

J.-F. ABADIE

Jeudi 15 mars 2018

Avant-propos

Ces quelques notes rassemblent la plupart des fonctionnalités de **PSTricks** présentées, ou qui du moins étaient censées l'être, lors de l'exposé d'infomath du jeudi 15 mars 2018. Bien qu'elles ne prétendent pas à l'exhaustivité ou l'originalité, elles faciliteront peut-être la prise en main de ce package à toute personne désirant s'essayer au dessin sous \LaTeX . Tel était en tout cas mon souhait lors de la préparation de cet exposé et de ce document qui l'accompagne.

Le texte qui suit ayant été tapé dans la précipitation, il comporte très probablement quelques imprécisions, incohérences, omissions, voire même erreurs. Même si tout a été fait pour en limiter le nombre, je tiens par avance à m'en excuser, et remercie en outre toute personne qui prendra la peine de me les signaler.

Table des matières

Introduction	2
1 Le package et son environnement	2
2 Figures de base	3
3 Paramètres généraux	4
4 Texte sous PSTricks	7
5 Quadrillages et axes	9
6 Tracé de courbes	10
7 Pour aller plus loin, ou ailleurs...	12
Références	13

Introduction

Que gagne-t-on à utiliser **PSTricks** lors de réalisation d'un quelconque document sous \LaTeX ? Pas mal d'ennuis pourraient avancer celles et ceux qui s'y sont risqués :

- La prise en main de **PSTricks** n'est pas immédiate, et son utilisation est parfois casse-tête.
- La réalisation de dessins sous **PSTricks** ne permet *a priori* plus la compilation **pdf \LaTeX** .

Mais en y mettant un soupçon de bonne volonté, les mêmes le reconnaîtraient :

- Son utilisation permet de réaliser des illustrations de qualité (vectorielle), et qui s'accordent avec le reste du document (utilisation des mêmes polices, références, etc.).
- Il s'agit d'un package extrêmement fourni, disposant de nombreuses extensions. À cet effet, il permet presque à coup sûr la réalisation des dessins que l'on pourrait souhaiter exécuter.

C'est donc conscient de ces atouts de taille mais des difficultés qui l'attendent que le lecteur pourra envisager plus sereinement de s'essayer à **PSTricks**. S'il y consent, nous nous proposons de l'accompagner dans cette entreprise. Qu'il se prépare sans plus attendre ; l'initiation commence dès à présent !

Insistons tout d'abord sur le caractère *introductif* des propos qui suivent : la documentation de **PSTricks** compte 338 pages, et ce package dispose par ailleurs de nombreuses extensions largement fournies. Toute présentation concise et exhaustive à son sujet se révélerait donc absurdement prétentieuse.

Dans ce document, nous commencerons par présenter le package et son environnement (section 1), quelques figures de base (section 2) et quelques paramètres généraux de **PSTricks** (section 3). Nous verrons ensuite comment placer du texte sur un dessin (section 4) ou comment doter ceux-ci d'une grille ou d'un repère (section 5), puis nous nous intéresserons au tracé de courbes en dimension 2 (section 6). Pour finir, nous listerons sans les détailler quelques-unes des (multiples) autres fonctionnalités offertes par **PSTricks** et ses extensions, et nous mentionnerons d'autres procédés alternatifs permettant le dessin avec \LaTeX (section 7).

1 Le package et son environnement

1.1 — Le package **PSTricks** se charge via la commande `\usepackage{pstricks}` ; précisons que celui-ci appelle automatiquement **xcolor**, package permettant la gestion des couleurs sous \LaTeX . Sauf indication contraire, l'intégralité des figures présentées dans ce document sont reproductibles à partir d'un fichier source **.tex** constitué comme suit :

```
\documentclass{article}
\usepackage{pstricks}
\begin{document}

\end{document}
```

1.2 — Attention ! la compilation **pdf \LaTeX** n'est *a priori* pas compatible avec l'utilisation des fonctionnalités de **PSTricks**¹. Le recours à ces dernières nécessite de suivre le schéma de compilation suivant :

latex → **dvips** → **ps2pdf**

En particulier, la compilation **latex** tolère seulement l'inclusion de fichiers PostScript (extensions **.ps** ou **.eps**). Pour inclure des fichiers au format **.jpg**, **.png** ou **.pdf** (pour ne citer qu'eux), une conversion préalable au format PostScript sera donc nécessaire.

