



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

L^AT_EX et P^ST_ricks

Michel VAN GARREL
<michel.vangarrel@epfl.ch>

Antonin DANALET
<antonin.danalet@epfl.ch>

Lausanne, le 16 mai 2005

Résumé

PSTricks est une extension de \LaTeX , développée par Timothy Van Zandt, qui permet la réalisation de graphismes élaborés en s'appuyant sur PostScript, et cela sans avoir à écrire de fichiers séparés.

Nous vous proposons ici une brève introduction à cet outil.

Table des matières

Introduction	2
Remarques préliminaires	3
1 Commandes de base	6
1.1 Tracer des lignes	6
1.2 Tracer des rectangles et des polygones	10
1.3 Tracer des cercles	12
1.4 Tracer une ellipse et une portion de disque	12
1.5 Couleur	13
2 Courbes	15
2.1 Système de coordonnées par défaut	15
2.2 Courbes non-paramétrisées :la commande <code>\pscurve</code>	15
2.2.1 Les options	16
2.3 Courbes paramétrées :la commande <code>\parametricplot</code>	17
2.3.1 Syntaxe de base	17
2.3.2 Autres opérateurs	18
2.3.3 Les options	19
3 Arbres	20
3.1 La commande <code>\pstree</code>	21
3.2 Exploiter les possibilités de <code>\pstree</code>	21
3.2.1 Les options de <code>\pstree</code>	21
3.2.2 Autre paramètres	22
4 Chaînes de Markov - Introduction aux Noeuds	24
4.1 La commande <code>\psmatrix</code>	25
4.1.1 Remarques conceptuelles	25
Bibliographie	28

Introduction

PSTricks permet de disposer, depuis \LaTeX , de certaines des possibilités offertes par PostScript.

PSTricks est un vaste ensemble de macros-commandes utilisables directement depuis \LaTeX . Elles permettent d'avoir du texte avec les couleurs choisies, de définir des objets graphiques simples (lignes, polygones, cercles, flèches, etc.) ou complexes (grilles, arbres, réseaux, etc.) - que l'on peut bien sûr combiner - et de manipuler des parties du texte et du document (rotations, changements d'échelles, transformations). L'éventail des possibilités est donc considérable et n'est guère limité que par l'imagination...

Le manuel de référence et d'utilisation de PSTricks [4] est très complet (plus d'une centaine de pages...), avec un grand nombre d'exemples simples, mais nous souhaitons présenter ici une brève introduction à cet outil, les notions de base ainsi que quelques sujets particuliers, et ceci à l'aide d'une série d'exemples.

Remarques préliminaires

Tout d'abord, pour pouvoir utiliser du code PSTricks dans un document L^AT_EX, il faut insérer dans le préambule, comme pour tous les packages, la ligne suivante :

```
\usepackage{nomdupackage}
```

`nomdupackage` peut être remplacé par le nom du package nécessaire (`pst-node` pour les réseaux, `pst-plot` pour les équations paramétriques, `pstricks`, la macro principale, etc...), mais peut aussi être remplacé par `pst-all`, ce qui permet de charger directement tous les packages de PSTricks¹.

Pour le chapitre 1, les commandes de bases, `pstricks` suffit. Pour le chapitre 2, il faut charger `pst-plot`, et pour le chapitre 3, `pst-tree`. En cas de problème, utiliser `pst-all`.

Ensuite, nous utiliserons plusieurs exemples dans ce document. Chaque exemple est constitué du code PSTricks et du résultat une fois compilé.

Pour alléger l'écriture et améliorer la compréhension, nous omettrons toutefois certaines balises utilisées dans chaque exemple.

Pour illustrer ceci, prenons un exemple simple, et dessinons un point avec PSTricks. Pour ce faire, écrivons ceci dans notre document :

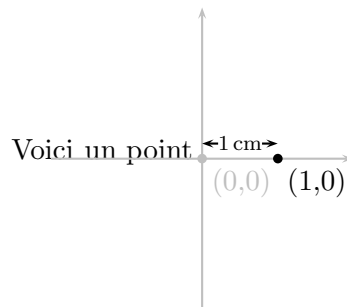
```
Voici un point \psdots(1,0)
```

Et on obtient :

Voici un point •

La commande `\psdots(x,y)` permet de dessiner des points de coordonnées (x,y) . Par défaut, PSTricks utilise comme origine du système d'axe le point d'insertion de la commande, et l'unité par défaut est de 1 cm. Ceci est illustré par la figure ci-dessous, dans laquelle les axes ont été colorés en gris.

¹A noter qu'il n'est pas nécessaire de charger le package "color" si "pstricks" ou "pst-all" sont chargés



Une simple commande `\psdots` peut être utilisée pour dessiner plusieurs points. Par exemple, l'insertion de

```
Voici plusieurs points \psdots(0,0)(2,0)(1,1)
```

crée le résultat suivant :

Voici plusieurs point. • •

Maintenant, essayons ceci :

```
Voici plusieurs points \psdots(0,0)(2,0)(1,1)
formant les sommets d'un triangle.
```

Et voici le résultat :

Voici plusieurs points • formant les • sommets d'un triangle.

Ceci est dû au fait que \LaTeX ne laisse pas de place pour l'image. L'image PSTricks est dessinée après la compilation \TeX , c'est-à-dire par dessus le texte.

