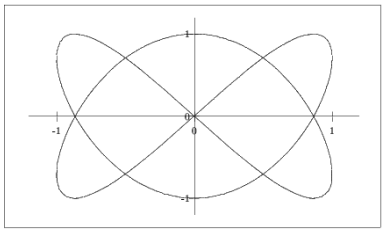
https://clx.asso.fr/spip/?Production-de-courbes-en

LaTeX

Production de courbes en environnement LaTeX

- Documentations - Education - Transition vers le Libre... -



Publication date: lundi 16 décembre 2002

Introduction

Pour produire des images avec LaTeX, il existe de nombreuses possibilités dont l'intégration dans le code source même de LaTeX. Mais souvent, on utilise des logiciels externes pour illustrer les documents. LaTeX nous impose un format unique d'image : l'EPS (Encapsulated PostScript). Cela peut sembler restrictif mais en définitive, cela permet d'unifier toutes les documents produits.
L'extension `graphicx` autorise avec la commande `\includegraphics` de manipuler les images au format EPS et metapost de manière assez aisée, permettant le redimensionnement absolu ou relatif, la rotation,
etc.
Néanmoins, il peut être intéressant de produire une figure dans le code source LaTeX ne serait-ce que pour y intégrer sans difficulté du code LaTeX et assurer ainsi une harmonie parfaite entre le texte et l'image.
Voici donc une petite présentation de quelques outils produisant des courbes de fonctions dans ou hors code source.

Intégration dans le code source

PsTricks

Le leader incontesté dans ce domaine est l'extension `pstricks`. Ses possibilités sont tellement étendues qu'il est impossible ici de citer tout ce qu'il peut faire. Son inconvénient majeur est le suivant : il produit du code PostScript donc non visualisable dans un afficheur dvi traditionnel.

PsTricks est un ensemble homogène d'extensions diverses. Ici l'extension appelée pour intégrer des courbes s'appelle `pst-plot` ; l'appel de cette extension charge automatiquement l'extension commune `pstricks`.

Nous définissons notre tracé dans un environnement `pspicture` analogue de l'environnement `picture` qui initialise, à chaque appel, les paramètres par défaut de `pstricks`.

Pour tracer des courbes, il nous faut rentrer une équation.
`Pstricks` utilise une syntaxe particulière (celle du PostScript tout simplement) qui ravira les amateurs de calculatrices HP : ce langage est basé sur une "pile" voici les syntaxes des opérations basiques :

Nom PS	Sign.	Utilisation	Résultat
add	+	2 3 add	5
div	/	6 2 div	3
mul	x	2 3 mul	6
sub	-	2 3 sub	-1

exp	2 3 exp	8

PostScript définit aussi les opérateurs :
[-] `neg` l'opérateur unaire de signe ;
[-] `abs` la valeur absolue ;
[-] `ln` et `log` les logarithmes respectivement néperien et à base 10 ;
[-] `sin` et `cos` désignent les sinus et cosinus d'un angle exprimé en degrés.

Notons que vous devrez calculer vous-même la valeur de la tangente en divisant le sinus par le cosinus, car elle est indéfinie. Pour en finir avec les lignes trigonométriques, remarquons l'opérateur binaire `atan` qui renvoie un angle en degrés (c'est-à-dire l'angle dont la tangente est le quotient du premier sur le second).

Notons aussi l'existence de `idiv` et `mod` qui désignent respectivement le quotient et le reste dans une division euclidienne. Des opérateurs d'arrondis ou de troncature sont définis mais leur utilisation dans notre cadre reste très limité.

N'oublions pas les commandes : `dup` qui copie le contenu de la pile, `exch` qui échange les lignes 1 et 2 de la pile, etc. Pour compléter ces informations, consultez la liste des opérateurs PostScript.