1.3 — Pour réaliser un dessin sous **PSTricks**, nous recommandons l'utilisation de l'environnement **pspicture**. Il se définit comme suit :

```
\begin{pspicture}(x_-,y_-)(x_+,y_+)

\end{pspicture}
```



Ceci crée, là où la commande a été déclarée, un domaine rectangulaire s'étendant des points de coordonnées (x_-, y_-) à (x_+, y_+) destiné à accueillir le dessin que nous souhaitons exécuter. Pour des raisons esthétiques,

1. Il existe cependant des moyens de pallier cette difficulté, mais nous ne les évoquerons pas ici.

nous conseillons de séparer texte et dessins au sein du document réalisé; nous suggérons à cet effet de placer les environnements `pspicture` en mode centré :

```
\begin{center}
\begin{pspicture}(x_-,y_-)(x_+,y_+)

\end{pspicture}
\end{center}
```

Signalons enfin que l'environnement `pspicture` s'intègre parfaitement à l'environnement `figure`, qui permet de titrer et référencer les figures sous \LaTeX :

```
\begin{figure}
\begin{pspicture}(x_-,y_-)(x_+,y_+)

\end{pspicture}
\end{figure}
```

REMARQUE. L'environnement `pspicture` permet donc de *délimiter* une surface sur laquelle le dessin sera réalisé. Cela dit, nous n'en restons pas moins libre de dessiner au-delà de la zone définie par cet environnement, ou même en dehors d'un tel environnement.

2 Figures de base

2.1 — Dans l'intégralité des sections 2 et 4 à 6, les crochets ou accolades laissés vides (`[]` ou `{}`) ont pour seul but d'indiquer où les paramètres optionnels associés aux commandes présentées pourront être renseignés; leur saisie s'avère donc facultative. Par ailleurs, n désignera ici un entier supérieur ou égal à 1.

2.2 — *Points.*

Pour créer n points ayant pour coordonnées $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, on saisit :

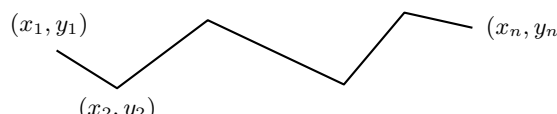
```
| \psdots[] (x_1, y_1) (x_2, y_2) \dots (x_n, y_n)
```



2.3 — *Segments et lignes brisés.*

Pour tracer une ligne brisée passant successivement par les points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, on saisit :

```
| \psline[]{} (x_1, y_1) (x_2, y_2) \dots (x_n, y_n)
```



Bien entendu, on obtient un segment en se limitant au cas où $n = 2$.

2.4 — *Rectangles.*

Pour construire un rectangle dont l'une des diagonale a pour extrémités (x_-, y_-) et (x_+, y_+) , on saisit :

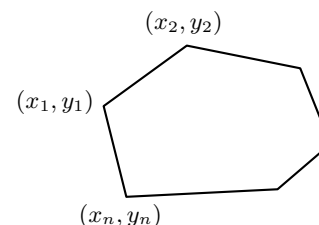
```
| \psframe[] (x_-, y_-) (x_+, y_+)
```



2.5 — *Polygones.*

Pour construire un polygone ayant pour sommets successifs $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, on saisit :

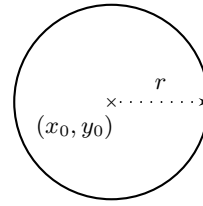
```
| \pspolygon[] (x_1, y_1) (x_2, y_2) \dots (x_n, y_n)
```



2.6 — Cercles.

Pour tracer un cercle de centre (x_0, y_0) et de rayon $r > 0$, on saisit :

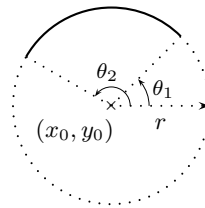
`\pscircle[] (x_0, y_0){r}`



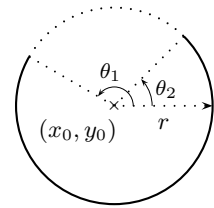
2.7 — Arcs de cercle.

Soient θ_1 et θ_2 deux angles exprimés en *degrés*. Pour tracer un arc de cercle γ centré en (x_0, y_0) , de rayon r , et reliant les points de coordonnées $(r \cos \theta_1, r \sin \theta_1)$ et $(r \cos \theta_2, r \sin \theta_2)$, on pourra saisir :

`\psarc[] {} (x_0, y_0){r}{\theta_1}{\theta_2}`



Cas où $\theta_1 \leq \theta_2$

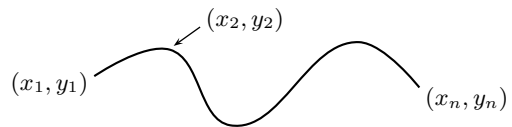


Cas où $\theta_2 > \theta_1$

2.8 — Courbes d'interpolation.

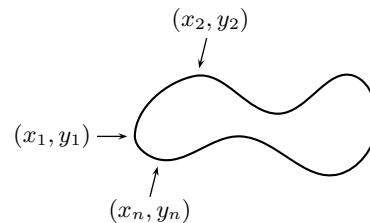
Pour créer une courbe d'interpolation² passant par les points (x_1, y_1) , (x_2, y_2) , \dots , (x_n, y_n) , on saisit :

`\pscurve[] {} (x_1, y_1) (x_2, y_2) \dots (x_n, y_n)`



Si l'on souhaite obtenir une courbe fermée, on saisira plutôt³ :

`\psccurve[] (x_1, y_1) (x_2, y_2) \dots (x_n, y_n)`



3 Paramètres généraux

3.1 — Nous indiquons à présent comment modifier les paramètres par défaut des figures **PSTricks**. Commençons avant tout par un préciser point de vocabulaire; sous **PSTricks**, on distinguera deux catégories de figures :

- Les figures *fermées*, c'est-à-dire celles qui délimitent un domaine clos du plan; les rectangles, polygones ou cercles ou courbes fermées, rencontrés en section 2, rentrent donc dans cette catégorie.
- Par opposition, toute autre figure sera qualifiée d'*ouverte*; points, segments et lignes brisées, arcs de cercles et courbes non fermées font partie de cette seconde catégorie.

