

# L<sup>A</sup>T<sub>E</sub>X et mathématiques

CHRONIQUES DE  
FRANÇOIS HACHE

<http://latexetmath.canalblog.com>

SAISON 2

*L'ironie n'est souvent qu'une pudeur de la tendresse.*  
*Étienne Rey*

## Chronique 1

# Couleurs en L<sup>A</sup>T<sub>E</sub>X

Comment gérer les couleurs en L<sup>A</sup>T<sub>E</sub>X ?

Voici un sujet que l'on va traiter ici de manière presque exhaustive.

### 1.1 Un peu de théorie

#### 1.1.1 Synthèse additive

En synthèse additive, on part du noir (absence de toute couleur) puis on ajoute soit du rouge, soit du vert, soit du bleu ou toute combinaison de ces trois couleurs.

L'addition du vert et du bleu donne le cyan, celle du rouge et du bleu donne le magenta, celle du rouge et du vert donne le jaune. Enfin l'addition du rouge, du vert et du bleu donne le blanc.

#### 1.1.2 Synthèse soustractive

En synthèse soustractive, on part du blanc et on filtre pour obtenir toutes les couleurs par soustraction :

- un filtre cyan laisse passer le vert et le bleu et élimine la composante rouge ;
- un filtre magenta laisse passer le rouge et le bleu et élimine la composante verte ;
- un filtre jaune laisse passer le rouge et le vert et élimine la composante bleue.

On dit que les couleurs rouge et cyan, vert et magenta, bleu et jaune, sont complémentaires.

#### 1.1.3 Couleurs primaires

On appelle donc couleurs primaires soit les couleurs rouge-vert-bleu, soit les couleurs cyan-magenta-jaune. Ces deux systèmes permettent de reconstruire toutes les autres couleurs.

### 1.2 Package color

Tout d'abord, on charge l'extension `color` en entrant `\usepackage{color}` dans le préambule du document, si ce n'est pas déjà fait.

Ou pas ! comme on verra au début du paragraphe suivant...

### 1.2.1 Couleurs prédéfinies

Le package `color` définit six couleurs :

red	<span style="background-color: red; color: black;">rouge</span>	green	<span style="background-color: green; color: black;">vert</span>	blue	<span style="background-color: blue; color: black;">bleu</span>
cyan	<span style="background-color: cyan; color: black;">cyan</span>	magenta	<span style="background-color: magenta; color: black;">magenta</span>	yellow	<span style="background-color: yellow; color: black;">jaune</span>

auxquelles on peut rajouter le noir (`black`) et le blanc (`white`).

### 1.2.2 Instructions

#### `\color`

C'est l'instruction `\color` qui fixe la couleur du texte courant ; pour écrire un paragraphe en rouge, il suffit de taper :

```
\color{red}                                % Pour écrire en rouge
Texte écrit en rouge
Texte toujours écrit en rouge
\color{black}                             % Pour revenir en noir
```

Un seul paramètre pour `\color` : le nom de la couleur que l'on veut utiliser.

#### `\textcolor`

Quand on n'a qu'un mot ou un bout de phrase à écrire dans une autre couleur, par exemple en rouge, on utilise plutôt `\textcolor` comme instruction.

Voici ce qu'il faut taper pour obtenir cette phrase :

```
Quand on n'a qu'un mot ou un bout de phrase à écrire dans une autre
couleur, \textcolor{red}{par exemple en rouge}, on utilise plutôt
\textcolor comme instruction.
```

Deux paramètres pour `\textcolor` : le nom de la couleur à utiliser, le texte que l'on veut écrire dans cette couleur.

#### `\pagecolor`

Pas de surprise pour cette instruction, elle remplit la page de la couleur que l'on veut :

```
\pagecolor{rouge} % applique un fond rouge à la page courante
```

Un seul paramètre pour `\pagecolor` : le nom de la couleur que l'on veut utiliser comme fond de page.

#### `\definecolor`

Comme on peut trouver que six couleurs ce n'est pas suffisant, on peut en redéfinir d'autres et les utiliser avec les instructions `\color`, `\textcolor`, `\pagecolor` ou celles que l'on verra par la suite.

L'instruction `\definecolor` est basée sur trois modèles que l'on va détailler ; le nom du modèle se place en deuxième paramètre.

##### `gray`

Ce modèle permet de définir de nouvelles nuances de gris et s'utilise ainsi :

```
\definecolor{nom}{gray}{coefficient}
```

Le `coefficient` est un nombre décimal compris entre 0 et 1 qui indique le pourcentage de blanc dans le noir ; ainsi le coefficient 0 donnera du noir, et le coefficient 1 donnera du blanc.

Par exemple `\definecolor{grisfoncé}{gray}{0.35}` définit la couleur `grisfoncé`.

```
\textcolor{grisfoncé}{gris foncé} donne gris foncé
```

**rgb pour Red Green Blue**

Ce modèle permet de définir une couleur en fonction de ses composantes en rouge, vert, bleu (couleurs primaires) :

```
\definecolor{nom}{rgb}{coeff_rouge, coeff_vert, coeff_bleu}
```

Les trois coefficients sont des nombres décimaux compris entre 0 et 1 qui indiquent respectivement le pourcentage de rouge, de vert et de bleu ; ces coefficients doivent être séparés par une virgule et la somme des trois nombres peut dépasser 1. Ainsi {1, 0, 0} donne le rouge, {0, 1, 0} donne le vert et {0, 0, 1} donne le bleu, {0, 1, 1} donne le cyan, {1, 0, 1} donne le magenta et {0, 1, 1} donne le jaune.

Par exemple `\definecolor{couleur1}{rgb}{0.5,0.5,0}` définit la couleur `couleur1` et `\definecolor{couleur2}{couleur2}{0,0.5,0.5}` définit la couleur `couleur2`.

```
\textcolor{couleur1}{couleur 1} donne couleur 1
```

```
\textcolor{couleur2}{couleur 2} donne couleur 2
```

Enfin `\definecolor{couleur3}{rgb}{1,1,1}` donne le blanc et

```
\definecolor{couleur4}{rgb}{0,0,0} donne le noir.
```

Mais comment s'y retrouver dans toutes ces combinaisons de couleurs ?

Je vous conseille le très bon document de ARNAUD GAZAGNES qui s'intitule :

**L<sup>A</sup>T<sub>E</sub>X... pour le prof de maths !**

Au paragraphe 3.12 de la version du 17 mars 2013, vous verrez un bel aperçu de couleurs. On peut télécharger le fichier en pdf à l'adresse :

<http://math.univ-lyon1.fr/irem/IMG/pdf/LatexPourProfMaths.pdf>

**cmyk pour Cyan Magenta Yellow black**

On peut aussi utiliser le deuxième système de couleurs primaires `cyan`, `magenta`, `jaune` (`yellow`) auxquelles on a rajouté le noir (`black` représenté par la lettre `k`) ; d'où le modèle `cmyk`.

Ce modèle permet de définir une couleur en fonction de ses composantes en cyan, magenta, jaune et noir :

```
\definecolor{nom}{cmyk}{c_cyan, c_magenta, c_jaune, c_noir}
```

Tout fonctionne exactement comme avec le modèle `rgb` sauf qu'il faut quatre coefficients entre 0 et 1 comme troisième paramètre au lieu de trois.

**Précisions sur `\color` et `\textcolor`**

Maintenant qu'on a vu la création de couleur en utilisant un modèle, on va voir que l'on peut utiliser une couleur sans définir son nom, par exemple si on ne veut l'utiliser qu'une seule fois ; il y a en effet une autre syntaxe pour les deux instructions `\color` et `\textcolor`.

- Pour `\color`

`\color[gray]{k}` définit comme couleur par défaut la couleur gris contenant  $k\%$  de blanc ;

`\color[rgb]{a b c}` définit comme couleur par défaut la couleur contenant  $a\%$  de rouge,  $b\%$  de vert et  $c\%$  de bleu ;

`\color[cmyk]{a b c d}` définit comme couleur par défaut la couleur contenant  $a\%$  de cyan,  $b\%$  de magenta,  $c\%$  de jaune et  $d\%$  de noir.

On remet le noir par défaut en tapant l'instruction `\color{black}`.

- Pour `\textcolor`

ça fonctionne exactement de la même façon que `color` :

l'instruction `\textcolor[rgb]{0.8 0 0}{texte}` écrit le mot `texte` en rouge foncé.

### `\colorbox`

L'instruction `\colorbox` dessine une boîte dont on peut déterminer la couleur de fond et dans laquelle on écrit un texte qui est dans la couleur courante :

`\colorbox{couleur}{texte}`

Ainsi `\colorbox{yellow}{texte}` va donner texte ce qui peut simuler un surlignage fluo.

Deux paramètres pour `\colorbox` : le nom de la couleur que l'on veut utiliser comme fond pour la boîte et le texte que l'on veut écrire dans la boîte.

On peut combiner `\colorbox` avec `\textcolor` par exemple pour écrire un texte blanc sur fond rouge ; ainsi `\colorbox{red}{\textcolor{white}{texte blanc sur fond rouge}}` donne :

texte blanc sur fond rouge

### `\fcolorbox`

Un peu plus complète que `\colorbox`, l'instruction `\fcolorbox` dessine une boîte dont on peut déterminer la couleur du cadre, la couleur de fond et dans laquelle on écrit un texte qui est dans la couleur courante ; on peut retenir que la lettre `f` de `\fcolorbox` signifie **f**rame (**c**adre) :

`\fcolorbox{couleur du cadre}{couleur du fond}{texte}`

Ainsi `\fcolorbox{black}{yellow}{texte}` va donner texte.

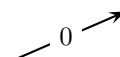
Ce sont donc trois paramètres qui suivent l'instruction `\fcolorbox` : le nom de la couleur que l'on veut utiliser comme cadre pour la boîte, le nom de la couleur que l'on veut comme fond et le texte à encadrer.

On peut également combiner `\fcolorbox` avec `\textcolor` par exemple pour écrire un texte bleu sur fond vert, avec un cadre noir ;

`\fcolorbox{black}{green}{\textcolor{white}{texte bleu sur fond vert}}` donne :

texte bleu sur fond vert

J'utilise aussi `\fcolorbox` pour dessiner ceci :



pour signifier dans un tableau de variations qu'une fonction continue et strictement croissante passe par 0 ; il suffit de placer au bon endroit une boîte dont le fond et le cadre sont blancs :

`\psline[arrowsize=3pt 2]{->}(-0.5,-0.2)(1,0.4)\fcolorbox{white}{white}{\$0\$}`

Ne pas oublier que `\psline` est une instruction qui nécessite l'extension `pstricks` dont on va (re)parler maintenant.

## 1.3 Package `pstricks-add`

Quand on dessine des figures de géométrie ou qu'on trace des courbes, on peut utiliser l'extension `pstricks` (c'est mon cas) ; il suffit d'entrer dans le préambule : `\usepackage{pstricks-add}` (version complète de `pstricks`). On en a déjà parlé.

Pourquoi cette extension dans une chronique consacrée aux couleurs ?

Parce qu'elle contient (ou charge) l'extension `color` et permet des possibilités supplémentaires ; il est donc inutile de charger l'extension `color` si on travaille en `pstricks` !

La conséquence de cette compatibilité fait que tout ce qui a été dit dans le paragraphe précédent reste vrai en ayant chargé l'extension `pstricks-add` à la place de `color`.

### 1.3.1 Couleurs prédéfinies et utilisation

En plus des six couleurs définies par l'extension `color`, du noir et du blanc, l'extension `pstricks-add` définit deux niveaux de gris : `darkgray` (gris foncé) et `lightgray` (gris clair).

Une première amélioration de l'extension `pstricks-add` est que l'on peut se passer de l'instruction `\textcolor`; en effet, pour écrire un mot ou un bout de phrase dans une autre couleur que celle du texte courant, il suffit de placer le texte entre accolades et de définir la couleur au début des accolades :

```
{\darkgray gris foncé} écrira gris foncé
```

Mais on ne peut utiliser cette fonctionnalité qu'avec les couleurs prédéfinies et pas avec les couleurs définies par `\definecolor`.

Mais peut-être y a-t-il un nouveau mode de définition de couleurs avec `pstricks-add` et que cette technique fonctionnera avec ces couleurs définies d'une nouvelle façon ?

### 1.3.2 Définition de nouvelles couleurs

Quatre nouvelles instructions permettent de définir de nouvelles nuances de gris ou de nouvelles couleurs sous `pstricks-add`.