Pour finir, voici quelques tracés de courbes :
[-] exemple 1 : une courbe paramétrée

```
\begin{pspicture}(7.2cm,7.2cm)
 \rput(3.6,3.6){
 % le repère
 \psset{xunit=3cm,yunit=3cm}
 \psaxes{->}(0,0)(-1.2,-1.2)(1.2,1.2)
 \rput(0.05,-0.1){$0$}

 % le tracé
 \parametricplot[plotpoints=200,linecolor=red]{0}{360}{t 2 mul sin 3 t mul cos}

 % la légende
 \rput(.3,-.7){Courbe de Lissajous%
 $\left\{\begin{array}{ll}x(t)=\sin(2t)\\y(t)=\cos(3t)\end{array}\right.$%
 }
 }
 \end{pspicture}
```

PNG - 10.3Â ko

Nous définissons les unités pour la figure avec la commandes `\psset` et les mots-clés `xunit` et `yunit`. Notre courbe de Lissajous étant incluse dans un carré de côté 2 centré sur l'origine, on prendra les valeurs de x dans l'intervalle [-1,2 ;1,2].

La commande `\parametricplot` permet de dessiner une courbe paramétrée. Ici, j'ai choisi de la tracer avec 200 points (on aurait pu inclure cette option dans le `\psset` et ce serait devenu une option globale des tracés), le paramètre t variant de 0 à 360 degrés.

On remarquera l'absence de séparation entre `x(t)` et `y(t)` dans leur définition, le travail par pile produira ici deux équations.

Pour conclure, le point origine du repère est normalement le coin inférieur gauche, ce qui aurait centré notre figure sur ce point-là, sans l'intervention d'une petite image par translation des tracés grâce à la commande `\rput` ; cette même commande est utilisée pour insérer un texte. Ce dernier est centré, par défaut, sur le point dont les coordonnées sont données en arguments.

[-] exemple 2 : une courbe de fonction

Prenons une fonction trigonométrique : x|â€">sin(x)/x.

```
\begin{pspicture}(7cm,8cm)
 \rput(3.5,2.5){
 % le repère
 \psset{xunit=.25cm,yunit=4cm,plotpoints=200}
 \psaxes[Dx=3,Dy=.5]{->}(0,0)(-13,-.6)(13,1.3)
 \rput(-1,-0.05){$0$}

 % le tracé
 \psplot[linewidth=.1mm]{-13}{13}{x 180 mul 3.14159 div sin x div}
 }
 \rput[l](0,7){$\displaystyle x\longmapsto \frac{\sin x}{x}$}
 \end{pspicture}
```

PNG - 5.9Â ko

Ici, l'enchaînement est le même, avec la conversion en radians pour la fonction sinus. Bien sûr le dénominateur lui ne sera pas converti.

[-] exemple 3 : une illustration de cours

Une petite illustration des fonctions de référence (non trigonométriques) sur l'intervalle [-1 ;1].

```
\begin{pspicture}(10cm,8cm)
 \rput(3.6,4){
 % le repère
 \psset{xunit=3cm,yunit=3cm}
 \psaxes{->}(0,0)(-1.2,-1.2)(1.2,1.2)               % les axes (origine) (coin_inf_gauche) (coin_sup_droit)
 \rput(0.05,-0.1){$0$}

 % les tracés
 \psplot[linestyle=dotted]{-1}{1}{x}                      % fonction identité
 \psplot[doubleline=true]{-1}{1}{x 2 exp}        % fonction carré
 \psplot[linestyle=dashed]{-1}{1}{x 3 exp}        % fonction cube
 \psplot[linecolor=red]{0}{1}{x sqrt}               % fonction racine carrée
```

```
% la légende
\psline[linestyle=dotted]{-}(1.3,1)(1.5,1)\rput[l](1.6,1){$x\longmapsto x$}
\psline[doubleline=true]{-}(1.3,0.9)(1.5,0.9)\rput[l](1.6,0.9){$x\longmapsto x^2$}
\psline[linestyle=dashed]{-}(1.3,0.8)(1.5,0.8)\rput[l](1.6,0.8){$x\longmapsto x^3$}
\psline[linecolor=red]{-}(1.3,0.7)(1.5,0.7)\rput[l](1.6,0.7){$x\longmapsto \sqrt{x}$}
}
\end{pspicture}
```

PNG - 8.7Â ko

Contrairement aux exemples identiques traités dans la suite de l'article avec l'extension `mfpic` et `metapost`, j'ai tracé la courbe de la fonction carré avec un double trait, option directement accessible en paramètre qui, à ma connaissance n'existe pas avec les autres mais qui pourrait faire sans doute l'objet d'une macro.