Dans cette section, nous verrons essentiellement comment :

- modifier les unités graphiques par défaut,
- (re)définir le style, l'épaisseur ou la taille, la couleur, et l'affichage des traits ou points d'une figure,
- (re)définir le remplissage une figure fermée,
- ajouter des extrémités à une figure ouverte.

2. La courbe obtenue serait par ce biais de Bézier par morceaux.

3. On rajoute donc un « c », pour signifier « closed ».

3.2 — À l'exception paramètres définissant les extrémités des figures ouvertes, qui eux se renseignent entre accolades {}, la modification des paramètres d'une quelconque figure se commande entre crochets []⁴, ou plus généralement par utilisation de la commande **psset** présentée au paragraphe 3.12.

Dans ce qui suit, nous appellerons *clé* le paramètre **PSTricks** que l'on souhaite ajuster, et *valeur* la manière dont on souhaite qu'il le soit. La modification des paramètres (sauf donc, ceux qui définissent les extrémités d'une figure ouverte) est alors commandée par la saisie de commandes, séparées par des virgules, de la forme :

$$\langle clé \rangle = \langle valeur \rangle.$$

De fait, lorsque la valeur saisie comporte un nombre décimal, on prendra garde à utiliser le point . (et non la virgule ,) comme séparateur entre sa partie entière et sa partie décimale.

3.3 — Extrémités des figures ouvertes.

Commençons donc par évoquer le cas de ces paramètres dont la modification s'opère différemment de ceux qui seront présentés par la suite : les extrémités des figures ouvertes. Pour renseigner les extrémités d'une telle figure, qui n'en possède pas par défaut, on utilise une commande la forme :

$$\langle extrémité\ 1 \rangle - \langle extrémité\ 2 \rangle$$

saisie entre accolades {}. Si aucune extrémité n'est renseignée, aucune extrémité n'est dessinée. Voici une liste de quelques valeurs et rendus pour $\langle extrémité\ 1 \rangle$ et $\langle extrémité\ 2 \rangle$:

Valeur	Rendu
< ou >	Simple flèche.
<< ou >>	Double flèche.
[ou]	Crochet.
(ou)	Arc.
*	Disque centré sur l'extrémité du trait.
**	Disque collé à l'extrémité du trait.
o	Cercle centré sur l'extrémité du trait.
oo	Cercle collé à l'extrémité du trait.
	Barre collée à l'extrémité du trait.
*	Barre centrée sur l'extrémité du trait.
c	Bout arrondi centré sur l'extrémité du trait.
cc	Bout arrondi collé à l'extrémité du trait.
< ou >	Combine les effets de , et < ou >.

3.4 — Unités.

Même si nous ne l'avons pas explicitement précisé jusqu'à présent, **PSTricks** suppose le plan muni d'un repère orthogonal dont les unités graphiques par défaut sont de 1 cm en abscisses et ordonnées. Les clés **xset** et **yset** permettent de modifier ces paramètres. Étant donnés deux réels λ et μ strictement positifs, on écrira :

$$xunit=\lambda cm \quad \text{et} \quad yunit=\mu cm$$

pour que les unités graphiques soient plutôt de λ cm en abscisses et μ cm en ordonnées.

Suivant le même principe, la clé **runit** permet d'ajuster l'unité de base des rayons de cercles ou arcs de cercle, qui égale 1cm par défaut.

Pour finir, précisons que les angles s'expriment par défaut en degrés sous **PSTricks**.

3.5 — Affichage de points.

La clé **showpoints**, ayant pour valeur les booléens **true** et **false**, gère l'affichage des sommets d'un polygone ou d'une ligne brisée, des points par lesquelles passe une courbe d'interpolation, etc. Ceux-ci n'étant de base pas rendus visibles par **PSTricks**, la valeur de cette clé, qui est donc de **false** par défaut, doit être affectée à **true** pour permettre leur affichage.⁵

3.6 — Style des points.

La clé associée au style des points est **dotstyle**; sa valeur par défaut est *. Voici quelques-unes des valeurs qui peuvent lui être attribuées et leur rendu :

4. Pour les figures rencontrées à la section 2, il s'agit bien entendu des accolades ou crochets laissés vides.

5. Évidemment, toute personne comprenant un minimum l'anglais trouvera tout ceci « obvious »...

Valeur	Rendu
<code>*</code>	●
<code>x</code>	×
<code>o</code>	○
<code>+</code>	+

Valeur	Rendu
<code>asterisk</code>	*
<code>square</code>	□
<code>triangle</code>	△
<code>diamond</code>	◇

3.7 — Taille des points.