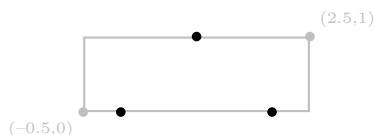
Pour s'assurer que \LaTeX laisse suffisamment de place pour l'image à dessiner, il faut définir une boîte, un cadre. PSTricks permet de le faire à l'aide de la commande `\spicture`.

L'exemple précédent devient :

```
Voici plusieurs points
\begin{pspicture}(-0.5,0)(2.5,1)
\psdots(0,0)(2,0)(1,1)
\end{pspicture}
formant les sommets d'un triangle.
```

Voici plusieurs points • • formant les sommets d'un triangle.

La commande `\pspicture` prend pour argument deux paires de coordonnées, dans notre exemple $(-0.5, 0)$ et $(2.5, 1)$. Ces coordonnées définissent les sommets inférieurs gauche et supérieur droit d'une boîte rectangulaire contenant l'image.



La première paire de coordonnées est optionnelle et vaut $(0, 0)$ par défaut. Autrement dit, la commande

```
\begin{pspicture}(1,2)... \end{pspicture}
```

est équivalente à

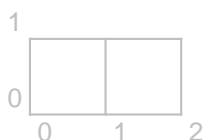
```
\begin{pspicture}(0,0)(1,2)... \end{pspicture}
```

Chaque exemple PSTricks dans la suite de ce document sera encadré des commandes `\begin{center}... \end{center}` et `\begin{pspicture}(x_0, y_0)(x_1, y_1)... \end{pspicture}`.

La plupart des exemples seront de plus muni d'une grille en arrière plan permettant de mieux visualiser le système d'axe.

Ceci est obtenu à l'aide de la commande `\psgrid[option](x_0, y_0)(x_1, y_1)`

```
\psgrid[subgriddiv=0, gridcolor=lightgray,
gridlabelcolor=lightgray](0,0)(2,1)
```



Chapitre 1

Commandes de base

1.1 Tracer des lignes

Par défaut, PSTricks utilise un système de coordonnées cartésiennes dont l'unité est de 1 cm, et dont l'origine est toujours définie en bas à gauche.

Ainsi, pour tracer une ligne depuis l'origine, on utilise la commande `\psline(x,y)`.

Exemple 1.1.

```
\psline(2,1)
```



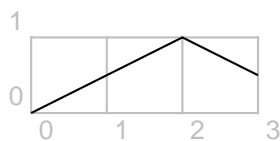
La commande `\psline` s'écrit plus généralement :

```
\psline[opt]{flèche}(x_0, y_0)(x_1, y_1)...(x_n, y_n)
```

$(x_0, y_0)(x_1, y_1)...(x_n, y_n)$ représentent les coordonnées des points reliés par les segments. Par défaut, s'il n'y a qu'une coordonnée, le segment relie ce point à l'origine $(0,0)$ (cf. exemple 1.1).

Exemple 1.2.

```
\psline(0,0)(2,1)(3,0.5)
```



`{flèche}` permet de mettre des flèches aux extrémités des traits, et peut prendre les valeurs suivantes :

`->` produit une ligne avec une flèche à la fin ;

`<-` produit une ligne avec une flèche au début ;

`<->` produit une ligne avec une flèche des deux cotés.

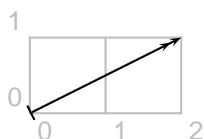
Ces trois valeurs sont les plus fréquentes, mais il y en a d'autres :

Valeur	Exemple	Nom
-	—	rien
<->	↔	flèches aux extrémités
>-<	↠	flèches inversées aux extrémités
<<->>	↔↔	doubles flèches aux extrémités
>>-<<	↠↠	doubles flèches inversées aux extrémités
-	┌──┐	extrémités en T
[-]	[──]	extrémités en crochets
(-)	(──)	extrémités en parenthèses
o-o	○──○	extrémités en cercles
_	●──●	extrémités en disques
c-c	—	extrémités arrondies

Ces valeurs peuvent bien entendu être combinées.

Exemple 1.3.

```
\psline{|->>}(2,1)
```



Selon Timothy Van Zandt, elles sont intuitives et sont la version $\text{T}_{\text{E}}\text{X}$ du WYSIWYG¹.

`[opt]` est une option qui permet de modifier les caractéristiques des objets représentés. En voici quelques unes :

linecolor = couleur lignes en couleurs.

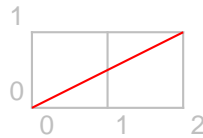
Couleur peut prendre les valeurs *red*, *green*, *blue*, *cyan*, *magenta* et *yellow*².

¹What You See Is What You Get

²Pour plus de détails sur l'utilisation des couleurs avec PSTricks, se référer au chapitre 1.5, page 13

Exemple 1.4.

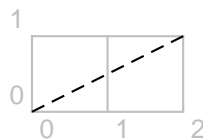
```
\psline[linecolor=red](2,1)
```



linestyle = dashed/dotted lignes en traitillés/pointillés.

Exemple 1.5.

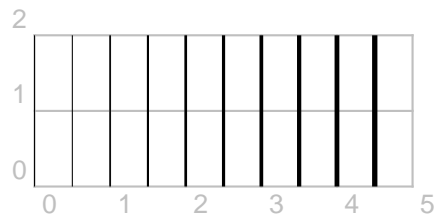
```
\psline[linestyle=dashed](2,1)
```



linewidth = largeur modifie l'épaisseur de la ligne.