mfpic

`Mfpic` est une sorte de `metapost` embarqué, il permet d'intégrer en ligne du code `metafont` et définit lui-même des commandes pour tracer des courbes. Cela peut sembler génialement intéressant mais la procédure de mise en place n'est pas des plus faciles ; en effet, à partir du code `mfpic` que vous saisissez, la compilation LaTeX produit le fichier `metafont` correspondant. Il faut donc compiler ce nouveau fichier avec `metafont` de telle sorte que soit produit un fichier de métriques de police tfm, puis compiler le fichier gf produit afin de créer l'image de la police elle-même : le fichier pk. Votre image est donc intégrée comme une police dans le document (ce qui ne nous oblige donc pas de visionner un fichier ps car Le fichier dvi intégrera votre image).

Pour utiliser `mfpic`, je l'ai téléchargé depuis Internet sur `ctan`, j'ai créé un répertoire `mfpic` dans `/usr/share/texmf/tex/latex` où j'y ai placé les fichiers sty et tex. J'ai ensuite copié le fichier `grafbase.mf` dans le répertoire `/usr/share/texmf/metafont/misc` car ces fichiers sont nécessaires à la compilation par metafont.

Notons que nous pouvons définir à l'instar de la variable d'environnement `TEXINPUTS`, les variables `MFINPUTS` et `MPINPUTS`. Elles permettent aux compilateurs associées de `metafont` et de `metapost` de trouver les fichiers précédement cités dans les bons répertoires.

Pour finir, voici une session :

[-] première compilation LaTeX

```
bash-2.05$ latex essai_mfpic.tex
This is TeX, Version 3.14159 (Web2C 7.3.1)
(essai_mfpic.tex

Babel <v3.7h> and hyphenation patterns for american, french, nohyphenation, loa
ded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2000/05/19 v1.4b Standard LaTeX document class
```

```
 (/usr/share/texmf/tex/latex/base/size10.clo))
 (/usr/share/texmf/tex/latex/mfpic/mfpic.sty
 (/usr/share/texmf/tex/latex/mfpic/mfpic.tex))
 No file essai_mfpic.aux.

 Underfull \hbox (badness 10000) in paragraph at lines 5--6


 MFpic: No file essai_mfpic.tfm .
 MFpic: Don't forget to process essai_mfpic.mf!
 MFpic: (Apply metafont to essai_mfpic.mf, then gftopk to the resulting gf file.
 )
 MFpic: Then reprocess this file (essai_mfpic).

 [1] (essai_mfpic.aux) )
 (see the transcript file for additional information)
 Output written on essai_mfpic.dvi (1 page, 432 bytes).
 Transcript written on essai_mfpic.log.
```

[-] compilation metafont

```
bash-2.05$ mf essai_mfpic.mf
 This is METAFONT, Version 2.7182 (Web2C 7.3.1)
 (essai_mfpic.mf (/usr/share/texmf/metafont/misc/grafbase.mf
 MFpic version 0.5.0 beta, 2001/10/30
 ) [1] )
 Font metrics written on essai_mfpic.tfm.
 Output written on essai_mfpic.360gf (1 character, 1784 bytes).
 Transcript written on essai_mfpic.log.
```

[-] création de la police (pour ceux qui s'étonne de la résolution employée j'ai encore une vieille mais très avantageuse Canon BJ-10ex... Bon, il est vrai qu'utiliser LaTeX avec une définition aussi faible est un crime de lèse-majesté).

```
bash-2.05$ gftopk essai_mfpic.360gf
```

[-] deuxième compilation LaTeX

```
bash-2.05$ latex essai_mfpic.tex
 This is TeX, Version 3.14159 (Web2C 7.3.1)
 (essai_mfpic.tex

 Babel <v3.7h> and hyphenation patterns for american, french, nohyphenation, loa
 ded.
 (/usr/share/texmf/tex/latex/base/article.cls
 Document Class: article 2000/05/19 v1.4b Standard LaTeX document class
 (/usr/share/texmf/tex/latex/base/size10.clo))
 (/usr/share/texmf/tex/latex/mfpic/mfpic.sty
 (/usr/share/texmf/tex/latex/mfpic/mfpic.tex)) (essai_mfpic.aux)
 Underfull \hbox (badness 10000) in paragraph at lines 5--6


 MFpic: Don't forget to process essai_mfpic.mf!
 MFpic: (Apply metafont to essai_mfpic.mf, then gftopk to the resulting gf file.
```

```
 )
 MFpic: Then reprocess this file (essai_mfpic).

 [1] (essai_mfpic.aux) )
 (see the transcript file for additional information)
 Output written on essai_mfpic.dvi (1 page, 372 bytes).
 Transcript written on essai_mfpic.log.
```