La taille des points peut être ajustée via la clé `dotsscale`; sa valeur par défaut est 1. Les valeurs prises par cette clé sont deux nombres séparés par un espace : le premier correspond à l'échelle horizontale et le second à l'échelle verticale. Si un seul nombre est saisi, il s'applique à ces deux échelles.

3.8 — Épaisseur des traits.

La clé définissant l'épaisseur des trait est `linewidth`. Sa valeur est un nombre (alors exprimé dans l'unité de base), ou un nombre avec une unité (par exemple le centimètre `cm` ou la taille de point `pt`). Étant donné un réel λ strictement positif, on écrira par exemple :

`linewidth= λ pt`

si l'on souhaite que l'épaisseur du trait égale λ pt. Par défaut, `linewidth` vaut 0.8pt.

3.9 — Style des traits.

La clé définissant le style de trait est `linestyle`. Voici ses différentes valeurs :

Valeur	Rendu
<code>solid</code>	—
<code>dashed</code>	- - - -




Valeur	Rendu
<code>dotted</code>
<code>none</code>	

Ainsi, sa valeur par défaut est `solid`.

3.10 — Remplissage des figures fermées.

Le remplissage des figures fermées est défini par la clé `fillstyle`, à laquelle il est possible d'attribuer les valeurs :

Valeur	Rendu
<code>none</code>	Aucun remplissage.
<code>solid</code>	Remplissage plein.

Valeur	Rendu
<code>hlines</code>	
<code>vlines</code>	
<code>crosshatch</code>	

Par défaut, la valeur de `fillstyle` est `none`. Notons que les valeurs `hlines`, `vlines` et `crosshatch` fournissent un remplissage transparent; si l'on souhaite obtenir un remplissage opaque, on utilisera à la place `hlines*`, `vlines*` ou `crosshatch*`.

Il est ensuite possible d'ajuster les paramètres des hachures produites par les valeurs `hlines`, `hlines*`, `vlines`, `vlines*`, `crosshatch` et `crosshatch*`. Pour cela, on utilisera les clés :

- `hatchsep`, pour modifier l'épaisseur des hachures.
- `hatchstyle` pour gérer l'espacement des hachures.
- `hatchangle` pour modifier l'angle d'inclinaison des hachures, qui est par défaut de 45 degrés pour `hlines` et `hlines*`, et de 225 degrés pour `vlines` et `vlines*`.

3.11 — Couleur des figures.

On utilise essentiellement trois clés pour modifier la couleur des figures (par défaut en noir et blanc) :

- `linecolor`, qui détermine la couleur des points ou des traits.
- `hatchcolor`, qui détermine la couleur des hachures remplissant les figures fermées.
- `fillcolor`, qui détermine la couleur de remplissage des figures fermées; pour que celle-ci soit visible lorsque la figure est en même temps hachurée, on devra utiliser les paramètres `hlines*`, `vlines*` ou `crosshatch*`.

Chacune d'elles prend des valeurs associées aux couleurs définies par le package `xcolor` qui, rappelons-le, est automatiquement chargé par `PSTricks`. Leur valeur par défaut est `black`⁶.

6. Ce qui, pour ceux qui l'ignorent, signifie « noir » en anglais...

3.12 — Extension de paramètres à plusieurs figures.

Comme nous avons pu le constater (et le constaterons encore), il existe une multitude de paramètres sous PSTricks. Pour ne pas systématiquement avoir à renseigner les paramètres pouvant être placés entre crochets [] et que l'on souhaite appliquer à plusieurs figures, on pourra utiliser la commande `\psset{}`.

Dès sa déclaration, l'effet de chaque paramètre qui lui est passé en argument est appliqué, à moins qu'une commande ultérieure n'impose son altération. En outre, lorsqu'elle est placée dans un domaine délimité par des accolades ou des commandes du type `\begin` et `\end` (e.g. environnement `center`), son champ d'application est restreint à ce domaine.

EXEMPLE. Illustrons l'effet de la commande `psset` sur les quatre situations suivantes :

<pre>\psline(0,0)(1,1) \psline(0,1)(1,0) \psframe[linestyle=dashed](0,0)(1,1)</pre>	
<pre>\psset{xunit=2cm, linestyle=dotted} \psline(0,0)(1,1) \psline(0,1)(1,0) \psframe[linestyle=dashed](0,0)(1,1)</pre>	
<pre>\psline(0,0)(1,1) \psset{xunit=2cm, linestyle=dotted} \psline(0,1)(1,0) \psframe[linestyle=dashed](0,0)(1,1)</pre>	
<pre>{ \psline(0,0)(1,1) \psset{xunit=2cm, linestyle=dotted} \psline(0,1)(1,0) } \psframe[linestyle=dashed](0,0)(1,1)</pre>	

4 Texte sous PSTricks

4.1 — Rappelons (voir 2.1) que les accolades ou crochets laissés vides (`{}` ou `[]`) servent seulement à indiquer où certains paramètres optionnels pourront être renseignés : leur saisie s'avère donc facultative.