Exemple 1.6.

```
\psline[linewidth=0.2pt](0,0)(0,2)  
\psline[linewidth=0.4pt](0.5,0)(0.5,2)  
\psline[linewidth=0.6pt](1,0)(1,2)  
\psline[linewidth=0.8pt](1.5,0)(1.5,2)  
\psline[linewidth=1pt](2,0)(2,2)  
\psline[linewidth=1.2pt](2.5,0)(2.5,2)  
\psline[linewidth=1.4pt](3,0)(3,2)  
\psline[linewidth=1.6pt](3.5,0)(3.5,2)  
\psline[linewidth=1.8pt](4,0)(4,2)  
\psline[linewidth=2pt](4.5,0)(4.5,2)
```

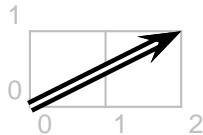


doubleline = true double la ligne.

doublesep = séparation modifie l'écart entre les deux lignes.

Exemple 1.7.

```
\psline[linewidth=0.6mm, doubleline=true,  
doublesep=0.5mm]{->}(2,1)
```

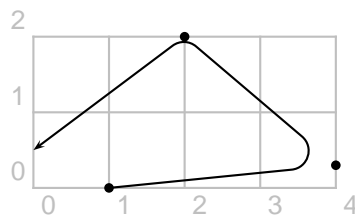


linearc = dim définit le rayon des arcs tracés aux coins des lignes.

showpoints = true montre les points qui dessinent l'objet.

Exemple 1.8.

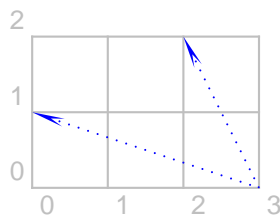
```
\psline[linearc=0.25, showpoints=true]  
{->}(1,0)(4,0.3)(2,2)(0,0.5)
```



arrowlength = longueur permet de modifier la longueur des flèches.

Exemple 1.9.

```
\psline[linestyle=dotted, linecolor=blue,  
arrowlength=10]{<->}(0,1)(3,0)(2,2)
```



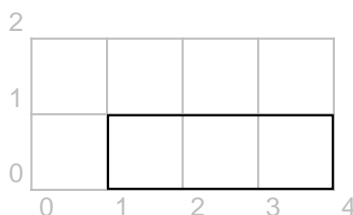
Il y a bien entendu d'autres options sur les lignes, que l'on peut retrouver dans le guide de l'utilisateur de PSTricks de Timothy Van Zandt [4].

1.2 Tracer des rectangles et des polygones

`\psframe[opt](x0,y0)(x1,y1)` permet de tracer un rectangle dont le point inférieur gauche est (x_0, y_0) et le point supérieur droit (x_1, y_1) .

Exemple 1.10.

```
\psframe(1,0)(4,1)
```



`[opt]` est une option qui permet de modifier les caractéristiques des objets représentés. En voici quelques unes :

fillcolor = couleur pour remplir avec une couleur³.

fillstyle=style permet de remplir l'objet selon le style défini. En voici quelques uns :

solid permet de remplir l'objet par la couleur sollicitée au moyen de la commande `\fillcolor` ;

vlines, vlines*, hlines, hlines*, crosshatch et crosshatch* tous ces styles permettent de remplir l'objet par des lignes.

Ces styles peuvent être contrôlés par les paramètres suivants :

fillcolor = color utilisée seulement avec le style *solid* et les styles en forme étoilés ;

hatchwidth = dim épaisseur des lignes ;

hatchsep = dim espace entre les lignes ;

hatchcolor = color couleur des lignes ;

hatchangle = rot rotation des lignes en degré.

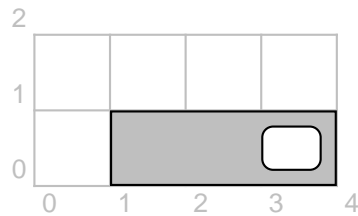
Exemple 1.11.

³Pour plus de détails sur l'utilisation des couleurs avec PSTricks, se référer au chapitre 1.5, page 13

```

\psframe[fillstyle=solid, fillcolor=lightgray]
(1,0)(4,1)
\psframe[fillstyle=solid, fillcolor=white,
framearc=0.5](3,0.2)(3.8,0.8)

```

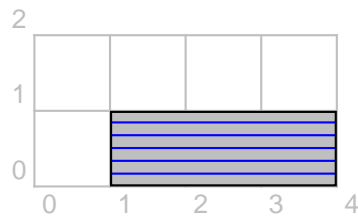


Exemple 1.12.

```

\psframe[fillstyle=vlines*, hatchangle=90,
hatchcolor=blue, fillcolor=lightgray](1,0)(4,1)

```



shadow = true permet d'afficher l'ombre de l'objet. L'ombre peut être contrôlée par les paramètres suivants :

shadowsize = dim épaisseur de l'ombre ;

shadowangle = angle position de l'ombre ;

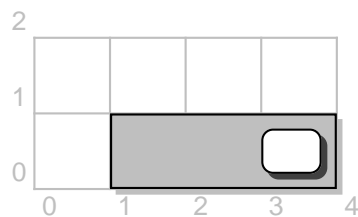
shadowcolor = color couleur de l'ombre.