#### `\newgray`

Cette instruction permet de définir ses propres nuances de gris; elle nécessite deux paramètres : le nom de la couleur, le pourcentage de blanc que l'on met dans le noir pour faire du gris.

```
\newgray{mongris}{0.2} définit un gris à 20 %.
```

On l'utilise ainsi :

```
{\mongris texte en gris} donne texte en gris
```

#### `\newrgbcolor`

C'est l'instruction que j'utilise pour créer de nouvelles couleurs.

Elle nécessite deux paramètres : le nom de la nouvelle couleur, les trois composantes rouge-vert-bleu sous forme de trois nombres décimaux entre 0 et 1 séparés par un espace.

Ainsi `\newrgbcolor{brun}{0.6 0.2 0}` définit le **brun** que l'on écrit `{\brun brun}`.

L'instruction `\renewrgbcolor` n'existe pas; si on souhaite modifier la couleur brun, par exemple en y rajoutant un peu de rouge, il suffit de la redéfinir :

```
\newrgbcolor{brun}{0.7 0.2 0}
```

#### `\newcmykcolor`

Voici la syntaxe de l'instruction `\newcmykcolor` :

```
\newcmykcolor{nom_couleur}{c_cyan c_magenta c_jaune c_noir}
```

où `c_cyan`, `c_magenta`, `c_jaune` et `c_noir` sont respectivement les coefficients de cyan, de magenta, de jaune et de noir dans la nouvelle couleur.

Le mode d'emploi officiel de `pstricks-add` précise que la définition des couleurs au moyen de cette instruction peut poser des problèmes en PostScript; laissons tomber !

#### `\newhsbcolor`

Voici la syntaxe de l'instruction `\newhsbcolor` :

```
\newhsbcolor{nom_couleur}{teinte saturation intensité}
```

où `teinte` (Hue), `saturation` (Saturation) et `intensité` (Brightness) sont des nombres décimaux entre 0 et 1.

Et comme la documentation officielle de `pstricks-add` qualifie l'emploi de l'instruction `\newhsbcolor` de *not recommended*, on ne passera pas plus de temps sur le sujet...

## 1.4 Noir et blanc

Je vous livre une petite astuce que j'emploie de temps en temps ; on l'améliorera dans une future chronique avec des définitions de variables.

On souhaite faire un petit lexique de formules mathématiques rangées dans un tableau et faire réciter ces formules à des élèves en ne leur donnant qu'une partie du tableau.

On crée un tableau avec quelques formules :

Forme développée	Forme factorisée
$a^2 + 2ab + b^2$	$= (a + b)^2$
$a^2 - 2ab + b^2$	$= (a - b)^2$
$a^2 - b^2$	$= (a + b)(a - b)$

et on voudrait rapidement présenter à des élèves ce tableau à remplir :

Forme développée	Forme factorisée
$a^2 + 2ab + b^2$	$=$
$a^2 - 2ab + b^2$	$=$
$a^2 - b^2$	$=$

sans rien effacer, bien sûr !

Il suffit de définir une couleur (`\hhh`), d'écrire le texte de la colonne à cacher (la troisième ici) dans cette couleur et de mettre alternativement cette couleur soit en noir, soit en blanc :

```
\newrgbcolor{hhh}{0 0 0} % (0 0 0) pour noir et (1 1 1) pour blanc
\renewcommand{\arraystretch}{1.5} % augmente la hauteur des lignes
$\begin{array}{r c l}
\hline
\textbf{Forme développée} & & \textbf{Forme factorisée} \\
\hline
a^2+2ab+b^2 & = & \hhh (a+b)^2 \\
a^2-2ab+b^2 & = & \hhh (a-b)^2 \\
a^2-b^2 & = & \hhh (a+b)(a-b) \\
\hline
\end{array} $
\renewcommand{\arraystretch}{1} % remet les lignes à la hauteur standard
```

Vous avez peut-être remarqué que je n'ai pas écrit `\hhh expression` entre accolades, mais `\hhh expression` sans les accolades ; elles sont en effet superflues dans une cellule d'un tableau.

Et vous avez certainement compris ce qu'il fallait modifier pour pouvoir obtenir :

Forme développée	Forme factorisée
$a^2 + 2ab + b^2$	$= (a + b)^2$
$a^2 - 2ab + b^2$	$= (a - b)^2$
$a^2 - b^2$	$= (a + b)(a - b)$

On peut utiliser ce truc pour créer des textes « à trous ».



## Chronique 2

# Maître et esclave

### 2.1 Principe

Pour construire un cours contenant plusieurs chapitres et donc quelques dizaines de pages, on peut choisir de créer un document en classe `report` ou en classe `book`. L'inconvénient est que l'on travaille toujours avec la totalité des chapitres, que l'on ajoute au fur et à mesure au document. On peut aussi créer un document de classe `article` par chapitre; l'inconvénient est alors que si on veut numérotter l'ensemble du document constitué de tous les chapitres, il faudra gérer soi-même les numéros de pages, et on ne pourra créer ni table des matières ni index pour le document complet. Une solution ?

C'est celle que j'ai utilisée dans la première saison de mon blog : j'ai créé un « document maître » qui appelle les « documents esclaves » correspondant aux chapitres.

Et quand le document maître est complet, je demande une table des matières.

On va voir ça de plus près.

### 2.2 Un document esclave

Recette pour créer un document esclave :

- prendre un document, le mettre au point, le corriger et quand le résultat convient, l'enregistrer sous le nom `chap01.tex`, par exemple ;
- placer le caractère `%` devant toutes les lignes, de la première à `\begin{document}` comprise ;
- placer le caractère `%` devant la ligne `\end{document}` ;
- enregistrer ce nouveau fichier sous un autre nom : `_chap01.tex` par exemple.

Le document esclave est prêt ; on ne peut naturellement pas le compiler car il manque tout le préambule, ainsi que `\begin{document}` et `\end{document}`.

On peut également effacer les lignes au lieu de les neutraliser au moyen du signe `%`, mais je ne le conseille pas car c'est alors irréversible (du moins quand on ouvre une nouvelle session).

Il peut sembler fastidieux de placer le caractère `%` devant les quelques dizaines de lignes que constitue le préambule ; mais ça ne l'est pas du tout avec **Texmaker** : en effet, il suffit de sélectionner les lignes en question et de taper simultanément sur la touche **Ctrl** et sur le caractère **T**.

Pratique, non ?

Regardez si votre éditeur **L<sup>A</sup>T<sub>E</sub>X** fait ça également.

Et pour retirer le caractère `%` à une série de lignes, il suffit de les sélectionner et de taper simultanément sur la touche **Ctrl** et sur le caractère **U**.

On trouve ça dans Editer-Commenter (**Ctrl**+**T**)-Décommenter (**Ctrl**+**U**) du menu de **Texmaker**.

## 2.3 Le document maître

Le document maître est le document principal ; il doit contenir le préambule qui va être utilisé par chaque document esclave que l'on va intégrer ; donc dès qu'on a besoin d'une nouvelle extension dans un document esclave, il faut la rajouter dans le document maître.

Ensuite il faut que le document maître appelle les documents esclaves ; pour intégrer le fichier `_chap01.tex` dans le document maître, il suffit de taper l'instruction :

```
\input{_chap01.tex}
```

Pour obtenir le résultat espéré, il faut que le fichier esclave soit dans le même répertoire que le document maître. Si ce fichier esclave se trouve dans le sous-répertoire `Chapitres`, il faudra taper :

```
\input{Chapitres/_chap01.tex}
```

On peut indiquer à **Texmaker** quel est le document maître ; on choisit dans son menu :

- Options
- Définir le document courant comme document 'maître'

Ainsi, quel que soit le document sur lequel on travaille, ce sera le document maître qui sera compilé lors du lancement de la compilation ; on peut donc corriger un document esclave et lancer la compilation, il n'y aura pas de message d'erreur.

Les autres éditeurs **L<sup>A</sup>T<sub>E</sub>X** (que je ne connais pas) doivent posséder cette même fonctionnalité.

Et quand on a terminé de travailler sur ces documents maître-esclaves, on remet la compilation en mode normal dans **Texmaker** ; il suffit de choisir :

- Options
- Mode normal (document maître actuel...)

## 2.4 La table des matières

C'est très simple de créer une table des matières ; il suffit de taper l'instruction `\tableofcontents` à l'endroit où on veut qu'elle apparaisse.

À l'appel de l'instruction `\tableofcontents` le moteur de **L<sup>A</sup>T<sub>E</sub>X** crée un fichier qui porte le nom du document sur lequel on travaille avec pour extension `toc` (**p**our **t**able **o**f **c**ontents). Il faut compiler deux fois le document sur lequel on travaille car lors de la première compilation, le fichier `toc` est créé, et il n'est interprété que lors de la deuxième compilation.

La table des matières est composée des titres et des numéros de pages des paragraphes commençant par un élément de structure connu par **L<sup>A</sup>T<sub>E</sub>X** : `\part`, `\section`, `\subsection`... Le `\chapter` n'est reconnu et interprété que dans un document de classe `book` ou de classe `report`.

Les versions non numérotées (`\part*`, `\section*`, `\subsection*`...) ne sont pas prises en compte dans la table des matières ; on verra comment les rajouter dans une future chronique.

On y verra également comment personnaliser cette table des matières et comment créer un index.

## Chronique 3

# Insertion d'image

### 3.1 Mode de compilation

Pour visualiser un document tapé en  $\text{\LaTeX}$ , il y a plusieurs façons. Moi je visualise en PostScript (au moyen de `Gsview` et `Ghostscript`). Ce mode de compilation et de visualisation permet aussi d'utiliser `PsTricks` sans problème.

On peut également visualiser en pdf ; mais là, il y a deux modes de compilation : soit par `PdfLatex`, soit par `ps2pdf` en passant par `dvips`.

Ce que je vais expliquer dans cette rubrique ne fonctionne qu'en PostScript ou en pdf compilé par `ps2pdf` (donc en passant également par PostScript).

L'éditeur que j'utilise, `Texmaker`, doit donc être configuré correctement ; on choisit dans le menu un de ces deux modes :

- |                           |                                     |
|---------------------------|-------------------------------------|
| • Options                 | • Options                           |
| • Configurer Texmaker     | • Configurer Texmaker               |
| • Compil rapide           | • Compil rapide                     |
| • LaTeX + dvips + View PS | • LaTeX + dvips + ps2pdf + View pdf |

Ensuite l'appui sur `F1` entraîne la compilation et affiche le résultat attendu.

### 3.2 Flottant ou pas

Il existe, en  $\text{\LaTeX}$ , deux modes de gestion des images : le mode flottant et le mode non flottant.

En fait la notion de « flottant » est plus générale, et on doit plutôt parler d'« objet flottant », l'objet en question pouvant être une image, une photo, une table...

On peut même rendre flottant n'importe quel objet.

Un objet flottant sera placé « au mieux » par  $\text{\LaTeX}$ , c'est-à-dire qu'il ne sera pas forcément à l'endroit où cet objet est inséré dans le texte source ; ainsi il peut être plus bas dans la page ou encore la page suivante, notamment s'il y a déjà un objet flottant dans la page courante.

Ce n'est donc pas de ce mode dont je vais parler dans cette chronique car je préfère que les figures soient à l'endroit précis où je le souhaite.

### 3.3 Format EPS

Le format de sortie originel de  $\text{\TeX}$  (puis celui de  $\text{\LaTeX}$ ) était le POSTSCRIPT (PS) ; le format d'images associé est l'ENCAPSULÉ POSTSCRIPT (EPS).

Il est donc tout à fait naturel que l'insertion d'image EPS dans un document écrit en  $\text{\LaTeX}$  se fasse simplement ; une seule instruction suffira.

### 3.3.1 Création d'une image EPS avec GeoGebra

Le très bon logiciel de géométrie GeoGebra permet d'exporter des graphiques et d'en faire des images EPS que l'on pourra directement insérer dans un texte en  $\text{\LaTeX}$ .

On crée une figure quelconque (fonction carré, un cercle et un trapèze rectangle), avec des couleurs, pas seulement pour faire plus joli !