[-] visualisation du binaire

```
bash-2.05$ xdvi essai_mfpic.dvi
```

OUF ! on pourra aisément automatiser ces tâches pour faciliter l'utilisation de cette extension.
Maintenant regardons un peu sur les mêmes exemples que précédemment ce que produit l'extension.
Voici le document source pour les trois exemples précédents avec `mfpic`.

```
\documentclass{article}
 \usepackage{mfpic}

 \begin{document}

 \opengraphsfile{essai_mfpic}
 \mfpicunit=1cm
 \centering

 \fbox{
 \begin{mfpic}[3][3]{-1.2}{1.2}{-1.2}{1.2}
 % le repère
 \axes
 \xmarks{-1,1}
 \ymarks{-1,1}
 \tlabel[tl](0.03,-0.03){O}
 \tlabel[tc](-1,-0.03){$-1$}\tlabel[tc](1,-0.03){1}
 \tlabel[cl](0.03,-1){$-1$}\tlabel[cl](0.03,1){1}
 % le tracé
 \parafcn{0,360,1}{(sind 2t,cosd 3t)}
 % la légende
 \tlabel[cc](.3,-.7){Courbe de Lissajous%
 $\left\{\begin{array}{ll}x(t)=\sin(2t)\\y(t)=\cos(3t)\end{array}\right.$%
 }
 \end{mfpic}
 }\\[.5cm]
 \fbox{
 \begin{mfpic}[.25][4]{-13}{13}{-.6}{1.3}
 % le repère
 \axes
 \xmarks{-12,-9,-6,-3,0,3,6,9,12}
 \ymarks{-.5,.5,1}
 \tlabel[tl](0.5,-0.03){O}
 \tlabel[tc](-12,-0.03){$-12$}\tlabel[tc](12,-0.03){12}
 \tlabel[tc](-9,-0.03){$-9$}\tlabel[tc](9,-0.03){9}
 \tlabel[tc](-6,-0.03){$-6$}\tlabel[tc](6,-0.03){6}
 \tlabel[tc](-3,-0.03){$-3$}\tlabel[tc](3,-0.03){3}
```

```
\tlabel[cl](0.5,-.5){$-0,5$}\tlabel[cl](0.5,1){1}
\tlabel[cl](0.5,0,5){$0,5$}

% le tracé
\function{-13,13,.13}{sind(x*180/3.14159)/x}

% la légende
\tlabel[tl](-13,1.1){$\displaystyle x\longmapsto \frac{\sin x}{x}$}
\end{mfpic}
}

\fbox{
\begin{mfpic}[3][3]{-1.2}{1.2}{-1.2}{1.2}
% le repère
\axes
\xmarks{-1,0,1}
\ymarks{-1,1}
\tlabel[tl](0.03,-0.03){O}
\tlabel[tc](-1,-0.03){$-1$}\tlabel[tc](1,-0.03){1}
\tlabel[cl](0.03,-1){$-1$}\tlabel[cl](0.03,1){1}

% les tracés
\dotted\function{-1,1,.1}{x}
\pen{1bp}\function{-1,1,.1}{x**2}
\pen{.5pt}
\dashed\function{-1,1,.1}{x**3}
\function{0,1,.1}{sqrt(x)}

% la légende
\dotted\lines{(1.3,1),(1.5,1)}\tlabel[l](1.6,1){$x\longmapsto x$}
\pen{1bp}\lines{(1.3,0.9),(1.5,0.9)}\tlabel[l](1.6,.9){$x\longmapsto x^2$}
\pen{.5pt}
\dashed\lines{(1.3,0.8),(1.5,0.8)}\tlabel[l](1.6,.8){$x\longmapsto x^3$}
\lines{(1.3,0.7),(1.5,0.7)}\tlabel[l](1.6,.7){$x\longmapsto \sqrt{x}$}
\end{mfpic}
}
\closegraphsfile

\end{document}
```

PNG - 6.4Â ko PNG - 4.5Â ko PNG - 4.5Â ko

Dès le départ, un fichier que j'ai nommé `essai_mfpic` est ouvert en écriture, c'est celui-ci qui va contenir après la première compilation le code `metafont` à compiler. Il est naturellement fermé à la fin du source LaTeX.
On remarque l'absence cruelle de couleur, mais `mfpic` dispose d'une commande `\usemetapost` pour produire non pas du code `metafont` mais du code `metapost` qui sera donc à compiler avec ce dernier et non plus `metafont`.