Exemple 1.13.

```

\psframe[fillstyle=solid, fillcolor=lightgray,
shadow=true, shadowcolor=lightgray](1,0)(4,1)
\psframe[fillstyle=solid, fillcolor=white,
framearc=0.5, shadow=true](3,0.2)(3.8,0.8)

```



1.3 Tracer des cercles

`\pscircle[opt](x0,y0){rayon}`

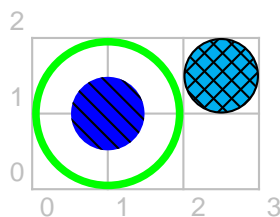
Cette commande trace un cercle de rayon *rayon* et de centre (x_0, y_0) .

opt : peut avoir n'importe quelle option parmi celles que l'on a vu précédemment.

La forme étoilée trace un cercle plein de rayon *rayon*.

Exemple 1.14.

```
\pscircle[linewidth=1mm, linecolor=green](1,1){1}
\pscircle[fillstyle=vlines*, linestyle=none,
fillcolor=blue](1,1){0.5}
\pscircle[fillstyle=crosshatch*, fillcolor=cyan]
(2.5, 1.5){0.5}
```



1.4 Tracer une ellipse et une portion de disque

`\psellipse[opt](x0,y0)(x1,y1)`

Cette commande trace une ellipse de centre (x_0, y_0) . x_1 et y_1 sont respectivement les rayons horizontal et vertical.

La forme étoilée trace une ellipse pleine.

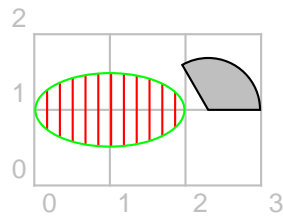
`\pswedge[opt](x0,y0){rayon}{angle1}{angle2}`

Cette commande trace une portion de disque de centre (x_0, y_0) , de rayon *rayon* et étendu de *angle1* à *angle2* dans le sens trigonométrique.

La forme étoilée trace un objet plein.

Exemple 1.15.

```
\psellipse[fillstyle=vlines, linecolor=green,
hatchangle=0, hatchcolor=red](1,1)(1,0.5)
\pswedge[fillstyle=solid,
fillcolor=lightgray](2.3,1){0.7}{0}{120}
```



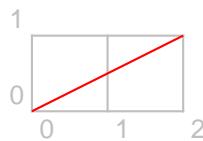
1.5 Couleur

Les différents tons de gris "black", "darkgray", "gray", "lightgray" et "white", ainsi que les couleurs "red", "green", "blue", "cyan", "magenta" et "yellow" sont prédéfinies dans PSTricks.

Cela signifie que ces noms peuvent être utilisés avec les objets graphiques qui ont été définis dans les sections précédentes. Cela signifie aussi que les commandes de type `\blue` (ou `\red`, etc.) peuvent être utilisées comme toute autre option sur le texte.

Exemple 1.16.

```
\psline[linecolor=red](2,1)
```



Exemple 1.17.

```
\blue{Texte en bleu}
```

Texte en bleu

Des tons de gris et des couleurs peuvent aussi être définis ou redéfinis avec les commandes `\newgray{couleur}{num}` et `\newrgbcolor{couleur}{num1 num2 num3}`.

Dans chaque exemple, *couleur* est le nom de la couleur créée.

Dans le cas de la commande `\newgray{couleur}{num}`, *num* est un nombre entre 0 et 1, exprimant le ton, 0 représentant noir et 1 blanc.

Dans le cas de la commande `\newrgbcolor{couleur}{num1 num2 num3}`, les couleurs s'expriment en rouge - vert - bleu (RGB)...

Exemple 1.18.

```
\newgray{grisfoncé}{0.25}
\grisfoncé Texte en gris foncé
```

Texte en gris foncé

Soulignons que les noms de couleurs ne peuvent pas prendre d'accent. Ici, on ne peut pas appelé la couleur définie "gris foncé", avec un accent aigu.

Exemple 1.19.

```
\newrgbcolor{vert}{0 1 0}  
\vert Texte en vert
```

Texte en vert

Chapitre 2

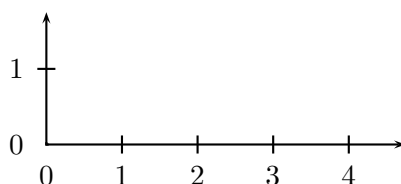
Courbes

2.1 Système de coordonnées par défaut

PSTricks travaille par défaut dans \mathbb{R}^2 avec le système de coordonnées $0xy$ où le point $(0,0)$ est situé en bas à gauche. (x,y) est la syntaxe pour désigner le point à l'abscisse x et l'ordonnée y .

Exemple 2.1.

```
\psaxes{->}(4.75,1.75)
```



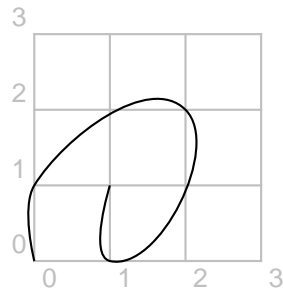
2.2 Courbes non-paramétrisées : la commande `\pscurve`

L'outil `\pscurve` peut s'avérer très utile lorsqu'il s'agit d'obtenir une simple courbe reliant des points. Evidemment, cette courbe peut être adaptée aux propres besoins à l'aide d'options.