4.2 — Pour placer du texte sous un dessin PSTricks, la solution la plus simple consiste à utiliser la commande :

```
\rput{(x, y)}{<texte>}
```

qui place le texte saisi dans le champ `<texte>` au point de coordonnées (x, y) . Si l'on souhaite en outre incliner ce texte d'un angle θ (valant 0 degré par défaut), on renseignera plus précisément :

```
\rput{\theta}(x, y){<texte>}
```

4.3 — Si l'on souhaite plutôt placer du texte en décalé par rapport à un point donné, on utilisera plutôt :

```
\uput{[alpha]}{(x, y)}{<texte>}
```

Ce faisant, le texte sera décalé par rapport au point de coordonnées (x, y) d'un angle α . S'il l'on veut de plus décaler d'une distance δ (de 5pt par défaut) le texte du point (x, y) et le tourner d'un angle θ (de 0 degré par défaut), il suffit de renseigner :

```
\uput{\delta}[alpha]{\theta}(x, y){<texte>}
```

4.4 — Il est possible également possible d'encadrer du texte par utilisation des commandes suivantes :

Commande	Rendu
<code>\psframebox{<texte>}</code>	
<code>\psframebox{<texte>}</code>	
<code>\psshadowbox{<texte>}</code>	
<code>\psdiabox{<texte>}</code>	

Commande	Rendu
<code>\pscirclebox{<texte>}</code>	
<code>\psovalbox{<texte>}</code>	
<code>\pstribox{<texte>}</code>	

5 Quadrillages et axes

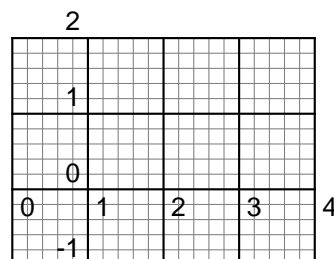
5.1 — Quadrillages.

Pour faire apparaître un quadrillage (qui recouvre par exemple un environnement `pspicture`) d'origine le point (x_0, y_0) et s'étendant du point (x_-, y_-) au point (x_+, y_+) , on déclarera :

```
| \psgrid[] (x_0, y_0) (x_-, y_-) (x_+, y_+)
```

EXEMPLE. Illustrons sur un cas concret l'effet de cette commande :

```
| \psgrid(1,0) (0,-1) (4,2)
```



Il est bien entendu possible de modifier l'apparence par défaut du quadrillage en saisissant, comme nous l'avons fait à la section 3, des commandes du type $\langle clé \rangle = \langle valeur \rangle$ entre les crochets `[]` laissés vides ou par utilisation de la commande `psset`.

- La clé `subgriddiv`, dont la valeur est un nombre, permet la gestion du nombre de sous-graduations entre deux graduations principales, qui égale 5 par défaut.
- La clé `griddots`, dont la valeur est un nombre, permet de remplacer les graduations principales par des pointillés ; la valeur saisie correspondra alors au nombre de pointillés par graduation. La clé `subgriddots` a un fonctionnement analogue avec les sous-graduations.
- Les clés `gridwidth` et `subgridwidth`, ayant pour valeur un nombre avec une unité, permettent respectivement de modifier l'épaisseur des graduations et sous-graduations.
- Les clés `gridcolor` et `subgridcolor`, qui ont pour valeur une couleur issue du package `xcolor`, permettent respectivement de modifier la couleur des graduations et sous-graduations.
- La clé `gridlabels`, ayant pour valeur un nombre avec une unité, permet de modifier la taille des nombres associés aux graduations principales. Pour ne pas afficher ces nombres, on pourra saisir `gridlabels=0`.

5.2 — Jusqu'à la fin de la section 6, les commandes présentées nécessiteront le chargement de l'extension `pstricks-add` du package `pstricks`. Pour cela, nous pourrions donc écrire :

```
| \usepackage{pstricks-add}
```

en lieu et place de `\usepackage{pstricks}`. Bien entendu, l'intégralité des fonctionnalités offertes par `pstricks` restent disponibles avec cette extension.

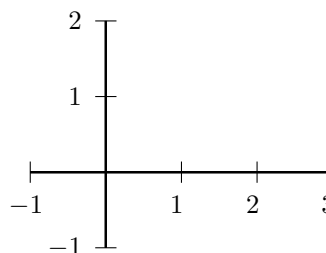
5.3 — Axes.