Bien entendu, on aura accès à plus de paramètres et notamment à la couleur, mais les figures produites ne seront visualisables qu'avec un visualiseur acceptant le format PostScript.

Le positionnement des étiquettes peut se faire à l'aide de 2 paramètres qui définissent respectivement la position verticale (b pour bottom (bas), c pour center (centré), l pour left (gauche) et r pour right (droite)) et la position horizontale (définie de la même manière).

La commande `function` prend trois paramètres : l'abscisse de départ, l'abscisse d'arrivée et le pas pour parcourir cet intervalle. Le reste est très compréhensible.

Mon impression

[-] `PsTricks` est vraiment un outil très puissant, jetez un coup d'oeil à la documentation et vous serez ébloui par toutes ces figures et ces graphiques. La nécessité de devoir visualiser en PostScript n'est pas très gênante pour moi car je ne visualise jamais mon dvi : j'emploie toujours la commande :

```
dvips monfichier.dvi -o
```

pour finalement visualiser avec `gv`.

[-] L'extension `mfpic` est remarquable. Dessiner une courbe de fonction est d'une simplicité inégalable. En revanche, je déteste la complexité (certes relative) de la compilation. J'avoue préférer tracer directement avec `metapost`. Ce n'est alors plus intégré dans le source LaTeX mais plutôt que d'apprendre des commandes pour construire un fichier `metafont` ou `metapost` autant apprendre le langage lui-même...

Intégration hors code source

L'intégration hors code source se fait avec la commande `\includegraphics` de l'extension `graphicx`, par exemple :

```
\includegraphics{monfichier.eps}
```

metapost

Je vous propose tout simplement de reprendre les exemples précédents en créant trois figures. Ici j'ai utilisé du code LaTeX pour mes étiquettes dans `metapost` alors que par défaut, `metapost` utilise TeX pour compiler... J'ai donc attribué la valeur latex à la variable d'environnement `TEX`.

```
export TEX=latex
```

Ensuite dans mon code source `metapost`, j'ai inséré au début et à la fin les déclarations habituelles des sources latex pour que les étiquettes soient correctement compilées par LaTeX.