La syntaxe est `\pscurve`(premier point)(deuxième point)...(dernier point)
Les points seront alors reliés ensemble :

Exemple 2.2.

```
\pscurve(1,1)(1,0)(2,2)(0,1)(0,0)
```



2.2.1 Les options

A nouveau, l'outil `\pscurve` est muni d'un ensemble d'options utiles. On peut uniquement mettre des espaces dans les options après les virgules qui séparent les différents options. On a donc :

1. Comme déjà exposé dans le premier chapitre, il est possible de choisir la flèche au bouts de la courbe.
2. On peut également définir des paramètres sur cette flèche. Si `num` est un nombre, `dim` sa dimension (par exemple en mm), on a alors les possibilités suivantes. Les options suivants sont écrits dans la bonne syntaxe.

`arrowsize=numdim`

`arrowlength=num`

`arrowinset=num`

`tbarssize=numdim`

`bracketlength=num`

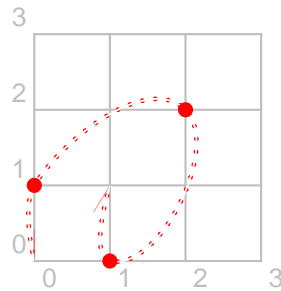
3. Les options `linecolor` et `doublecolor` permettent de choisir la couleur de la courbe.
4. Avec `linewidth`, on peut choisir la largeur de la courbe, il faut l'écrire à nouveau avec la syntaxe `numdim`.
5. `linestyle` définit le style de la courbe. Les styles possibles sont les suivants :

`none solid dashed dotted`

6. `showpoints` et `doubleline` sont est des options binaire.

Exemple 2.3. *Reprenons notre exemple de base modifié :*

```
\pscurve[arrowsize=2mm, arrowlength=2,
arrowinset=1, tbarssize=0.1mm, bracketlength=2,
doubleline=true, linecolor=red, doubleline=cyan,
linewidth=0.2mm, linestyle=dotted,
showpoints=true]{<->}(1,1)(1,0)(2,2)(0,1)(0,0)
```



2.3 Courbes paramétrées : la commande `\parametricplot`

Pour la paramétrisation de fonctions analytiques, l'outil `\pscurve` est bien insuffisant. Pour réaliser des paramétrisations, PSTricks propose l'outil `\parametricplot`. Pour comprendre son utilisation, il faut comprendre les paramétrisations. Nous nous limitons pour cela à la paramétrisation de courbes de \mathbb{R}^2 . Une paramétrisation est essentiellement une fonction f de \mathbb{R} en \mathbb{R}^2 , de laquelle on dessine le graphe. f a donc deux composantes : $f = (f_1, f_2)$. Cette fonction permet de ramener une courbe compliqué à un intervalle de \mathbb{R} , typiquement $[0,1]$. Pour cela, elle fait correspondre à chaque élément de $[0,1]$ un élément de \mathbb{R}^2 . Quand t parcourt $[0,1]$, $(f_1(t), f_2(t))$ parcourt la courbe paramétrée dans \mathbb{R}^2 .

2.3.1 Syntaxe de base

Une paramétrisation f est une fonction analytique pour laquelle il faut définir l'intervalle que t parcourt. La syntaxe de base est :

```
\parametricplot{a}{b}{
f_1(t)
f_2(t)}
```

Avec $[a,b]$ l'intervalle sur lequel f prend ses valeurs. $f_1(t)$ est la valeur par rapport à l'axe $0x$ de la paramétrisation, $f_2(t)$ la valeur par rapport à l'axe $0y$. La syntaxe pour écrire $f_1(t)$ et $f_2(t)$ n'est pas très intuitif. Mais elle est rapide et empêche toute ambiguïté. De plus elle évite des parenthèses.

Pour comprendre une définition de $f_1(t)$ ou $f_2(t)$, il faut la lire de droite à gauche. Un opérateur définit une opération entre deux variables. Il est écrit à droite des deux variables auquel il fait subir l'opération. Les opérateurs et les variables doivent être séparées par un espace. Nous appelons un nombre également une variable, pour simplifier. Illustrons tout ceci à partir de l'exemple suivant :

Posons

```
f_1(t) = 5 t mul
```

L'opérateur est mul. Il est écrit à droite. On a une multiplication de 5 et t.

Exemple 2.4.

$$f_1(t) = 5 \ t \ mul, \ f_2(t) = t, \\ a = 0 \ \text{et} \ b = 1,$$



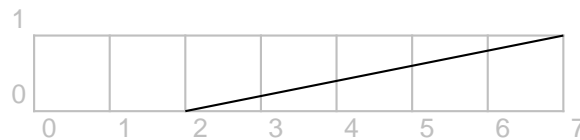
Quand on utilise plusieurs opérateurs, chaque opérateur se réfère au deux variables qui sont à gauche de celui-ci. Si ces variables sont également obtenues à partir d'opérations mathématiques, il sera plus difficile de désigner les deux variables auxquelles l'opérateur se réfère. Néanmoins, la syntaxe est toujours logique.

Exemple 2.5.