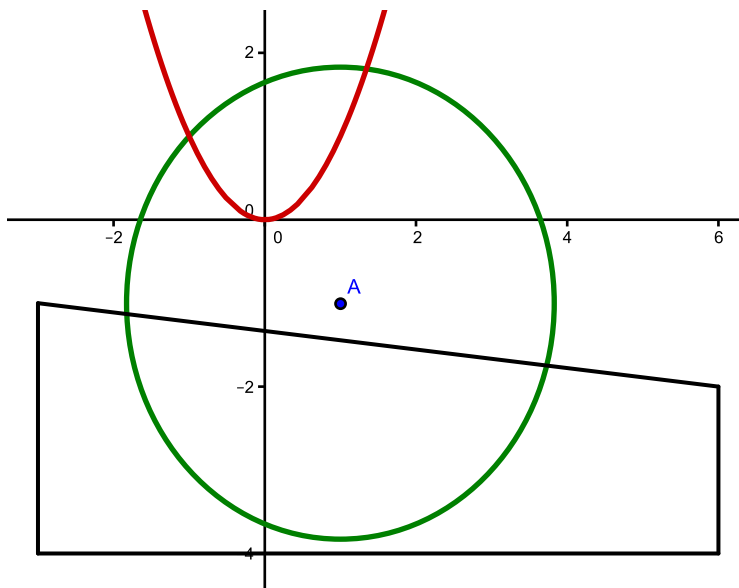
À la souris, on sélectionne la partie du graphique à découper puis on choisit :

- Fichier
- Exporter      ►
- Graphique en tant qu'image (png, eps)
- Format Postscript encapsulé (eps)
- on laisse 1 cm pour échelle
- Sauvegarder
- sous le nom de FigGeo

On a donc créé une figure qui porte le nom `FigGeo.eps` que l'on va copier dans le même répertoire que le document  $\text{\LaTeX}$  sur lequel on travaille ; si cette figure n'est pas dans le même répertoire, il faudra indiquer le chemin d'accès de cette figure.

### 3.3.2 Insertion de l'image

$\text{\LaTeX}$  sait lire les caractéristiques de toute image au format EPS ; il suffit d'une instruction pour inclure l'image FigGeo dans le texte : `\includegraphics{FigGeo}`



La figure a la taille de la figure découpée dans GeoGebra.

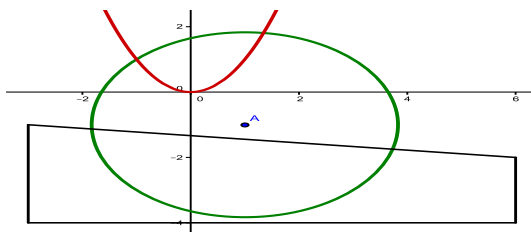
### 3.3.3 Options

Tout d'abord, pour utiliser des options avec l'instruction `\includegraphics`, il faut avoir chargé l'extension `graphicx` par un `\usepackage{graphicx}` dans le préambule du document.

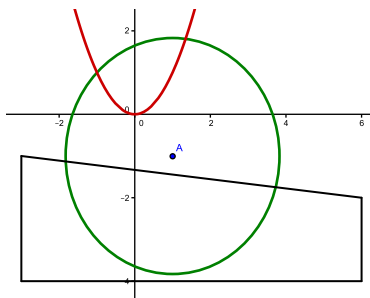
La syntaxe complète est alors : `\includegraphics[options]{nom_image}`.

On peut très facilement alors fixer la largeur (`width`) ou la hauteur (`height`) de la figure à afficher ; il suffit de rajouter des options entre crochets :

- `\includegraphics[width=7cm]{FigGeo}` impose une largeur de 7 cm ;
- `\includegraphics[height=3cm]{FigGeo}` impose une hauteur de 3 cm ;
- `\includegraphics[width=7cm, height=3cm]{FigGeo}` impose à la fois une largeur de 7 cm et une hauteur de 3 cm ; la figure est donc déformée :



Une autre façon simple de modifier les dimensions d'une figure est d'utiliser le facteur d'échelle `scale` : `\includegraphics[scale = 0.5]{FigGeo}` multiplie chaque dimension par 0,5.



On peut tourner les figures en entrant en option la valeur, en degrés, de l'angle de la rotation que l'on veut appliquer :

```
\includegraphics[scale = 0.5, angle=45]{FigGeo}
```

Et on peut même imposer le centre de la rotation en rajoutant l'option `origin = label` ou `label` est une combinaison de lettres parmi : l (left), r (right), c (center), t (top), b (bottom) ou B (baseline). L'origine par défaut est bl, en bas à gauche.

L'inclusion d'image est naturellement compatible avec le multicolonnage, ainsi qu'avec la notion de minipage (voir chroniques de la première saison).

### 3.3.4 Légende

On peut ajouter une légende très simplement à une figure.

Pour cela il faut d'abord charger l'extension `caption` par `\usepackage{caption}`.

Puis, après avoir inséré la figure, on en donne le nom au moyen de l'instruction `\captionof` comme dans l'exemple :

```
\begin{center}
\includegraphics[height=4cm]{FigGeo}
\captionof{figure}{GeoGebra}
\end{center}
```

ce qui donne :

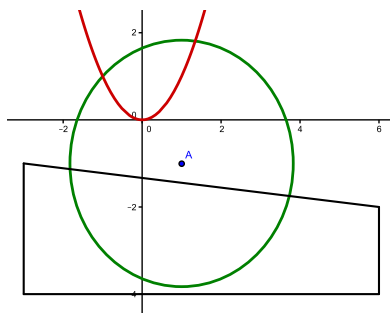


FIGURE 3.1 – GeoGebra

Le mot **figure** qui est le premier paramètre de la commande `\captionof` signifie que le nom « GeoGebra » est le nom d’une figure et apparaîtra ainsi dans une éventuelle table des figures.

Vous remarquerez que devant le mot **GeoGebra** se trouve le mot **FIGURE** (en petites capitales) suivi de deux nombres séparés par un point : à gauche du point se trouve le numéro du chapitre en cours, et à droite du point le numéro de la figure dans le chapitre.

À condition toutefois d’être dans une classe (**report** ou **book**) dans laquelle le niveau de structure `\chapter` est reconnu ; sinon, seul le nombre à droite du point est affiché.

## 3.4 Format JPG

Disons-le tout net : la sortie en **POSTSCRIPT** n’est pas faite pour insérer des images au format **JPG**. Il faut pour cela compiler avec **pdflatex**, et tout ce qui a été dit précédemment fonctionne ; on entrera donc une commande du style : `\includegraphics[options]{nom_image.jpg}`.

Mais si l’on est très têtue et que l’on veut à tout prix insérer une image au format **JPG** dans un fichier visualisé en **POSTSCRIPT** (par exemple si on travaille avec **PsTricks** qui n’aime pas une compilation en **pdflatex**) ?

### 3.4.1 Insertion de l’image

On place une photo intitulée **Hoss01.jpg** dans le répertoire de ce document source et on entre :

```
\includegraphics{Hoss01.jpg}
```

On obtient un magnifique message d’erreur :

```
Cannot determine size of graphic in Hoss01.jpg (no BoundingBox)
```

En effet, **L<sup>A</sup>T<sub>E</sub>X** sait lire les caractéristiques d’une image en **EPS**, mais pas la taille d’une image en **JPG**. Il faut donc passer en paramètre les dimensions de la photo, et en même temps une largeur pour que la photo rentre dans la page !

Cette photo est de 490 pixels sur 368 pixels, donc on entrera :

```
\begin{center}
\includegraphics[bb=0 0 490 368, width=8cm]{Hoss01.jpg}
\captionof{figure}{Café de Paris}
\end{center}
```

Vous aurez compris que le **bb** signifie **BoundingBox**, et que l’on indique ainsi à **L<sup>A</sup>T<sub>E</sub>X** les dimensions de la photo à insérer.

Et que se passe-t-il alors ?



FIGURE 3.2 – Café de Paris

Catastrophe! Les couleurs de la photo ont disparu et je ne connais pas de moyen d'empêcher ça! On oubliera donc l'insertion d'image JPG dans un texte compilé en POSTSCRIPT. Mais peut-être qu'on peut convertir le format JPG en format EPS?

### 3.4.2 Conversion de format

J'ai trouvé sur Internet un logiciel de dessin vectorisé qui permet (entre autres) de passer du format JPG au format EPS; il s'agit de `inkscape` que l'on peut télécharger (gratuitement) à l'adresse <http://inkscape.org/?lang=fr>

Il y a une grosse documentation pour ce logiciel sur Internet, la plupart du temps en anglais. Moi je ne l'ai utilisé que pour convertir les formats : j'ouvre un fichier JPG et je l'exporte en EPS en utilisant `Enregistrer sous`.

Seul défaut, le fichier EPS obtenu est dix fois plus volumineux que le fichier JPG d'origine.

Après transformation du format si on entre :

```
\includegraphics[width=8cm]{Hoss01}
\captionof{figure}{Café de Paris}
```

on obtient :



FIGURE 3.3 – Café de Paris

C'est quand même mieux en couleur !





## Chronique 4

# Table des matières

Petit guide pour utiliser au mieux une table des matières.

### 4.1 Classe book

Pour qu'une table des matières vaille la peine, il faut un document avec suffisamment de pages et suffisamment de niveaux de structure; on va donc utiliser un document de classe **book**.

Les éléments de structure d'un tel document sont dans l'ordre décroissant d'importance :

- `\part`
- `\chapter`
- `\section`
- `\subsection`
- `\subsubsection`
- `\paragraph`
- `\subparagraph`

J'ai donc créé un document bidon de classe **book** contenant tous les éléments de structure possibles dans cette classe pour en extraire une (ou plusieurs) table(s) des matières; au niveau `\part` j'ai entré `\part{Partie}`, au niveau `\chapter` j'ai entré `\chapter{Chapitre}`, etc. jusqu'au niveau `\subparagraph` où j'ai entré `\subparagraph{Sous-paragraphe}`.

On pourrait aussi utiliser un document de classe **report** qui possède les mêmes éléments de structure. Avec un document de classe **article**, il n'y a pas de `\chapter` donc la numérotation se fait de façon différente, ce qui a naturellement des répercussions sur la table des matières.

### 4.2 Principe

Il n'y a pas d'extension particulière à charger pour obtenir une table des matières, une seule instruction suffit :

`\tableofcontents`

En classe **book**, une table des matières va se construire dans une nouvelle page qui sera forcément une page impaire; en effet les documents de classe **book** sont destinés à être imprimés en recto-verso, chaque chapitre commençant automatiquement sur une page de droite (impaire), tout comme la table des matières. Ce n'est pas le cas dans la classe **report** pour laquelle la table des matières démarrera sur une nouvelle page, pas forcément de numérotation impaire. Enfin, en classe **article**, la table des matières démarre exactement à l'endroit où on l'appelle.

Pour obtenir la table des matières, il faut compiler le document deux fois ; lors du premier passage, un fichier d'extension `toc` (`table of contents`) est créé et il n'est interprété que lors de la seconde compilation.

Avec `Texmaker` il suffit d'appuyer sur F2 pour la première compilation `LaTeX`, puis normalement sur F1 pour la seconde.

Après les deux compilations, voici ce qu'on obtient :

# Table des matières

<b>I</b>	<b>Partie</b>	<b>1</b>
<b>1</b>	<b>Chapitre</b>	<b>3</b>
1.1	Section . . . . .	3
1.2	Section . . . . .	3
1.2.1	Sous-section . . . . .	4
1.3	Section . . . . .	5
1.4	Section . . . . .	6
<b>2</b>	<b>Chapitre</b>	<b>7</b>
2.1	Section . . . . .	7
2.1.1	Sous-section . . . . .	7
2.1.2	Sous-section . . . . .	8
2.2	Section . . . . .	8
2.2.1	Sous-section . . . . .	8
2.2.2	Sous-section . . . . .	8
<b>II</b>	<b>Partie</b>	<b>9</b>
<b>3</b>	<b>Chapitre</b>	<b>11</b>
<b>4</b>	<b>Chapitre</b>	<b>13</b>

Le nom de la table des matières est « Table des matières » car on a chargé un module français dans le préambule avec `\usepackage[français]{babel}`.

Ce qui est plus surprenant dans cette table des matières, c'est que n'apparaissent comme éléments de structure aucune des entrées de niveau `\subsubsection`, `\paragraph` ou `\subparagraph`.

`LaTeX` doit considérer que ça suffit le plus souvent !

## 4.3 Niveau de profondeur

Comment modifier le niveau de profondeur d'une table des matières ?

C'est très simple ; ce niveau de profondeur est contrôlé par un compteur `tocdepth` qui vaut 2 par défaut et que l'on gère avec l'instruction `\setcounter`.