Pour finir, les étiquettes utilisant LaTeX voit leur contenu (dans le source) encadré par les marqueurs `btex` et `etex`.

```
verbatimtex
 \documentclass{article}
 \begin{document}
```

```
etex

beginfig(1);
path axe,taquet,courbe;
u=3cm;

% les axes
axe=(-1.2u,0)--(1.2u,0);
drawarrow axe;
drawarrow axe rotated 90;

% les marques
for i=-1,1 :
draw (i*u,-.5mm)--(i*u,.5mm);
label.bot(decimal(i),(i*u,-1mm));
draw (-.5mm,i*u)--(.5mm,i*u);
label.lft(decimal(i),(0,i*u));
endfor
label.llft("O",(-.5mm,-.5mm));

% le calcul de la courbe
k=0;
for i=1 upto 200 :
% on divise en 200 valeurs les valeurs de 0 à 360
monangle:=(i-1)*360/199;
x:=sind(2*monangle);y:=cosd(3*monangle);
% si ce n'est pas le premier point on ajoute au chemin précédent
if i=1: courbe=(x*u,y*u) else: courbe:=courbe--(x*u,y*u) fi;
endfor

% le tracé de la courbe
draw courbe withcolor red;

% l'étiquette
label(btex Courbe de Lissajous%
$\left\{\begin{array}{ll}x(t)=\sin(2t)\\y(t)=\cos(3t)\end{array}\right.$%
etex,(.3u,-.7u));

endfig;

beginfig(2);
path axe[],courbe;
ux=.25cm;uy=4cm;

% les axes
axe0=(-13ux,0)--(13ux,0);
axe1=(0,-.6uy)--(0,1.3uy);
drawarrow axe0;
drawarrow axe1;

% les marques
for i=-12 step 2 until 12 :
```

```
draw (i*ux,-.5mm)--(i*ux,.5mm);
if i<>0 : label.bot(decimal(i),(i*ux,-1mm)) else : label.llft(btex O etex,origin) fi;
endfor
for i=-0.5,0.5,1 :
draw (-.5mm,i*uy)--(.5mm,i*uy);
label.lft(decimal(i),(0,i*uy));
endfor

% le calcul de la courbe
for i=1 upto 200 :
% on divise en 200 valeurs les valeurs de --13 à 13
x:=-13+(i-1)*26/199;
y:=sind(x*180/3.14159)/x;
% si ce n'est pas le premier point on ajoute au chemin précédent
if i=1: courbe=(x*ux,y*uy) else: courbe:=courbe--(x*ux,y*uy) fi;
endfor

% le tracé de la courbe
draw courbe withpen pencircle scaled .1mm;

% l'étiquette
label.lrt(btex $\displaystyle x\longmapsto {\sin x \over x}$ etex,(-13ux,1.3uy));
endfig;

beginfig(3);
path axe,taquet,courbe[],leg;
u:=3cm;

% les axes
axe=(-1.2u,0)--(1.2u,0);
drawarrow axe;
drawarrow axe rotated 90;

% les marques
for i=-1,1 :
draw (i*u,-.5mm)--(i*u,.5mm);
label.bot(decimal(i),(i*u,-1mm));
draw (-.5mm,i*u)--(.5mm,i*u);
label.lft(decimal(i),(0,i*u));
endfor
label.llft("O",(-.5mm,-.5mm));

% le calcul des courbes
for i=1 upto 200 :
% on divise en 200 valeurs les valeurs de -1 à 1
x:=-1+(i-1)*2/199;
for j=1 upto 3 :
y:=x**j;
if i=1: courbe[j]=(x*u,y*u) else: courbe[j]:=courbe[j]--(x*u,y*u) fi;
endfor;
if i=101: courbe4=(x*u,sqrt(x)*u) fi;
if i>100: courbe4:=courbe4--(x*u,sqrt(x)*u) fi;
```

```
endfor

% le tracé de la courbe
draw courbe[1] dashed withdots;
draw courbe[2] withpen pencircle scaled 1bp;
draw courbe[3] dashed evenly;
draw courbe[4] withcolor red;

% la légende
leg=(1.3u,u)--(1.5u,u);
label.rt(btex $x\longmapsto x$ etex,(1.6u,u));
draw leg dashed withdots;
label.rt(btex $x\longmapsto x^2$ etex,(1.6u,.9u));
draw leg shifted (0,-.1u) withpen pencircle scaled 1bp;;
label.rt(btex $x\longmapsto x^3$ etex,(1.6u,.8u));
draw leg shifted (0,-.2u) dashed evenly;
label.rt(btex $x\longmapsto \sqrt x$ etex,(1.6u,.7u));
draw leg shifted (0,-.3u) withcolor red;
endfig;

verbatimtex
\end{document}
etex

end
```

PNG - 8.7Â ko PNG - 6Â ko PNG - 6.6Â ko

La grande différence avec les solutions précédemment proposées est qu'ici, vous devez TOUT faire, au risque de rebuter les adeptes de la facilité, mais l'avantage est de pouvoir maîtriser beaucoup plus de choses.

On notera deux aspects intéressants dans le code :
[-] l'inclusion de code LaTeX dans les étiquettes (cf. remarque du début) ;
[-] l'utilisation d'un test systématique pour établir nos courbes. En effet, celles-ci sont déclarées comme étant de type `path`, si nous essayons de faire sans, nous coderions tout simplement, quelque chose du genre :

```
courbe:=courbe--(x*ux,y*uy);
```

Mais voilà au premier appel la variable courbe n'a pas de valeur définie. Donc pour commencer la courbe, on teste à l'aide d'un compteur s'il s'agit du premier point calculé, auquel cas on attribue à la variable courbe cette valeur de type `pair`, le code devient donc :

```
if i=1: courbe=(x*ux,y*uy) else: courbe:=courbe--(x*ux,y*uy) fi;
```

où i est le compteur dans la boucle de calcul.

gnuplot

Connaissant fort mal gnuplot, je ne vous présente ici que quelques aspects bien simples de ce logiciel.

gnuplot est un logiciel de tracés de courbes scientifiques. Il permet en mode interactif (*via* la ligne de commande) de définir le style des courbes, mais aussi de préciser le type de sortie voulu : sortie écran, sortie fichier au format PNG ou autre, sortie en code LaTeX (environnement picture), sortie pstricks (environnement pspicture), etc.