Changeons notre exemple en :

$$f_1(t) = 5 \ t \ mul \ 2 \ add$$

Ceci veut dire que on multiplie 5 et t puis on l'additionne à 2. On déplace ainsi notre paramétrisation de 2 centimètres à droite le long de l'axe 0x. Ceci est la même chose que $f_1(t) = 2 \ 5 \ t \ mul \ add$. Nous obtenons



2.3.2 Autres opérateurs

PSTricks possède des fonctions analytiques prédéfinies qui permettent de réaliser des paramétrisations. En voici les plus importantes :

opérateur	description	opérateur	description
add	addition	sin	sinus
sub	soustraction	cos	cosinus
abs	valeur absolue	exp	exponentielle
mul	multiplication	ln	logarithme népérien
div	division	log	logarithme de base 10
neg	négation	round	composition de opérateurs
sqrt	racine carrée		

2.3.3 Les options

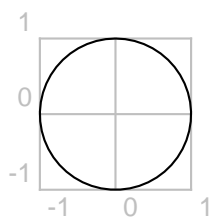
Les deux options essentielles sont :

1. `plotpoints=int` qui permet de choisir le nombre `int` de points qu'on veut voir affichés dans le graphe de la paramétrisation.
2. `plotstyle` qui définit le style de la courbe. Les possibilités sont :

dots line polygon
curve ecurve ccurve

Exemple 2.6.

```
\parametricplot{0}{360}{  
t sin  
t cos}
```

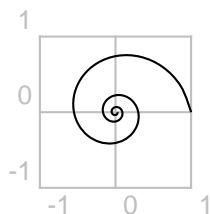


Remarque

En PSTricks, les angles sont mesurées en degrés et non en radian.

Exemple 2.7.

```
\parametricplot[plotstyle=curve]{0}{360}{  
3 t mul cos 2.718 -0.01 t mul exp mul  
3 t mul sin 2.718 -0.01 t mul exp mul}
```

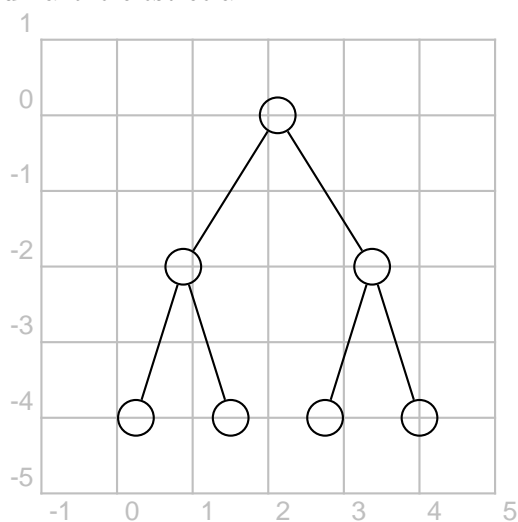


Chapitre 3

Arbres

Beaucoup de situations mathématiques peuvent être représentés par des arbres. Les arbres modélisent par exemple bien des structures hiérarchiques.

A priori, un arbre c'est cela :



la définition mathématique d'un arbre est :

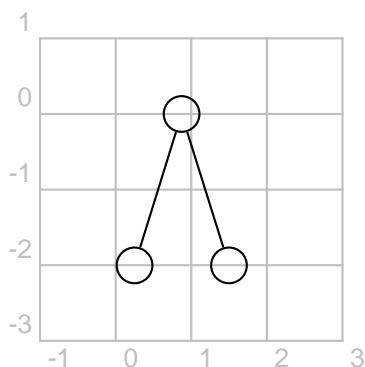
Définition 3.1. *Un arbre est un graphe non orienté sans cycle et connexe.*

Un graphe est un triple, contenant deux ensembles finis et une fonction. Nous nous intéressons cependant à la représentation d'un arbre et non à sa structure mathématique. Voyons donc plutôt l'implémentation PSTricks d'un arbre.

3.1 La commande `\pstree`

Exemple 3.1.

```
\pstree{\TC}{\TC \TC}
```



La commande `\pstree{\TC}` crée un sommet de l'arbre. Ensuite, la commande `\TC` mise en accolades crée des sous-sommets, qui seront reliés au sommet principale.

Pour obtenir un arbre plus complexe, il suffit de généraliser (de complexifier) cette syntaxe. Pour créer un nouveau sommet auquel on pourra associer des sous-sommets, il faut utiliser la commande `\pstree{\TC}`. Or uniquement les derniers sous-sommets peuvent être écrits sans cette commande, i.e. avec `{\TC ... \TC}`. Voici la syntaxe de l'exemple que nous avons affichés au début de ce chapitre :

Exemple 3.2.

```
\pstree{\TC  
{\pstree{\TC}{\TC \TC}  
\pstree{\TC}{\TC \TC}  
}
```

L'ensemble des sous-sommets suivant un sommet est affiché de gauche à droite. Ceci sera important pour la suite si nous voulons des arbres plus complexes.