Autrement dit, avant d'appeler la table des matières :

- entrer `\setcounter{tocdepth}{2}` ne sert à rien ;
- entrer `\setcounter{tocdepth}{3}` augmente le niveau de 1 ;
- entrer `\setcounter{tocdepth}{4}` augmente le niveau de 2 ;
- etc.
- entrer `\setcounter{tocdepth}{1}` diminue le niveau de 1.
- etc.

Plus précisément, en utilisant mon fichier bidon, entrer :

```
\setcounter{tocdepth}{5}
\tableofcontents
```

produira :

# Table des matières

<b>I</b>	<b>Partie</b>	<b>1</b>
<b>1</b>	<b>Chapitre</b>	<b>3</b>
1.1	Section . . . . .	3
1.2	Section . . . . .	3
1.2.1	Sous-section . . . . .	4
	Sous-sous-section . . . . .	5
	Sous-sous-section . . . . .	5
	Paragraphe . . . . .	5
	Sous-paragraphe . . . . .	5
	Sous-paragraphe . . . . .	5
	Paragraphe . . . . .	5
1.3	Section . . . . .	5
1.4	Section . . . . .	6
<b>2</b>	<b>Chapitre</b>	<b>7</b>
2.1	Section . . . . .	7
2.1.1	Sous-section . . . . .	7
2.1.2	Sous-section . . . . .	8
2.2	Section . . . . .	8
2.2.1	Sous-section . . . . .	8
2.2.2	Sous-section . . . . .	8
<b>II</b>	<b>Partie</b>	<b>9</b>
<b>3</b>	<b>Chapitre</b>	<b>11</b>
<b>4</b>	<b>Chapitre</b>	<b>13</b>

On voit que tous les éléments de structure sont bien présents, de `\part` à `\subparagraph`.

Tandis qu'en entrant :

```
\setcounter{tocdepth}{0}
\tableofcontents
```

on obtiendra :

## Table des matières

<b>I</b>	<b>Partie</b>	<b>1</b>
<b>1</b>	<b>Chapitre</b>	<b>3</b>
<b>2</b>	<b>Chapitre</b>	<b>7</b>
<b>II</b>	<b>Partie</b>	<b>9</b>
<b>3</b>	<b>Chapitre</b>	<b>11</b>
<b>4</b>	<b>Chapitre</b>	<b>13</b>

On peut donc mettre une courte table des matières en début de livre, et une très complète en fin d'ouvrage (ou le contraire!).

Et en mettant `tocdepth` au niveau `-1`, on n'aura que les éléments de niveau `\part`.

## 4.4 Changements de nom

### 4.4.1 Nom de la table des matières

Et si le nom « Table des matières » ne nous convient pas ?

Facile, il suffit de dire à  $\text{\LaTeX}$  de le changer en « Sommaire » par exemple :

```
\renewcommand{\contentsname}{Sommaire}
```

On renomme tout simplement le nom de la table des matières (`contentsname`) en un autre nom.

Et si en plus on veut que le mot « Sommaire » soit centré sur la page ?

On utilise des `\hfill` de chaque côté du mot :

```
\renewcommand{\contentsname}{\hfill Sommaire \hfill\,}
```

## Sommaire

<b>I</b>	<b>Partie</b>	<b>1</b>
----------	---------------	----------

L'option `\centering` renvoie un message d'erreur.

### 4.4.2 Nom d'entrée dans la table

Si on a un titre de chapitre ou de section trop long et qu'on veut n'en écrire qu'une partie dans la table des matières, il suffit d'entrer en option ce que l'on veut écrire dans la table des matières.

Ainsi `\section[Titre court]{Titre très très long}` permettra d'écrire « Titre très très long » comme titre de rang `section` dans le corps du document, mais n'écrira que « Titre court » comme entrée correspondante dans la table des matières.

## 4.5 Entrée forcée

Si on utilise les versions étoilées des éléments de structure pour définir des titres, comme `\section*` ou `\subsection*`, le titre n'est pas numéroté et, c'est ce qui nous intéresse ici, il n'est pas répertorié dans la table des matières. Si on souhaite que ce titre apparaisse quand même dans la table des matières, il faut forcer son entrée.

C'est ce que fait l'instruction `\addcontentsline` (comme son nom l'indique!).

Cette instruction nécessite trois paramètres :

- le nom de la table dont on veut rajouter une entrée, ici `toc` ;
- le niveau de structure auquel on veut rajouter cette entrée : `part`, `chapter`, etc. ;
- le nom qui doit figurer dans la table des matières.

Par exemple, si on veut faire figurer le mot « Sommaire » au niveau `part` dans la table des matières, on entrera :

```
\addcontentsline{toc}{part}{Sommaire}
```

juste avant d'appeler la table des matières par `\tableofcontents`.

## 4.6 Le point sur le i

Un problème survient quand une section a pour nom « Le document maître » et que l'on veut une table des matières ; la première compilation n'entraîne pas de message d'erreur, mais le document ne compile plus à partir de la deuxième (et le message d'erreur n'est guère explicite!).

Cette erreur à partir de la deuxième compilation seulement, m'a fait penser qu'elle trouvait son origine dans la table des matières : en court-circuitant la ligne `\tableofcontents` la compilation n'affiche plus de message d'erreur... au deuxième passage seulement puisque le premier passage tient encore compte de l'ancienne table des matières.

En fait la table des matières n'admet pas d'entrée comportant le caractère `î` c'est-à-dire `i accent circonflexe`.

Il est donc impératif, dans un `\part{}`, `\chapter{}`, `\section{}`, etc. dont les accolades contiennent un `î`, de l'écrire `\^i` comme dans `\section{Le document ma\^itre}`.

Et tout fonctionne normalement alors !

J'ai naturellement essayé avec `â`, `ê`, `ô` et `û`, qui ne posent aucun problème : on peut définir un paragraphe par `\section{Formule du binôme}` sans avoir de message d'erreur.

Pas d'ennui non plus avec `ä`, `ë`, `ö` ou `ü`, mais le `ï` n'est pas accepté dans la table des matières.

Bizarre, bizarre !

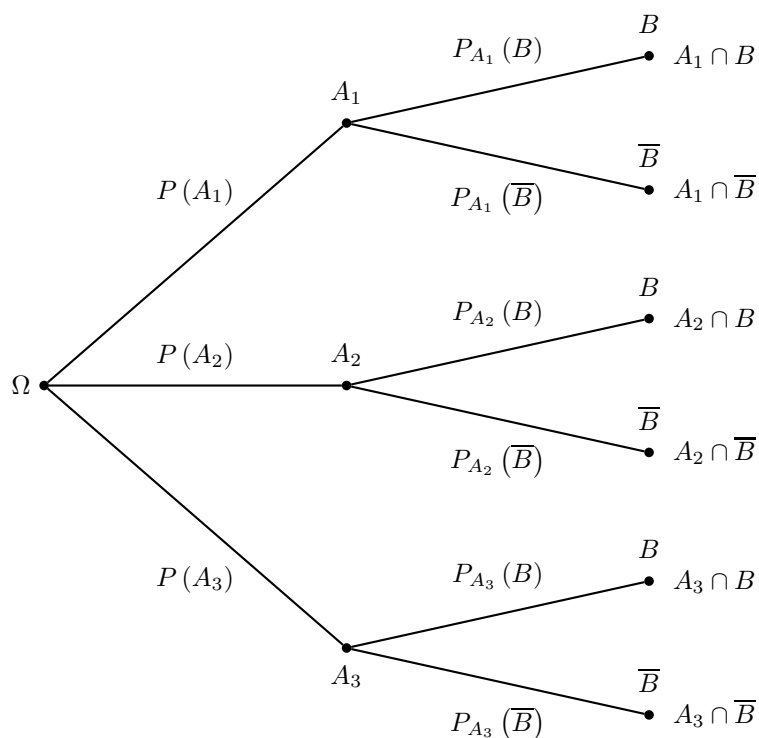


## Chronique 5

# Arbres

### 5.1 Objectif

Dans cette chronique, on va voir comment créer des arbres et comment les modifier. On aboutira à un arbre assez compliqué qui devrait couvrir quelques besoins :



Il y a des dizaines de façons de présenter des arbres, je ne vous en montrerai que quelques-unes.

### 5.2 Premiers pas

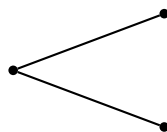
Il existe plusieurs extensions qui permettent de construire des arbres ; j'utilise `pst-tree` que je charge dans le préambule par `\usepackage{pst-tree}` (`pst` pour `PsTricks`).

On trouve la documentation de ce package (en anglais) sur l'immanquable site CTAN à l'adresse :

<http://ctan.mines-albi.fr/graphics/pstricks/contrib/pst-tree/pst-tree-doc.pdf>

Il y a même le document source `tex`.

On va commencer par un tout petit bout d'arbre :



dont voici le code (très simple) :

```
\psset{treemode=R, nodesep=0mm, levelsep=20mm, treesep=15mm}
\pstree{\Tdot}{\Tdot \Tdot}
```

En première ligne on définit les principaux paramètres de l'arbre :

- `treemode=R` désigne le sens vers lequel va être construit l'arbre : R pour **right**, L pour **left**, U pour **up** et D (valeur par défaut) pour **down** ;
- `nodesep=0mm` désigne la distance entre le nœud et le début de la branche qui va y être attachée ;
- `levelsep=20mm` désigne la distance horizontale entre deux nœuds quand l'arbre est dirigé vers la droite, et c'est aussi la longueur d'une branche horizontale ;
- `treesep=15mm` désigne la distance entre deux nœuds situés en bout d'arbre (ou à peu près!).

En deuxième ligne, on rencontre l'instruction `\pstree` qui permet tout !

Cette instruction nécessite deux paramètres ; le premier désigne ce que l'on va mettre à la racine de l'arbre, le deuxième paramètre correspond aux successeurs :

- `{\Tdot}` en premier paramètre désigne un point comme racine ;
- `{\Tdot \Tdot}` en second paramètre désigne les deux branches des successeurs.

On peut mettre d'autres choses que `\Tdot` :

- `\Tf` pour dessiner un petit carré ;
- `\TC` pour dessiner un cercle dont on fixe le rayon par exemple à 2 points en rajoutant l'option `radius=2pt` dans la ligne `\psset` ;
- `\TC*` pour dessiner un disque (dont le rayon est lui aussi géré par `radius`) ;
- `\Tp` pour ne rien dessiner du tout (mais il faut mettre quelque chose quand même!).

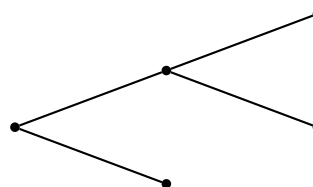
### 5.3 Arbres plus fournis

Pour faire évoluer le petit arbre précédent, il suffit de remplacer un point `\Tdot` par un sous-arbre `\pstree{}{}`.

Voici quelques variations sur le même principe.

```
\psset{treemode=R,nodesep=0mm,%
      levelsep=20mm,treesep=15mm}
      % paramètres

\pstree{\Tdot} % racine
{
  \pstree % sous-arbre supérieur
    {\Tdot} % racine
    {\Tdot \Tdot} % successeurs
  \Tdot % branche inférieure
}
```

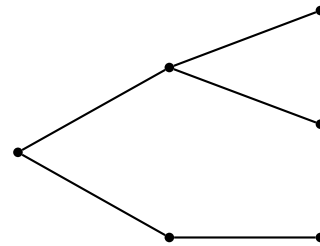




```

\psset{treemode=R,nodesep=0mm,%
        levelsep=20mm,treesep=15mm}
\pstree{\Tdot} % racine
{
  \pstree{\Tdot}{\Tdot \Tdot}
    % sous-arbre du haut
  \pstree{\Tdot}{\Tdot}
    % sous-arbre du bas
}

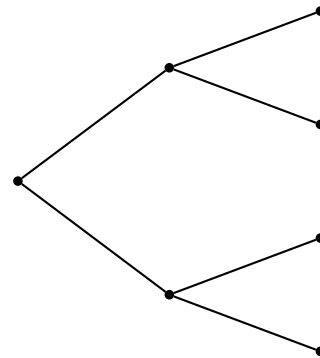
```



```

\psset{treemode=R,nodesep=0mm,%
        levelsep=20mm,treesep=15mm}
\pstree{\Tdot}
{
  \pstree{\Tdot}
    {\Tdot \Tdot}
  \pstree{\Tdot}
    {\Tdot \Tdot}
}

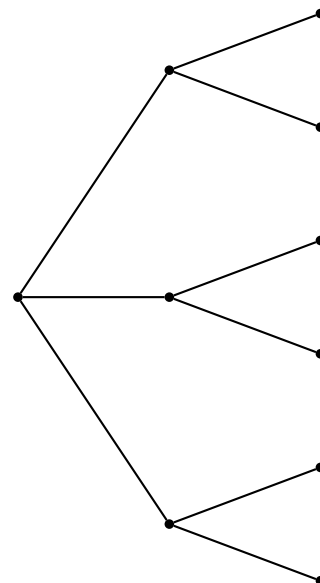
```



```

\psset{treemode=R,nodesep=0mm,%
        levelsep=20mm,treesep=15mm}
\pstree{\Tdot}
{
  \pstree{\Tdot}
    {\Tdot \Tdot}
  \pstree{\Tdot}
    {\Tdot \Tdot}
  \pstree{\Tdot}
    {\Tdot \Tdot}
}

```



On s'approche de l'objectif de cette chronique : on a construit un arbre à trois branches qui se séparent chacune en deux branches.