Le mode interactif est certes intéressant, mais on préférera sans doute le passage par un fichier. Par exemple, j'ai constitué, pour la première courbe, un fichier de commandes gnuplot que j'ai appelé 'figure1.gnu'. Pour évaluer son contenu, je vais appeler la commande gnuplot puis à l'invite interactive, je vais saisir la commande :

```
load 'figure1.gnu'
```

Pour paramétrer la sortie, il faut changer la valeur de la variable `term`. Par défaut celle-ci est fixée à `X11`, c'est-à-dire que les tracés se feront dans une fenêtre de X11 que gnuplot affichera. Pour générer une sortie différente, par exemple une sortie LaTeX, on écrira :

```
set term latex
```

Dans ce cas, la sortie (en code LaTeX) se fera sur la sortie standard, donc sans doute la console elle-même. Pour diriger ce flux vers un fichier, il suffit de changer la valeur de la variable `outout`, par exemple :

```
set output figure1.tex
```

Voilà l'ensemble des sorties possibles :

aed512, aed767, aifm, bitgraph, cgm, corel, dumb, dxf,
eepic, emtex, epson-180dpi, epson-60dpi,
epson-lx800, fig, gif, gpic, hp2623a, hp2648, hp500c, hpdj,
hpgl, hpljii, hppj, imagen,
kc-tek40xx, km-tek40xx, latex, mf,
mif, mp, nec-cp6, okidata,t
pbm, pcl5, png, postscript,
pslatex, pstex, pstricks, qms,
regis, selanar, starc, table,
tandy-60dpi, tek410x, texdraw, tgif,
tkcanvas, tpic, vttek, x11,
xlib.

On y trouvera notamment emtex, latex, mf
mp, pslatex, pstex, pstricks, tex et texdraw.

Pour produire les images de cet article, j'ai demandé une sortie png. Bien entendu, dans les exemples, il s'agit de les inclure dans un document LaTeX. J'ai donc demandé une sortie latex (fichier qui porte l'extension .tex). Il suffira par la suite d'intégrer le
fichier dans le code source LaTeX en utilisant :

```
\input figure1.tex
```

[-] exemple 1

```
set term latex
```

```
set output 'figure1.tex'
set parametric
set noborder
set zeroaxis
set xtics axis 1
set ytics axis 1
set xrange [-1.2:1.2]
set yrange [-1.2:1.2]
set samples 200
plot [0:2*pi] sin(2*t),cos(3*t) notitle
```

PNG - 12.2Â ko

[-] exemple 2

```
set term latex
set output 'figure2.tex'
set noborder
set zeroaxis
set xtics axis 3
set ytics axis .5
set xrange [-13:13]
set yrange [-.6:1.3]
set samples 200
plot sin(x)/x notitle lw 1
```

PNG - 7.4Â ko

[-] exemple 3

```
set term latex
set output 'figure3.tex'
set noborder
set zeroaxis
set xtics axis 1
set ytics axis 1
set xrange [-1.2:1.2]
set yrange [-1.2:1.2]
set samples 200
plot [-1:1] x notitle, x**2 notitle,x**3 notitle,sqrt(x) notitle
```

PNG - 7.4Â ko

Je ne développe pas les possibilités d'intégration de code LaTeX dans les images. Je ne les connaîs pas mais je sais qu'elles existent.

Les autres

Gnuplot possède aussi des capacités de tracés de surfaces et courbes en dimension 3. Pour les allergiques à gnuplot, il existe quelques front-ends sur Internet.

Pour finir, on trouvera sur Internet nombre de traceurs de courbes qui exportent en EPS.

Netographie

[-] pstricks ;
[-] mfpic ;
[-] metapost ;
[-] gnuplot.