3.2 Exploiter les possibilités de `\pstree`

3.2.1 Les options de `\pstree`

Nous avons énoncés dans le premier chapitre des options pour des lignes. Ces options sont valables pour les lignes reliant les sommets. Les options

sont mis après le `\pstree`, peuvent donc être mises pour chaque sommet intermédiaire de l'arbre. Voici un choix d'options :

syntaxe	possibilités	description
treemode	R (right), L (left) U (up) ou D (down)	direction dans laquelle l'arbre pousse
treesepl	par exemple 0.75cm	distance entre les sommets
treefit	tight ou loose	loose fait en sorte d'utiliser toute la largeur de la page pour les arbres. tight est la valeur par défaut

3.2.2 Autre paramètres

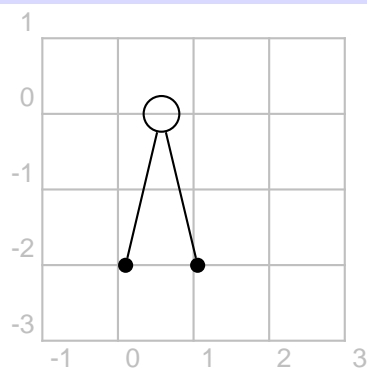
Suivant les principes de PSTricks, il y a plein de possibilités pour arranger les arbres selon de propres besoins. Nous en examinons quelques-uns.

- Remplacer la commande `\TC` par `\TC*` donne des sommets remplis :
- Il est possible de préciser le rayon des sommets. Pour cela, il faut écrire `\Tc` ou `\Tc*`, donc écrire le `c` en minuscules, puis ajouter en accolades derrière `\Tc` ou `\Tc*` le rayon souhaité en mm.

Exemple 3.3.

Pour illustrer ceci, reprenons notre exemple de base, légèrement modifié :

```
\pstree{\TC}{\Tc*{1mm}\Tc*{1mm}}
```



Jusqu'à présent nous avons dessiné uniquement des sommets en cercle ou en disque. Or PSTricks dispose d'autres possibilités pour définir l'allure des sommets. Il faut alors remplacer la commande `\TC` et ses extensions par les commandes qui génèrent des sommets différents. Voici une liste de ces commandes.

commande	description
<code>\Toval{\color texte}</code>	On écrit avec la couleur <i>color</i> le texte <i>texte</i> dans un sommet ovale.
<code>\Tcircle{\couleur texte}</code>	On écrit dans un sommet de la forme d'un cercle
<code>\TR{\couleur texte}</code>	Comme avant, sauf que PSTricks n'affiche pas le cercle.
<code>\Tr{\ps... \couleur texte}</code>	On change l'allure du sommet qui sera donné par la commande <code>\ps...</code> Ceci sera une des commandes énoncées dans le premier chapitre, par exemple <code>\psframe</code> ou <code>\pscircle</code>

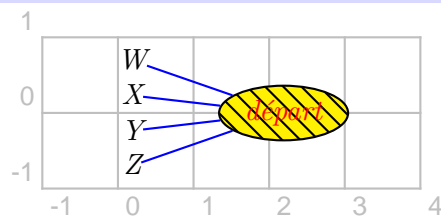
Pour les commandes `\Toval` et `\Tcircle` il y a encore deux options particulièrement intéressantes qui concernent les sommets :

- Avec `fillcolor` on peut décider de la couleur avec laquelle on remplit les sommets.
- Les sommets peuvent être remplis de différents façons `fillstyle` est la commande associée. Les possibilités sont :

<code>none</code>	<code>solid</code>	<code>vlines</code>	<code>vlines*</code>
<code>hlines</code>	<code>hlines*</code>	<code>crosshatch</code>	<code>crosshatch*</code>

Exemple 3.4.

```
\pstree[linecolor=blue,treesep=2mm,treemode=L,treefit=loose]
{\Toval[fillstyle=vlines*,fillcolor=yellow]{\red
\textit{départ}}}}
{\TR{W}
\TR{X}
\TR{Y}
\TR{Z}
}
```



Chapitre 4

Chaînes de Markov - Introduction aux Noeuds

Nous illustrons dans ce chapitre comment il est possible sur PSTricks de réaliser des graphes représentatifs de chaînes de Markov. Ceci servira comme introduction aux noeuds. Les noeuds sont la solution de PSTricks pour dessiner des graphes. Ils permettent de dessiner toutes sortes de sommets, d'y écrire dedans de toutes les façons possibles, puis de relier ces sommets entre eux selon les propres besoin, en y écrivant par exemple quelque chose dessus. L'implémentation PSTricks pour les noeuds est très vaste et offre un grand nombre de possibilités. Nous nous limitons à la réalisation de graphes représentatifs de chaînes de Markov afin de ne pas excéder la taille de ce rapport.

La théorie des chaînes de Markov (ici nous nous occupons de chaînes finies, homogènes et à temps discret) permet de modéliser des systèmes dynamiques particuliers. Ces systèmes sont composés d'un nombre fini d'états. Le système évolue dans ces états à temps discret. L'évolution du système à chaque itération est donnée par des lois de probabilités. Concrètement, on a une distribution initiale. Cette distribution donne les probabilités respectives que le système se trouve dans les différents états. Puis, on définit des probabilités entre les états, i.e. les probabilités que le système, s'il est dans l'état i passe dans l'état j . Toutes ces informations peuvent être résumées à partir d'une matrice de transition P , matrice carrée vérifiant certaines propriétés. L'élément P_{ij} de P donne la probabilité que le système étant dans l'état i passe dans la prochaine itération à l'état j . La matrice de transition P a un sens précis, la théorie mathématique est basée sur elle. Mais cela n'est pas l'objet de ce chapitre, nous voulons simplement montrer comment avec PSTricks il est possible de dessiner des graphes représentatifs de chaînes de Markov.