La différence entre ce dernier arbre et celui de la page 23 est la longueur des branches qui a été allongée dans le premier arbre pour permettre l'écriture des légendes ; cela a été obtenu par `levelsep=40mm` au lieu de `levelsep=20mm`.

Il ne reste plus qu'à nommer les nœuds et mettre des poids sur les branches, comme dans tout arbre pondéré qui se respecte !

## 5.4 Les nœuds

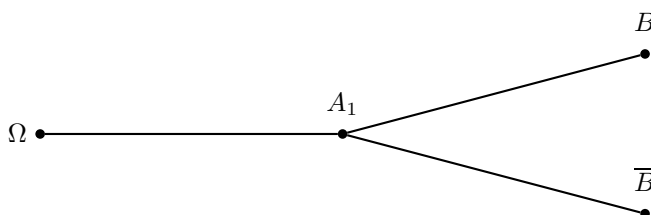
Pour écrire le nom d'un nœud, c'est très simple; il faut l'attacher au moyen d'un tilde ( $\sim$ ) au point que l'on trace; le tilde s'obtient par `Alt Gr 2`.

On définit la position du nom au moyen de la variable `\tnpos` à laquelle on peut donner la valeur `a` pour `above`, `b` pour `below`, `l` pour `left` ou `r` (par défaut) pour `right`; ainsi pour la racine on écrira `\Tdot~[tnpos=l]{\$\Omega\$}`, pour  $A_1$  on écrira `\Tdot~[tnpos=a]{\$A_1\$}`, etc.

La branche supérieure de l'arbre de la page 23 sera donc construite ainsi :

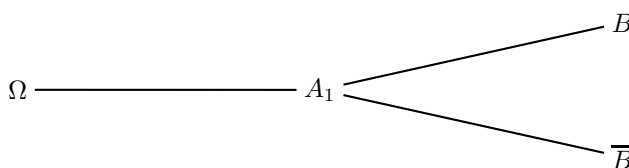
```
\psset{treemode=R, nodesep=0mm, levelsep=40mm, treesep=15mm}
\pstree{\Tdot~[tnpos=l]{\$\Omega\$}}
{
  \pstree
    {\Tdot~[tnpos=a]{\$A_1\$}}
    {
      \Tdot~[tnpos=a]{\$B\$}
      \Tdot~[tnpos=a]{\$\overline{B}\$}
    }
}
```

et donnera :



## 5.5 Autre version

Je sens bien que certains d'entre vous préféreraient l'arbre précédent représenté ainsi :



Il y a le nom du nœud à la place du point; on va donc remplacer `\Tdot~` par `\Tr` et laisser un peu de place autour du nom en modifiant la variable `nodesep` dans `\psset` :

```
\psset{treemode=R, nodesep=1mm, levelsep=40mm, treesep=15mm}
\pstree{\Tr{\$\Omega\$}}
{
  \pstree
    {\Tr{\$A_1\$}}
    {
      \Tr{\$B\$}
      \Tr{\$\overline{B}\$}
    }
}
```

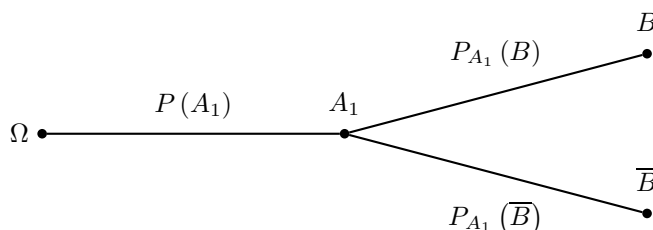
## 5.6 Les branches

Pour mettre des légendes sur les branches de l'arbre, on utilise les instructions `\taput` (avec **a** pour **a**bove), `\tbput` (avec **b** pour **b**elow) ou encore `\tlput` et `\trput` (respectivement **l**eft et **r**ight) quand l'arbre est dirigé vers le haut ou vers le bas.

Voici la version de l'arbre précédent avec les probabilités sur les branches :

```
\psset{treemode=R, nodesep=0mm, levelsep=40mm, treesep=15mm}
\pstree{\Tdot~[tnpos=1]{\Omega}}
{
  \pstree
  {
    \Tdot~[tnpos=a]{A_1}\taput{P(A_1)}
    {
      \Tdot~[tnpos=a]{B}\taput{P_{A_1}(B)}
      \Tdot~[tnpos=a]{\overline{B}}\tbput{P_{A_1}(\overline{B})}
    }
  }
}
```

Ce qui donne comme résultat :

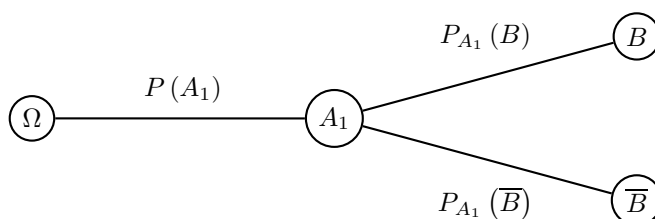


## 5.7 Nouvelle version

Peut-être voulez-vous une autre forme d'arbre ?

En voici une avec les noms des nœuds entourés par des cercles qui s'adaptent à la taille des noms ; il faut pour cela utiliser `\Tcircle` :

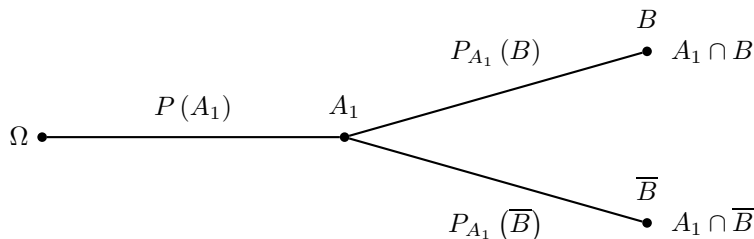
```
\psset{treemode=R, nodesep=0mm, levelsep=40mm, treesep=15mm}
\pstree{\Tcircle{\Omega}}
{
  \pstree
  {
    \Tcircle{A_1}\taput{P(A_1)}
    {
      \Tcircle{B}\taput{P_{A_1}(B)}
      \Tcircle{\overline{B}}\tbput{P_{A_1}(\overline{B})}
    }
  }
}
```



Pour avoir des cercles de même taille, il faut utiliser `\TCircle` (avec un **C** majuscule) à la place de `\Tcircle` et définir le rayon des cercles par une option du style `radius=20pt`.

## 5.8 La touche finale

Et comment écrire  $A_1 \cap B$  et  $A_1 \cap \overline{B}$  au bout de l'arbre ?



Il suffit de rajouter les séquences  $\{\$A_1 \ \backslash \cap B\}$  et  $\{\$A_1 \ \backslash \cap \overline{B}\}$  précédées d'un tilde au bon endroit :

```
\psset{treemode=R, nodesep=0mm, levelsep=40mm, treesep=15mm}
\pstree{\Tdot~[tnpos=1]{\$ \Omega \$}}
{
  \pstree
  { \Tdot~[tnpos=a]{\$A_1\$} \taput{\$P \ (A_1 \ ) \$} }
  {
    \Tdot~[tnpos=a]{\$B\$} ~ {\$A_1 \ \cap B\$} \taput{\$P_{A_1} \ (B \ ) \$}
    \Tdot~[tnpos=a]{\$ \overline{B} \$} ~ {\$A_1 \ \cap \overline{B} \$}
      \tbput{\$P_{A_1} \ ( \overline{B} \ ) \$}
  }
}
```

Une remarque : la séquence  $\backslash \Tdot~[tnpos=a]{\$B\$} ~ {\$A_1 \ \cap B\$}$  ne donne pas exactement le même résultat que la séquence  $\backslash \Tdot~{\$A_1 \ \cap B\$} ~ [tnpos=a]{\$B\$}$ .

Ne me demandez pas pourquoi !

## 5.9 Le code

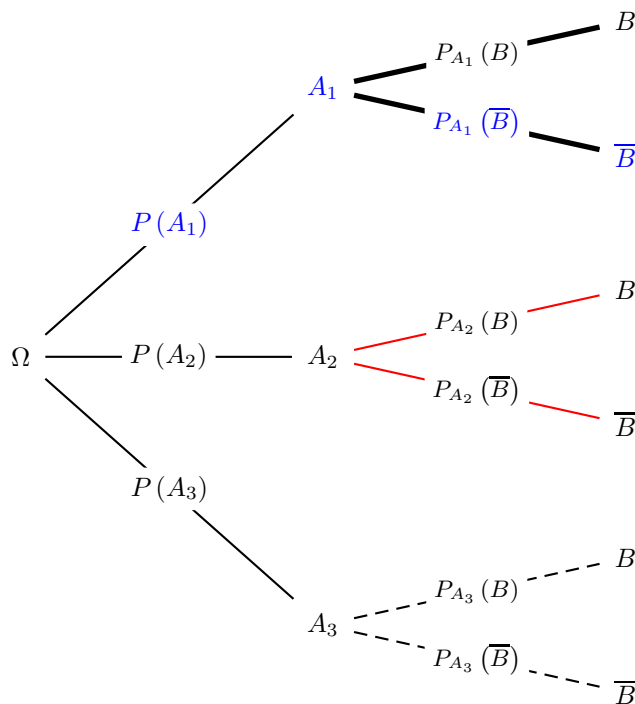
Voici le code complet de l'arbre de la page 23 :

```
\psset{treemode=R, nodesep=0mm, levelsep=40mm, treesep=10mm}
\pstree{\Tdot~[tnpos=1]{\$ \Omega \$}}
{
  \pstree
  { \Tdot~[tnpos=a]{\$A_1\$} \taput{\$P \ (A_1 \ ) \$} }
  {
    \Tdot~[tnpos=a]{\$B\$} ~ {\$A_1 \ \cap B\$} \taput{\$P_{A_1} \ (B \ ) \$}
    \Tdot~[tnpos=a]{\$ \overline{B} \$} ~ {\$A_1 \ \cap \overline{B} \$}
      \tbput{\$P_{A_1} \ ( \overline{B} \ ) \$}
  }
  \pstree
  { \Tdot~[tnpos=a]{\$A_2\$} \taput{\$P \ (A_2 \ ) \$} }
  {
    \Tdot~[tnpos=a]{\$B\$} ~ {\$A_2 \ \cap B\$} \taput{\$P_{A_2} \ (B \ ) \$}
    \Tdot~[tnpos=a]{\$ \overline{B} \$} ~ {\$A_2 \ \cap \overline{B} \$}
      \tbput{\$P_{A_2} \ ( \overline{B} \ ) \$}
  }
  \pstree
  { \Tdot~[tnpos=b]{\$A_3\$} \tbput{\$P \ (A_3 \ ) \$} }
  {
    \Tdot~[tnpos=a]{\$B\$} ~ {\$A_3 \ \cap B\$} \taput{\$P_{A_3} \ (B \ ) \$}
    \Tdot~[tnpos=a]{\$ \overline{B} \$} ~ {\$A_3 \ \cap \overline{B} \$}
      \tbput{\$P_{A_3} \ ( \overline{B} \ ) \$}
  }
}
```

## 5.10 Variante

Au lieu de placer les légendes au dessus ou en dessous des branches, on peut les écrire sur les branches en utilisant l'instruction `\ncput*` à la place de `\taput` et `\tbput`.