Pour faire apparaître des axes horizontaux et verticaux, par exemple au sein d'un environnement `pspicture`, se coupant en un point (x_0, y_0) et s'étendant du point (x_-, y_-) au point (x_+, y_+) , on utilisera la commande :

```
| \psaxes[] {} (x_0, y_0) (x_-, y_-) (x_+, y_+)
```

EXEMPLE. Voici le rendu de la commande `psaxes` sur cas particulier :

```
| \psaxes(1,0) (0,-1) (4,2)
```

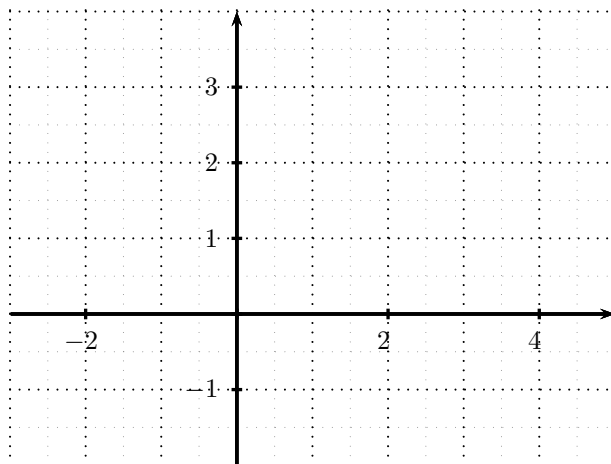


Tous les paramètres de traits présentés aux paragraphes 3.3, 3.8, 3.9 et 3.11 permettront de modifier l'aspect des axes. En particulier, renseignant `->` entre les accolades `{}` laissées vides, les extrémités des axes seront fléchées. Remarquons que les axes sont par défaut étiquetés et numérotés ; mais contrairement à ce que produisait la commande `psgrid`, la numérotation obtenue ici n'est pas fidèle à utilisée lors de la saisie des commandes `PSTricks`. Pour modifier ces paramètres d'étiquetage et de numérotation, on saisira là encore des commandes du type $\langle clé \rangle = \langle valeur \rangle$.

- La clé `Ox`, ayant pour valeur un nombre égal à 0 par défaut, permet de modifier la numérotation de l'axe horizontal à partir de l'origine. Pour en faire autant sur l'axe vertical, on utilisera de la même façon la clé `Oy`.
- La clé `Dx`, ayant pour valeur un nombre égal à 1 par défaut, permet d'ajuster le pas des numérotations sur l'axe horizontal. Pour en faire autant sur l'axe vertical, on utilisera de la même façon la clé `Dy`.
- La clé `labels` gère l'affichage des numéros sur les deux axes. Ses valeurs sont `all` (par défaut), `x` pour n'afficher les numéros que sur l'axe horizontal, `y` pour n'afficher les numéros que sur l'axe vertical, et `none` pour supprimer la numérotation des deux axes.
- La clé `ticks` gère l'affichage des étiquettes sur les deux axes. Ses valeurs sont `all` (par défaut), `x` pour n'afficher les étiquettes que sur l'axe horizontal, `y` pour n'afficher les étiquettes que sur l'axe vertical, et `none` pour supprimer les étiquettes des deux axes.
- Pour modifier la taille des étiquettes, on utilisera la clé `ticksize`. Cette clé a pour valeur un nombre négatif avec une unité et un nombre positif avec une unité séparés par un espace (`-4pt 4pt` par défaut), correspondant respectivement à la taille des étiquettes sur la partie négative et la partie positif des axes.
Si l'on souhaite avoir des paramètres différents sur les deux axes, on utilisera de la même façon les clés `xticksize` pour l'axe horizontal, et `yticksize` pour l'axe vertical.
- Si les clés `linewidth` et `linecolor` permettent d'ajuster l'épaisseur et couleur des axes, celles-ci n'affectent pas les étiquettes. Pour modifier l'épaisseur et couleur de ces dernières, on utilisera de la même manière les clés `tickwidth` et `tickcolor`.

5.4 — EXEMPLE. Pour illustrer les propos tenus aux paragraphes 5.1 et 5.3, on propose l'exemple suivant :

```
\psgrid[gridlabels=0,
  griddots=8,
  subgriddiv=2,
  subgriddots=4]{}(0,0)(-3,-2)(5,4)
\psaxes[ticksize=-2pt 2pt,
  linewidth=1.25pt,
  tickwidth=1.25pt,
  Dx=2]{}(0,0)(-2.99,-1.99)(5,4)
```



6 Tracé de courbes

6.1 — Pour rappel (voir 5.2), les fonctionnalités présentées dans cette section nécessitent le chargement de l'extension `pstricks-add` du package `pstricks`.

6.2 — Nous nous intéresserons ici à deux types de courbes : courbes représentatives d'une fonction et courbes paramétrées. Dans cette section, nous supposons l'ensemble des commandes saisies précédées d'un :

```
| \psset{algebraic=true}
```

Le paramètre `algebraic=true`⁷ permet de saisir les expressions mathématiques sous leur forme classique⁸. Indiquons à cet effet l'équivalent PSTricks de quelques opérations ou fonctions usuelles :

Opération/fonction	Commande
+	+
−	−
×	*
÷	/
puissance	^
$\sqrt{}$	sqrt
exp	EXP

Opération/fonction	Commande
ln	ln
cos	cos
sin	sin
tan	tan
arccos	acos
arcsin	asin
arctan	ATAN

7. Qui pourrait, néanmoins, aussi être passé entre crochets...

8. Sans quoi il faudrait adopter la notation polonaise inversée !

6.3 — Courbes représentatives de fonctions.