4.1 La commande `\psmatrix`

`\psmatrix` est une commande très générale qui offre beaucoup de possibilités. Nous nous limitons cependant à la réalisation de graphes représentatifs. L'implémentation PSTricks offre un vaste spectre de possibilités. L'apprentissage de l'utilisation de PSTricks se fait par la pratique. Dans cette idée, nous allons expliquer l'utilisation de `\psmatrix` par construction d'un exemple concret qui explique la structure générale de cette commande.

4.1.1 Remarques conceptuelles

- `\psmatrix` est en fait un environnement comme on verra par la suite. En effet, écrire
`\psmatrix ... code ... \endpsmatrix`
est équivalent à écrire
`\begin{psmatrix} ... code ... \end{psmatrix}`.
- Il s'ensuit que pour la commande `\psmatrix`, point de `pspicture` nécessaire. Nous ne traçons non plus de grillage puisqu'ici, il ne sert à rien.

Exemple 4.1.

```
\psmatrix
1 \\
2 & 3 \\
\endpsmatrix
```

1

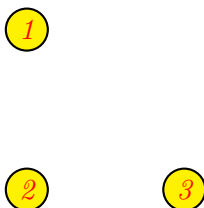
2

3

La syntaxe de `\psmatrix` est très semblable à celle pour les tableaux.

Exemple 4.2.

```
\psmatrix[fillstyle=solid,
fillcolor=yellow, mnode=circle]
\red 1 \\
\red 2 & \red 3 \\
\endpsmatrix
```



L'option `mnode` définit l'allure des sommets, en voici les plus importants possibilités :

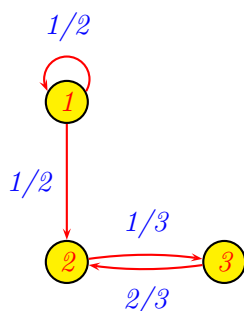
<code>circle</code>	pour un cercle	<code>oval</code>	Un oval, meilleur pour des inscriptions plus longues.
<code>dia</code>	pour un diamant	<code>tri</code>	pour un triangle

Continuons avec les arcs qui relient les sommets :

Exemple 4.3.

```
\psmatrix[fillstyle=solid, fillcolor=yellow, mnode=circle]
\red 1 \\\
\red 2 & \red 3 \\\
\endpsmatrix

\psset{linecolor=red, arrows=->}
\blue
\nccircle{1,1}{0.5cm}\taput{1/2}
\ncline{1,1}{2,1}\tlput{1/2}
\ncarc{2,1}{2,2}\taput{1/3}
\ncarc{2,2}{2,1}\tbput{2/3}
```



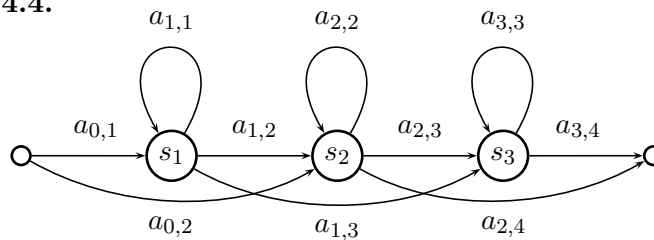
- La commande `\psset` permet de définir les caractéristiques des arcs.
- La flèche choisie est appropriée pour des chaînes de Markov.

- `\nccircle` crée un arc allant d'un sommet à lui même, `\ncline` relie deux sommets.
- Avec les commandes `\taput`, `\tbput`, `\tlput` ou `\trput`, on décide où est écrit le texte, en dessus, en dessous, à gauche ou à droite de l'arc.

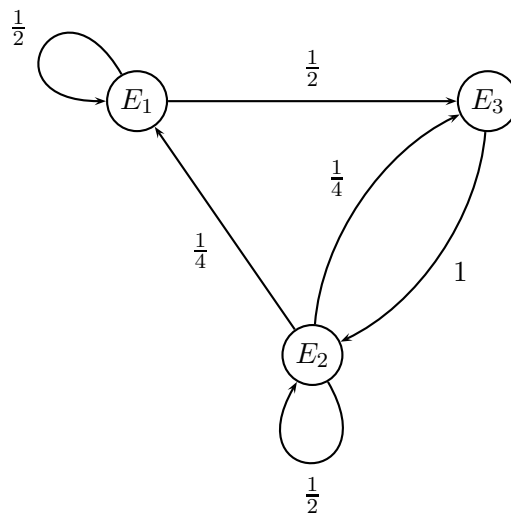
Avec ceci, nous avons tous les outils nécessaires pour dessiner des graphes représentatifs de chaînes de Markov.

Voici encore deux exemples conclusifs. Ils utilisent plus de spécifications de PSTricks que nous avons énoncés dans ce chapitre, mais illustrent bien les possibilités que offre l'environnement psmatrix.

Exemple 4.4.



Exemple 4.5.



Bibliographie

- [1] Ahmad Daaboul. Cours de PSTricks. <http://www-igm.univ-mlv.fr/daaboul/>.
- [2] Denis Girou. Présentation de PSTricks. *Cahiers GUTenberg*, (16) :21–70, Février 1994.
- [3] Indian TeX User Group. Seventh chapter of Online Tutorial on LaTeX Graphics. <http://sarovar.org/projects/pstricks/>.
- [4] Timothy Van Zandt. Pstricks : PostScript macros for Generic TeX, user's guide, March 1993. <http://www.tug.org/applications/PSTricks/>.