Voici un exemple (à ne pas suivre!) :



et le code pour l'obtenir :

```
\psset{treemode=R, nodesep=2mm, levelsep=40mm, treesep=15mm}
\pstree{\Tr{\Omega}}
{
  \pstree[linewidth=2pt]
  {\Tr{\blue $A_1$} \ncput*{\blue $P(A_1)$}}
  {\Tr{$B$} \ncput*{\small $P_{A_1}(B)$}
  \Tr{\blue $\overline{B}$}
  \ncput*{\blue \small $P_{A_1}(\overline{B})$}}
  \pstree[linecolor=red]
  {\Tr{$A_2$} \ncput*{$P(A_2)$}}
  {\Tr{$B$} \ncput*{\small $P_{A_2}(B)$}
  \Tr{$\overline{B}$} \ncput*{\small $P_{A_2}(\overline{B})$}}
  \pstree[linestyle=dashed]
  {\Tr{$A_3$} \ncput*{$P(A_3)$}}
  {\Tr{$B$} \ncput*{\small $P_{A_3}(B)$}
  \Tr{$\overline{B}$} \ncput*{\small $P_{A_3}(\overline{B})$}}
}
```

J'ai rajouté quelques options et un peu de couleur par-ci par-là pour vous donner des idées de modification.

## 5.11 Raccourcis

Il y a une variable appelée `shortput` qui permet d'utiliser des raccourcis pour les deux instructions `\taput` et `\tbput`.

L'instruction `\taput` place la légende au dessus de la branche, comme en exposant ; on pourra donc remplacer `\taput{\$P\{A_1\}}` par `^{\$P\{A_1\}}`.

De même comme l'instruction `\tbput` place la légende en dessous de la branche, comme en indice, on remplacera `\tbput{\$P_{A_1}\{\overline{B}\}}` par `_{\$P_{A_1}\{\overline{B}\}}`.

Il faut quand même activer la variable `shortput` et lui donner la valeur `tab`, ce que l'on fait dans `\psset` en entrant `shortput=tab`.

Voici le code qui donne exactement le diagramme du paragraphe 5.8, en utilisant les raccourcis :

```
\psset{treemode=D,nodesep=0mm,levelsep=40mm,treesep=15mm,shortput=tab}
\pstree{\Tdot~[tnpos=1]{\$ \Omega \$}}
{
  \pstree
  {\Tdot~[tnpos=a]{\$A_1\$}~^{\$P\{A_1\}}}
  {
    \Tdot~[tnpos=a]{\$B\$}~^{\$A_1 \cap B\$}~_{\$P_{A_1}\{B\}}
    \Tdot~[tnpos=a]{\$ \overline{B} \$}~^{\$A_1 \cap \overline{B} \$}
    _{\$P_{A_1}\{\overline{B}\}}
  }
}
```

Une interprétation (toute personnelle) de ce que peut vouloir dire `tab`.

Le `t` fait référence à `tree`.

Quant aux deux lettres de `ab`, elles veulent dire tout simplement `above` et `below` que l'on retrouve dans `\taput` et `\tbput`.

Il y a également une valeur de la variable `shortput` qui est `tabl` ; je ne suis pas loin de penser que dans ce cas le `l` signifie `left`, et le `r` signifie `right`!!!

Enfin dans une prochaine chronique sur les graphes, on verra que `shortput` peut prendre la valeur `nab`, avec `n` pour `node` (nœud), `a` pour `above` et `b` pour `below`.

## Chronique 6

# Petits trucs mathématiques

### 6.1 Gras ou très gras

Voici deux tableaux de signes ; voyez-vous la différence entre les deux ?

$x$	$-\infty$	$3$	$+\infty$
$3-x$	$+$	$0$	$-$

$x$	$-\infty$	$3$	$+\infty$
$3-x$	$+$	$0$	$-$

Bien sûr ! Dans le tableau de droite, les  $+\infty$  et  $-\infty$ , mais aussi le  $+$  et le  $-$  sont en gras. On a déjà vu dans la chronique 8 de la saison 1 comment écrire en gras des formules mathématiques en utilisant `\boldmath` ou `\boldsymbol` ; le package `amsmath` propose une autre fonction pour mettre en gras des formules mathématiques ; il s'agit de `\pmb` :

<code>+\$\infty\$</code>	<code>{\boldmath \$+\infty\$}</code> ou <code> \${\boldsymbol{+\infty}}\$</code>	<code> \${\pmb{+\infty}}\$</code>
$+\infty$	$+\infty$	$+\infty$

En fait `\pmb` est décrit dans la documentation de `amsmath` comme « poor man's bold » autrement dit le « gras du pauvre » parce que ce n'est pas une fonte particulière : ce n'est que la copie multiple du texte que l'on veut en gras avec de légers décalages, d'où l'impression de graisse. Personnellement, je n'utilise `\pmb` que dans l'écriture des signes  $+$  et  $-$  pour qu'ils apparaissent plus visibles à l'écran et à l'impression.

### 6.2 Environnement cases

Si on veut définir une fonction par morceaux, on peut procéder ainsi :

```
$f(x)=
\left\lbrace                % accolade gauche
\begin{array}{r l}          % début tableau
4x+1 & \text{si } x<1\\
5    & \text{si } x=1\\
-x+6 & \text{si } x>1
\end{array}                % fin tableau
\right.$                   % fin d'accolade
```

$$f(x) = \begin{cases} 4x+1 & \text{si } x < 1 \\ 5 & \text{si } x = 1 \\ -x+6 & \text{si } x > 1 \end{cases}$$

Il existe un environnement qui permet d'écrire ça de façon un peu plus rapide : c'est `cases` qui est intégré au package `amsmath`.

Inutile de définir l'accolade, et le tableau à deux colonnes est défini directement :

$$f(x) = \begin{cases} 4x+1 & \text{si } x < 1 \\ 5 & \text{si } x = 1 \\ -x+6 & \text{si } x > 1 \end{cases}$$

Je trouve le résultat un peu moins joli, mais il est obtenu plus rapidement !

Au passage, je rappelle les commandes `\left` et `\right` expliquées dans la chronique 8 de la saison 1 et qui remplacent `\left(` et `\right)` :

```
\renewcommand{\left}{\left(}
\renewcommand{\right}{\right)}
```

### 6.3 Virgule en mode mathématique

Peut-être avez-vous remarqué que lorsqu'on écrit un nombre à virgule en mode mathématique, il y a un petit espacement disgracieux juste après la virgule ?

Voyons ça : 12,345. Et de plus près : 12,345.

Ce serait quand même mieux écrit comme ça 12,345, non ?

Cet espacement provient du package `babel` et de son option `français` que l'on entre dans le préambule par `\usepackage[français]{babel}`. Pour ne plus avoir cet espace après la virgule, il faut rentrer l'instruction `\DecimalMathComma` juste après l'appel de l'extension :

```
\usepackage[français]{babel}
\DecimalMathComma
```

Et comment j'ai fait pour écrire 12,345 avec l'espace ?

En entrant `\StandardMathComma $12,345$` car `\StandardMathComma` va contrecarrer l'action de `\DecimalMathComma`.

Quand `\DecimalMathComma` sera activé, il faudra se rappeler qu'il n'y a plus cet espace après la virgule en mode mathématique, notamment si on définit un intervalle :

$\begin{array}{c} \$\cd a,b \cg$ \\ \text{donne} \\ \\ [a,b] \end{array}$	<div style="border-left: 1px solid black; height: 100px; margin: 0 auto;"></div>	$\begin{array}{c} \$\cd a\,,\,,b \cg$ \\ \text{donne} \\ \\ [a,b] \end{array}$
---	--	--

Les commandes `\cg` et `\cd` ont été définies dans la chronique 8 de la saison 1 par :

```
\newcommand{\cg}{\rceil \hspace{-4.5pt} \rfloor}
\newcommand{\cd}{\lceil \hspace{-4.5pt} \lfloor}
```

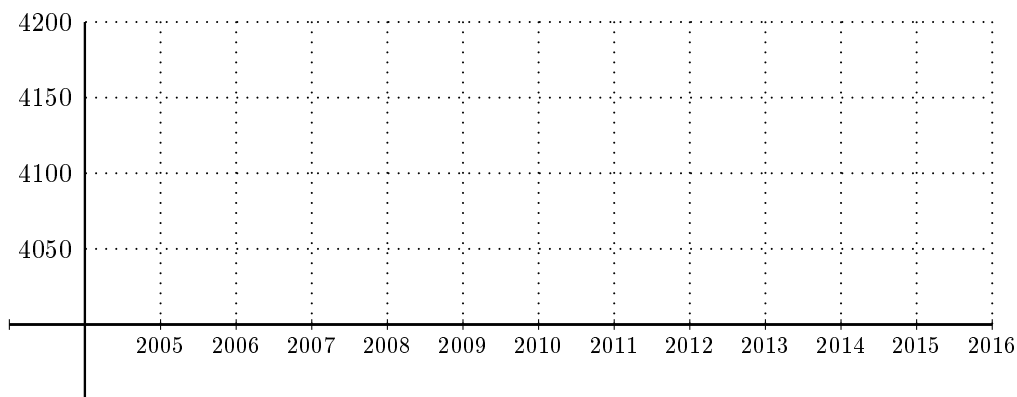
Elles permettent de définir des crochets plus visibles.

L'usage veut que l'on mette comme séparateur des bornes d'un intervalle une virgule, sauf si on travaille avec des nombres à virgule où dans ce cas le point-virgule s'impose.



## 6.4 Repère

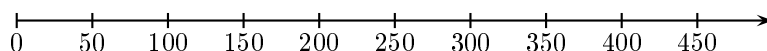
On a parfois besoin de tracer des repères dans lesquels les légendes n'ont pas grand-chose à voir avec les unités, notamment si on veut représenter des ajustements :



On peut naturellement tout traiter à la main et entrer autant de `\uput` qu'il y a de légendes à écrire; vous vous doutez que ce n'est pas ce que l'on va faire...

### 6.4.1 Premier exemple

Commençons par quelque chose de simple :



Il s'agit d'un axe horizontal régulièrement gradué tous les centimètres; la graduation va de 0 à 450 par pas de 50 et il y a 10 graduations.

On va d'abord régler les unités en abscisse : si on veut que la grandeur 50 soit représentée par 1 cm, il faut prendre pour unité 0,02 cm : c'est ce que fait l'affectation `xunit=0,02cm` dans `\psset`.

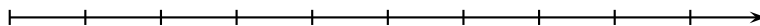
Ensuite on va définir une variable entière `\i` (*i* comme *integer*) qui va prendre 10 valeurs de 0 à 450 par pas de 50. Puis on va créer une boucle utilisant cette variable comme compteur au moyen de l'instruction `\multido` : `\multido{\i=0+50}{10}{...}`

La variable `\i` va prendre successivement les valeurs 0, 50, 100, 150, 200, 250, 300, 350, 400 et 450 (10 valeurs en tout, c'est le deuxième paramètre).

On mettra ce qu'il faut répéter à la place des pointillés comme troisième paramètre.

On va utiliser cette variable pour graduer l'axe en traçant un segment tous les centimètres :

```
\multido{\i=0+50}{10}{\psline(\i,-0.1)(\i,0.1)}
```

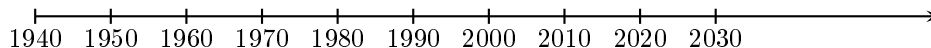


Il ne reste plus qu'à écrire les légendes en dessous des segments tracés, ce que l'on réalise avec des `\uput` bien placés :

```
\psset{xunit=0.02cm,yunit=1cm}           % unités
\begin{pspicture}(0,-0.6)(500,0.5)        % zone de tracé
\psline[arrowsize=2pt 3]{->}(0,0)(500,0) % tracé de l'axe
\multido{\i=0+50}{10}                     % définition de la variable \i
{                                           % nombre de valeurs prises par \i
  {                                         % début de ce qu'il faut répéter
    \psline(\i,-0.1)(\i,0.1)              % tracé des segments
    \uput[d](\i,0){\i}                    % placement des légendes
  }                                         % fin de la répétition
}
```

### 6.4.2 Deuxième exemple

Voici un cas où les graduations et les légendes sont dissociées :



Il s'agit d'un axe gradué tous les centimètres avec des légendes représentant des dates en partant de 1940 par pas de 10. On va donc définir une variable `\i` qui va servir de compteur et qui va permettre de tracer les traits verticaux, et une autre variable `\n` pour écrire les légendes.

La seule contrainte est qu'on doit avoir le même nombre de valeurs pour chaque variable, donc 10.