Soient a et b deux réels tels que $a \leq b$, et f une fonction dont l'ensemble de définition contient l'intervalle $[a, b]$. Pour tracer la courbe représentative de f sur $[a, b]$, on utilise la commande `psplot`. Une première utilisation de cette commande consiste à tout simplement écrire :

```
| \psplot[ ]{a}{b}{\langle expression de f(x) \rangle}
```

où, dans l'expression de $f(x)$, la variable x doit être notée \mathbf{x} .

Une seconde méthode consiste à préalablement utiliser la commande `def` afin de définir une fonction `PSTricks` associée à f , par exemple nommée `fct`, en déclarant :

```
| \def\fct{\langle expression de f(x) \rangle}
```

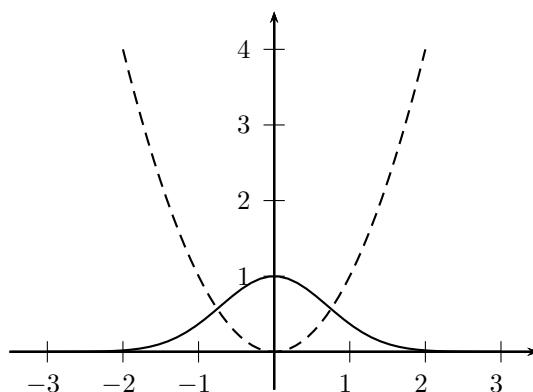
Là encore, la variable x dans l'expression de $f(x)$ devra être notée \mathbf{x} . On utilisera alors `psplot` comme suit :

```
| \psplot[ ]{a}{b}{\fct}
```

L'avantage de cette seconde méthode est qu'elle permet d'utiliser les fonctions définies via la commande `def` pour en construire d'autres par utilisation des opérations ou fonctions usuelles présentées au paragraphe 6.1.

EXEMPLE. Si l'on souhaite tracer les courbes représentatives des fonctions $f : x \mapsto x^2$ et $g : x \mapsto \exp(-x^2)$ sur les intervalles $[-2, 2]$ et $[-3, 3]$ respectivement, on pourra procéder comme suit :

```
| \psset{algebraic=true}
| \psaxes{->}(0,0)(-3.5,-0.5)(3.5,4.5)
| \def\f{x^2}
| \def\g{EXP(-\f)}
| \psplot[linestyle=dashed]{-2}{2}{\f}
| \psplot{-3}{3}{\g}
```



6.4 — Courbes paramétrées.

Sur le même principe qu'en 6.3, étant donnés deux réels a, b tels $a \leq b$ et deux fonctions x, y de $[a, b]$ dans \mathbb{R} , on peut tracer la courbe $\Gamma = \{(x(t), y(t)) \mid t \in [a, b]\}$ en utilisant la commande `psparametricplot` comme suit :

```
| \psparametricplot{a}{b}{\langle expression de x(t) \rangle | \langle expression de y(t) \rangle}
```

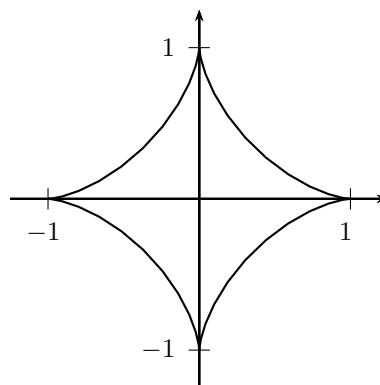
Il est également possible de tracer Γ en définissant au préalable des fonctions `x` et `y` associées à x et y à partir de la fonction `def`, puis d'utiliser ensuite `psparametricplot` comme suit :

```
| \psparametricplot{a}{b}{\x|\y}
```

La seule différence est qu'ici, la variable t dans les expressions de $x(t)$ et $y(t)$ devra être notée \mathbf{t} .

EXEMPLE. Supposons ici $a = -\pi$, $b = \pi$ et $x : t \mapsto \cos(t^3)$, $y : t \mapsto \sin(t^3)$, Γ peut être représentée comme suit :

```
| \psset{algebraic=true, xunit=2cm, yunit=2cm}
| \psaxes{->}(0,0)(-1.25,-1.25)(1.25,1.25)
| \def\x{(cos(t))^3}
| \def\y{(sin(t))^3}
| \psparametricplot{-3.1416}{3.1416}{\x|\y}
```



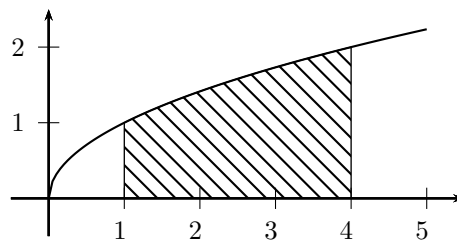
6.5 — REMARQUE. Comme on aura pu le voir dans les exemples précédents, il ne faut pas hésiter à user (voire abuser) de parenthèses pour s'assurer que \LaTeX effectue bien les opérations dans l'ordre attendu.

6.6 — Par défaut, \LaTeX calcule 50 points pour réaliser la représentation graphique d’une courbe. Pour améliorer la qualité des sorties graphiques, qui s’avère parfois grossière, on pourra ajuster ce paramètre à partir de la clé `plotpoints`, dont la valeur correspond au nombre de points calculés (et qui est donc de 50 par défaut).