Voici le code :

```
\psset{xunit=0.1cm,yunit=1cm}           % unité en x de 0,1 cm
\begin{pspicture}(0,-0.4)(120,0.2)       % zone de tracé
\psline[arrowsize=2pt 3]{->}(0,0)(120,0) % tracé de l'axe
\multido{\i=0+10, \n=1940+10}           % définitions des variables
{10}                                     % nombre de valeurs prises par les variables
{   \psline(\i,-0.1)(\i,0.1)           % tracé des segments
    \uput[d](\i,0){\n}                 % placement des légendes
}
```

Deux remarques :

- l'instruction `\multido` a besoin de l'extension `multido` mais cette extension est contenue dans l'extension `pstricks-add` donc ce n'est pas utile de la charger si on a déjà chargé `pstricks-add`;
- on ne peut pas faire de calculs avec les variables du style `\i` et `\n`; à la place de `\n` on aurait aimé écrire `1940 + 10*\i` mais ce n'est pas possible.  
Il existe des extensions qui permettent de tels calculs.

### 6.4.3 Le code du repère

Il faut aussi penser à tracer « à la main » le quadrillage, car on ne peut pas employer `\psgrid` à cause des unités; on a donc rajouté une ligne dans chaque boucle `\multido`.

Voici le code pour tracer le repère du début du paragraphe :

```
\psset{xunit=1cm, yunit=0.02cm, runit=1cm}
\def\xmin {-1}   \def\xmax {12}
\def\ymin {-50}  \def\ymax {200}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\psaxes[ticks=-2pt 2pt,ticks=x,labels=none]%
(0,0)(\xmin,\ymin)(\xmax,\ymax)
\multido{\i=50+50, \n=4050+50}
{4}                                     % nombre de légendes sur (y'y)
{\uput[l](0,\i){\n}                   % légendes sur l'axe des ordonnées
 \psline[linestyle=dotted](0,\i)(\xmax,\i)}
    % quadrillage horizontal
\multido{\i=1+1, \n=2005+1}
{12}                                    % nombre de légendes sur (x'x)
{\uput[d](\i,0){\small \n}           % légendes sur l'axe des abscisses
 \psline[linestyle=dotted](\i,0)(\i,\ymax)}
    % quadrillage vertical
\end{pspicture}
```

Tout ce qu'il faut savoir sur les repères et les tracés de courbes en `PsTricks` se trouve dans la chronique 4 de la saison 1.

## Chronique 7

# Graphes

Tout comme pour les arbres,  $\text{\LaTeX}$  est très adapté pour la construction de graphes. L'indispensable package `pstricks-add` est nécessaire pour tracer les graphes de cette chronique.

### 7.1 Le principe

Un graphe est constitué de sommets, qui sont reliés entre eux par des arcs, orientés ou non. Sur ces arcs, on peut écrire des légendes qui, dans le cadre de graphes probabilistes, représentent des probabilités de changement d'état.

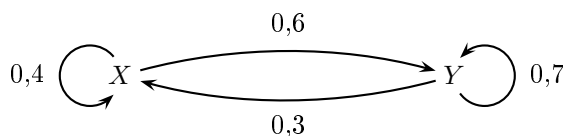
On va donc voir quelques façons de définir et de nommer des sommets que l'on appelle aussi parfois des nœuds (`nodes`) ; on verra également comment tracer des arcs orientés ou des segments reliant ces sommets, et enfin comment écrire des probabilités le long des arcs.

On pourrait tout faire en utilisant les outils que l'on a déjà vus pour tracer des figures de géométrie : `\dots`, `\psline` ou `\psarc` ; mais dans ce cas-là, il faudrait gérer soi-même les coordonnées des points, les rayons des arcs, etc.

Il y a beaucoup mieux avec des outils plus spécifiques que l'on pourra d'ailleurs utiliser dans d'autres circonstances.

### 7.2 Un premier graphe

On commence par un graphe probabiliste à deux états :



#### 7.2.1 Les sommets

En  $\text{\LaTeX}$ , il y a deux façons de définir des sommets : avec ou sans coordonnées. Pour ce premier graphe qui est dessiné horizontalement (il faut que ce soit comme ça !), on va utiliser des sommets définis sans coordonnées (inutiles ici).

L'instruction que j'ai utilisée est `\Rnode`. Il existe une instruction qui ressemble beaucoup à celle-ci qui est `\rnode` et qui semble donner les mêmes résultats. Attention, elle peut provoquer certains décalages horizontaux parfois ; je privilégie donc `\Rnode`, avec un `R` majuscule.

L'instruction `\Rnode` a besoin de deux paramètres (plus éventuellement d'autres optionnels que l'on entrera entre crochets); le premier paramètre est le nom que portera le sommet dans le dessin du graphe (il doit commencer par une lettre et ne comporter que des lettres non accentuées et des chiffres), le second paramètre est ce qui va être affiché dans le graphe.

Ainsi `\Rnode{X}{X}` sera un sommet du graphe qui sera affiché  $X$  (donc  $X$  en mode mathématique) et qui portera le nom  $X$ ; c'est-à-dire qu'on pourra tracer un arc partant de  $X$  ou arrivant à  $X$  en faisant référence à ce sommet par son nom  $X$ .

On va définir ainsi deux sommets, séparés par une distance de 4 cm; j'ai utilisé pour ça l'instruction `\hskip 4cm`. Attention, mettre 4 cm entre accolades provoque une erreur.

Pour définir les deux sommets  $X$  et  $Y$ , on écrira donc :

```
\Rnode{X}{X} \hskip 4cm \Rnode{Y}{Y}
```

### 7.2.2 Les arcs

L'instruction qui permet de tracer un arc est `\ncarc`; et comme on a nommé les sommets  $X$  et  $Y$ , pour tracer un arc entre  $X$  et  $Y$ , il suffit d'écrire : `\ncarc{X}{Y}`

Si on souhaite que l'arc soit orienté de  $X$  vers  $Y$ , on écrira : `\ncarc{->}{X}{Y}`

Mais si on se contente de :

```
\Rnode{X}{X} \hskip 4cm \Rnode{Y}{Y} % création des sommets
\nccarc{->}{X}{Y}                    % arc orienté de X vers Y
\nccarc{->}{Y}{X}                    % arc orienté de Y vers X
```

on obtient :  ce qui est très vilain!

On va donc :

- éloigner un peu les arcs du nom des sommets en entrant `nodesep=3pt`;
- augmenter un peu la courbure de l'arc en entrant `arcangle=15`. La variable `arcangle` désigne l'angle en degrés entre le départ de l'arc et la ligne droite entre les deux sommets;
- agrandir un peu les flèches en entrant `arrowsize=2pt 3`.

Comme ces options doivent s'appliquer à chaque arc tracé, on les définira au moyen d'un `\psset` :

```
\psset{nodesep=3pt,arcangle=15,arrowsize=2pt 3}
```

Pour une option qui ne serait à appliquer que pour un seul arc, on l'entrerait localement; ainsi si on veut dessiner l'arc qui va de  $X$  vers  $Y$  en rouge, on écrira : `\ncarc[linecolor=red]{->}{X}{Y}`.

### 7.2.3 Les probabilités

Il ne reste plus qu'à mettre des poids sur les arcs; là encore il y a plusieurs possibilités.

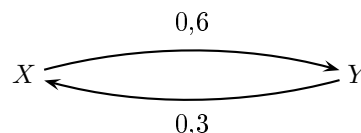
J'ai utilisé l'instruction `\Aput`.

La documentation de `pst-node` dit que le `A` signifie `Above` donc `au-dessus`; il existe aussi `\Bput` avec `B` pour `Below` donc `en dessous`. C'est à voir...

Ainsi :

```
\Rnode{X}{X} \hskip 4cm \Rnode{Y}{Y}
\psset{nodesep=3pt,arcangle=15,arrowsize=2pt 3}
\nccarc{->}{X}{Y} \Aput{0,6}
\nccarc{->}{Y}{X} \Aput{0,3}
```

donne :



On ne peut pas dire que le 0,3 de `\Aput{0,3}` soit situé au dessus de l'arc!

La distance entre la légende et l'arc peut être modifiée en entrant en option la distance souhaitée comme dans `\Aput[1pt]{0,3}`.

### 7.2.4 Les boucles

Il reste les boucles à tracer au moyen de `\nccircle` qui nécessite deux paramètres obligatoires et peut en avoir des optionnels.

L'instruction `\nccircle` dessine un cercle ou une partie de cercle qui passerait, s'il était entier, par le centre du sommet. Le premier paramètre est le nom du sommet, le second paramètre est le rayon du cercle. Enfin on peut orienter le cercle ou pas :

```
\Rnode{X}{$X$}
\nccircle{X}{0.5cm}
```



```
\Rnode{X}{$X$}
\nccircle{->}{X}{0.5cm}
```



```
\Rnode{X}{$X$}
\nccircle{<-}{X}{0.5cm}
```



Les paramètres optionnels permettent, entre autres, de faire démarrer le cercle vers la gauche, le bas ou la droite au lieu de le faire démarrer vers le haut ; il faut modifier la valeur de la variable `angleA` qui est à 0 par défaut :

```
\Rnode{X}{$X$}
\nccircle[angleA=90]
{->}{X}{0.5cm}
```



```
\Rnode{X}{$X$}
\nccircle[angleA=180]
{->}{X}{0.5cm}
```



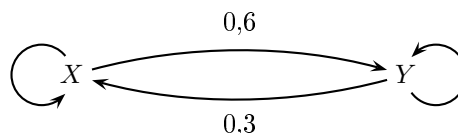
```
\Rnode{X}{$X$}
\nccircle[angleA=-90]
{->}{X}{0.5cm}
```



La variable `angleA` désigne l'angle que fait l'arc avec l'horizontale au départ du premier sommet. Il existe de même la variable `angleB` qui désigne l'angle que fait l'arc avec l'horizontale à l'arrivée sur le second sommet (quand l'arc va d'un sommet à un autre). La variable `angleA` fait invariablement référence au premier sommet, ou au sommet s'il n'y en a qu'un, tandis que `angleB` fait toujours référence au second sommet, quels que soient les noms des sommets.

On en reparlera plus loin.

À ce stade, on sait faire ce diagramme :



### 7.2.5 Le code

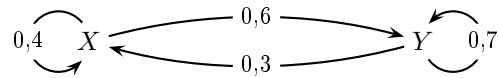
Il ne reste que les probabilités à écrire sur les boucles ; pour être honnête, j'ai essayé `\Aput` et `\Bput` et j'ai vu que c'était `\Bput` qu'il fallait utiliser !

Voici donc le code du diagramme de la page 35 :

```
\Rnode{X}{$X$} \hspace 4cm \Rnode{Y}{$Y$}           % définition des sommets
\psset{nodesep=3pt,arcangle=15,arrowsize=2pt 3}      % différents paramètres
\ncarc{->}{X}{Y}                                     \Aput{0,6} % arc pondéré de X vers Y
\ncarc{->}{Y}{X}                                     \Aput{0,3} % arc pondéré de Y vers X
\nccircle[angleA=90]{->}{X}{.4cm} \Bput{0,4}          % boucle autour de X
\nccircle[angleA=-90]{->}{Y}{.4cm} \Bput{0,7}          % boucle autour de Y
```

### 7.2.6 Variante

Légère variante du diagramme précédent :



Pour écrire les probabilités au milieu des arcs, il suffit de remplacer partout `\Aput` et `\Bput` par `\mput*` (essayez sans l'étoile!). J'ai également écrit les probabilités en un peu plus petit en entrant `\mput*{\small ...}` à la place de `\mput*{...}`.

### 7.2.7 Placement des diagrammes

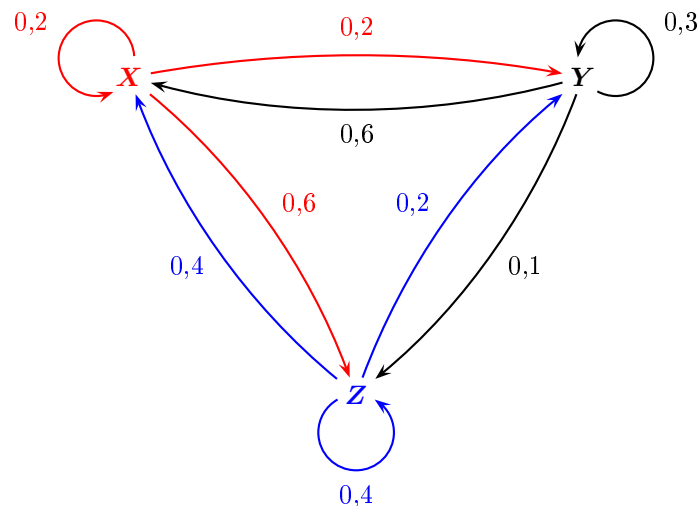
Pour placer exactement où l'on veut les graphes dans la page, je vous conseille d'insérer le code du graphe dans un environnement `pspicture` en ajustant `xmin`, `ymin`, `xmax` et `ymax` pour que le graphe complet soit contenu dans la zone ainsi définie.