7 Pour aller plus loin, ou ailleurs...

7.1 — `PSTricks`, c’est donc une foule de fonctionnalités et paramètres dont ce document donne un tout petit aperçu. Mais c’est aussi (et surtout ?) de nombreuses extensions tout aussi riches et variées. Pour n’en citer que quelques-unes, signalons :

- `pstricks-add`, à laquelle nous avons déjà eu recours dans ce qui précède, avec laquelle on peut aussi :
 - tracer des tangentes à un cercle, un courbe, etc. ;
 - représenter des aires ou domaines entre plusieurs courbes :

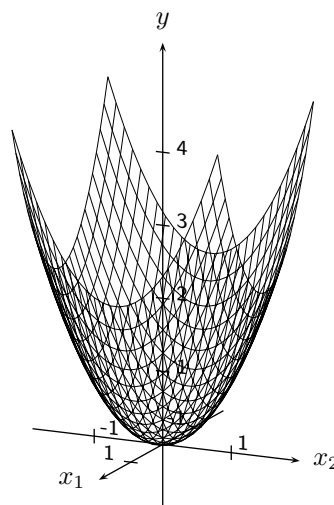


- tracer les solutions approchées d’une équation différentielle ordinaire par différentes méthodes ;
- et bien d’autres choses encore !
- `pst-node`, extension permettant la réalisation d’arbres, graphes, diagrammes ou organigrammes ; par exemple, un diagramme commutatif illustrant la formule du changement de base :

$$\begin{array}{ccc}
 (E, \mathcal{B}) & \xrightarrow{B} & (E, \mathcal{B}) \\
 \uparrow \text{id}_E & \searrow f & \downarrow \text{id}_E \\
 (E, \mathcal{E}) & \xrightarrow[A]{} & (E, \mathcal{E})
 \end{array}$$

P^{-1} on the left, P on the right

- `pst-3d` et `pst-3dplot`, avec lesquelles il est possible de dessiner et réaliser de jolies courbes en dimension 3 :



- `multido`, qui met à disposition la commande du même nom permettant d’effectuer des boucles, et de pouvoir ainsi automatiser certaines commandes `PSTricks` ;
- `pst-math`, qui propose une plus grande diversité de fonctions et opérateurs mathématiques que ceux proposés au paragraphe 6.2 ;

- ou bien encore des extensions un peu plus originales que les précédentes, parmi lesquelles :
 - `pst-text`, permettant de disposer de manière « exotique » du texte :

Comme par exemple le long d'une jolie courbe...

- ou même `pst-fun`, avec laquelle on pourra égayer nos articles, thèses, rapports ou supports de cours :



7.2 — Évidemment, le dessin sous \LaTeX ne se limite pas à `PSTricks`. Il est tout d'abord possible de faire du dessin sans utiliser le moindre package (c'est par exemple ce que nous faisons lorsque nous utilisons la commande `hrulefill`, qui permet de tracer un trait horizontal), mais les possibilités restent alors extrêmement limitées. À celles et ceux qui n'auraient pas été convaincus par la diversité des fonctionnalités offertes par `PSTricks`⁹, ou qui ne parviennent pas à exécuter le dessin espéré sous `PSTricks`¹⁰, nous suggérons l'utilisation de :

- `TikZ`, package aux fonctionnalités à peu près similaires à celles proposées `PSTricks`¹¹, mais qui utilise une syntaxe différente.
- `Asymptote`, logiciel permettant la réalisation de nombreuses figures géométriques s'intégrant facilement à un document \LaTeX .

Références

- [1] D. BITOUZÉ, J.-C. CHARPENTIER, *ETEX, l'essentiel*, Pearson, 2010.
- [2] Espace pédagogique de l'académie de Poitiers, *Des courbes en LaTeX : Extension pstricks-add*, mise à jour du 14/12/2013.
- [3] D. RODRIGUEZ, M. SHARPE, H. VOß, *pstricks-add, additional Macros for pstricks*, disponible sur le site du CTAN, 2018.
- [4] T. TANTAU, *The TikZ and PGF packages*, disponible sur le site du CTAN, 2015.
- [5] G. TISSEAU et J. DUMA, *TikZ pour l' impatient*, disponible sur le site <http://math.et.info.free.fr>, 2017.
- [6] T. VAN ZANDT, *PSTricks, User's guide*, disponible sur le site du CTAN, 2003.
- [7] Wikibooks, *LaTeX, Dessiner avec LaTeX, Dessiner avec PSTricks*, mise à jour du 23/8/2017.

9. Mais qui, il faut l'avouer, sont tout de même un peu difficiles...

10. Nous invitons cette deuxième catégorie de personnes à poursuivre leurs recherches : d'autres ont sans doute déjà été confronté à un problème similaire, et trouvé comment le solutionner !

11. Si la richesse de la documentation `PSTricks` peut être rebutante, qu'on ne s'y méprenne pas : celle de `TikZ` compte 1 161 pages.