Le code du graphe de la page 35 est en fait :

```
\begin{pspicture}(-2,-1)(6,1.5) % zone de tracé du graphe
\Rnode{X}{$X$} \hskip 4cm \Rnode{Y}{$Y$} % définition des sommets
\psset{nodesep=3pt,arcangle=15,arrowsize=2pt 3} % différents paramètres
\ncarc{->}{X}{Y} \Aput{0,6} % arc pondéré de X vers Y
\ncarc{->}{Y}{X} \Aput{0,3} % arc pondéré de Y vers X
\ncircle[angleA=90]{->}{X}{4mm} \Bput{0,4} % boucle autour de X
\ncircle[angleA=-90]{->}{Y}{4mm} \Bput{0,7} % boucle autour de Y
\end{pspicture}
```

## 7.3 Un graphe plus compliqué

Voici un graphe probabiliste à trois sommets :



La différence essentielle entre ce graphe et le précédent est que dans celui-ci les trois sommets ne sont pas alignés horizontalement : il faut donc définir les sommets au moyen de leurs coordonnées. C'est l'instruction `\psnode` que j'ai utilisée; cette instruction a besoin de trois paramètres : les coordonnées du sommet, son nom dans le dessin, le nom qui sera affiché dans le graphe.

On définira les trois sommets de ce graphe ainsi :

```
\psnode(0,0){X}{$X$}
\psnode(6,0){Y}{$Y$}
\psnode(3,-4.2){Z}{$Z$}
```

On peut rajouter de la couleur (pour les affichages des noms des sommets ou pour colorer les arcs), ou du gras (au moyen de `\boldmath`).

Le code complet de ce graphe probabiliste à trois états est :

```
\begin{pspicture}(-2,-6)(8,1)      % zone de tracé du graphe

    % définitions des sommets
\psnode(0,0){X}{\red \boldmath $X$}
\psnode(6,0){Y}{\boldmath $Y$}
\psnode(3,-4.2){Z}{\blue \boldmath $Z$}

\psset{nodesep=3pt,arcangle=15,arrowsize=2pt 3} % paramètres

    % arcs partant de X en rouge
\ncarc[linecolor=red,arcangle=10]{->}{X}{Y}      \Aput{\red 0,2}
\ncarc[linecolor=red]{->}{X}{Z}                  \Aput{\red 0,6}
\ncircle[angleA=60,linecolor=red]{->}{X}{.5cm}    \Bput{\red 0,2}

    % arcs partant de Y en noir
\ncarc{->}{Y}{X}                                \Aput{0,6}
\ncarc{->}{Y}{Z}                                \Aput{0,1}
\ncircle[angleA=-60]{->}{Y}{.5cm}                \Bput{0,3}

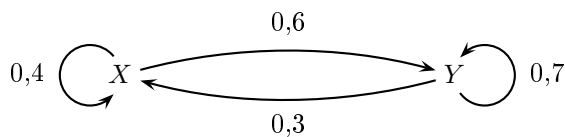
    % arcs partant de Z en bleu
\ncarc[linecolor=blue]{->}{Z}{X}                  \Aput{\blue 0,4}
\ncarc[linecolor=blue]{->}{Z}{Y}                  \Aput{\blue 0,2}
\ncircle[angleA=180,linecolor=blue]{->}{Z}{.5cm}  \Bput{\blue 0,4}

\end{pspicture}
```

## 7.4 Raccourcis

Tout comme pour les arbres (voir page 30), on peut définir les légendes sur les arcs en utilisant des raccourcis ; il faut pour cela activer la variable `shortput` en lui donnant la valeur `nab`, puis on utilise l'exponentiation `^` au lieu de `\Aput`, et la mise en indice `_` au lieu de `\Bput`.

Ainsi le diagramme :



s'obtient en entrant le code :

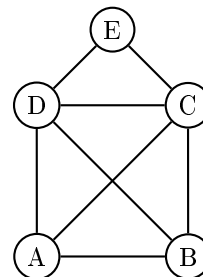
```
\begin{pspicture}(-2,-1)(6,1.5)
\Rnode{X}{$X$} \hskip 4cm \Rnode{Y}{$Y$}
\psset{nodesep=3pt,arcangle=15,arrowsize=2pt 3,shortput=nab}
\ncarc{->}{X}{Y}^{0,6}      % exposant
\ncarc{->}{Y}{X}^{0,3}
\ncircle[angleA=90]{->}{X}{4mm}_0,4 % indice
\ncircle[angleA=-90]{->}{Y}{4mm}_0,7
\end{pspicture}
```

## 7.5 Autre graphe

On peut également avoir besoin d'un graphe non orienté comme celui représenté ci-contre.

Les arcs ont été remplacés par des segments que l'on trace avec `\ncline`, et les sommets sont définis au moyen de l'instruction `\cnodeput` ; comme les sommets ne sont pas alignés horizontalement, il faut définir leurs coordonnées.

Avec `\cnodeput`, les noms des sommets sont dans un cercle dont la taille s'adapte au texte ; si on ne veut pas que les noms soient entourés de ce cercle, il suffit d'utiliser l'instruction `\cnodeput*`.



Le code de ce diagramme est :

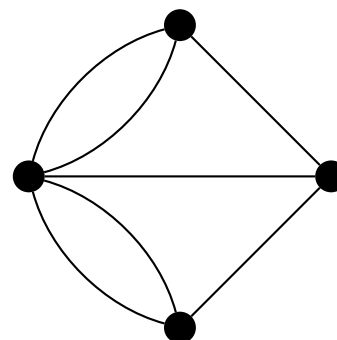
```
\psset{nodesep=0pt}
\begin{pspicture}(0,0)(2,3.2)
% définition des sommets
\cnodeput(0,0){A}{A}
\cnodeput(2,0){B}{B} \cnodeput(2,2){C}{C}
\cnodeput(0,2){D}{D} \cnodeput(1,3){E}{E}
% tracés des segments reliant les sommets
\ncline{A}{B} \ncline{A}{C} \ncline{A}{D}
\ncline{B}{C} \ncline{B}{D} \ncline{C}{D}
\ncline{D}{E} \ncline{C}{E}
\end{pspicture}
```

## 7.6 Les ponts de Königsberg

Un grand classique à l'origine de la théorie des graphes : le problème des ponts de Königsberg résolu par EULER.

Voici un diagramme représentant la situation :

```
\psset{nodesep=0pt,radius=6pt,arcangle=30}
\begin{pspicture}(0,-2)(4,2)
% définition des sommets
\cnode*(0,0){A}
\cnode*(4,0){B}
\cnode*(2,2){C}
\cnode*(2,-2){D}
% tracés des arcs reliant les sommets
\ncline{A}{B} \ncline{B}{C} \ncline{B}{D}
\ncarc{A}{C} \ncarc{C}{A}
\ncarc{A}{D} \ncarc{D}{A}
\end{pspicture}
```



Les sommets ont une nouvelle forme : ils ont été définis au moyen de l'instruction `\Cnode*` ; ainsi `\Cnode*(0,0){A}` définit le sommet appelé A qui est représenté par un disque centré en (0,0) et dont le rayon est donné par la variable `radius` définie dans `\psset`.

On obtient un cercle à la place d'un disque en entrant `\Cnode` à la place de `\Cnode*`.

Enfin si on veut des cercles (ou des disques) de rayons différents, on utilisera `\cnode` (ou `\cnode*`) ; il faudra alors rajouter comme deuxième paramètre le rayon du cercle : `\cnode(0,0){6pt}{A}`.



## 7.7 Tout faire avec `\ncurve`

Dans les graphes précédents, on a tracé les arcs en utilisant `\ncarc` et `\ncline`; ces instructions sont simples et faciles à utiliser.

Il en existe une autre, un peu plus complexe, mais plus universelle : `\ncurve`.

La documentation dit « `\ncurve` draws a bezier curve between the nodes », ce que l'on comprend facilement.





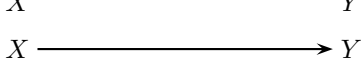
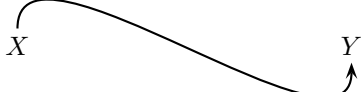
La syntaxe de cette instruction est :

`\ncurve[angleA=..., angleB=...]{sommet_A}{sommet_B}`

où `angleA` est l'angle en degrés que fait la courbe au départ du premier sommet, et `angleB` est l'angle que fait la courbe à l'arrivée au second sommet.

Par défaut, ces angles valent  $0^\circ$  et ils sont mesurés comme sur un cercle trigonométrique.

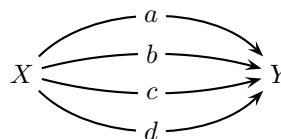
Voici un petit tableau avec différentes valeurs des angles pour voir ce que l'on peut faire :

Angle de départ <code>angleA</code>	Angle d'arrivée <code>angleB</code>	Résultat
$45^\circ$	$135^\circ$	
$-15^\circ$	$-165^\circ$	
$-60^\circ$	$180^\circ$	
$90^\circ$	$90^\circ$	
$0^\circ$	$180^\circ$	
$90^\circ$	$-90^\circ$	

La version `\ncurve*` remplit la zone sous la courbe; je n'en ai pas trop vu l'utilité. Essayez !

Il est donc très facile de faire un diagramme comme celui-ci en utilisant `\ncurve` :

```
\rnode{X}{$X$} \hskip 3cm \rnode{Y}{$Y$}
\psset{nodesep=3pt,arrowsize=2pt 3}
\ncurve[angleA=45,angleB=135]{->}{X}{Y} \mput*{$a$}
\ncurve[angleA=15,angleB=165]{->}{X}{Y} \mput*{$b$}
\ncurve[angleA=-15,angleB=-165]{->}{X}{Y} \mput*{$c$}
\ncurve[angleA=-45,angleB=-135]{->}{X}{Y} \mput*{$d$}
```



Vous en saurez autant que moi en allant consulter la documentation du package `pst-node` disponible à l'adresse <http://www.ctan.org/pkg/pst-node>

Il est inutile de charger explicitement ce package car il est intégré à `pstricks-add`.



## Chronique 8

# Applications des graphes

On peut utiliser les outils définis pour dessiner des graphes pour d'autres applications ; il suffit qu'il y ait des flèches à tracer pour que le procédé soit intéressant.

Pour profiter pleinement de cette chronique, il faut bien connaître les outils qui ont été explicités dans la chronique précédente consacrée aux graphes (voir page 35).

### 8.1 Distributivité

Voici une formule de distributivité bien connue :

$$(a + b) \times (c + d) = ac + \textcolor{red}{ad} + \textcolor{blue}{bc} + \textcolor{green}{bd}$$

Pour tracer une flèche partant de  $a$  et allant vers  $c$ , il suffit de définir un nœud en  $a$  (qu'on appelle A), un autre en  $c$  (qu'on appelle C), et de tracer un arc orienté entre ces deux nœuds.

J'ai utilisé `\Rnode` pour définir les nœuds et `\ncurve` pour tracer les arcs entre les nœuds ; voir la chronique précédente à la page 41.

Enfin j'ai agrémenté le tout en rajoutant des couleurs ; j'ai même défini une nouvelle couleur au moyen de `\newrgbcolor` (voir page 7) :

```
\newrgbcolor{vert}{0 0.5 0}
\psset{arrowsize=2pt 2,nodesep=1pt}

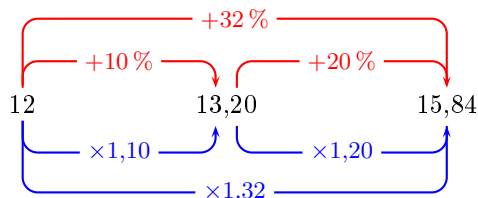
% on écrit (a+b)x(c+d) en définissant des noeuds au passage
$(\Rnode{A}a+\Rnode{B}b) \times (\Rnode{C}c+\Rnode{D}d)=
ac+\textcolor{red}{ad}+\textcolor{blue}{bc}+\textcolor{vert}{bd}$

% on trace les flèches qui partent vers le haut
\ncurve[angleA=45,angleB=135,linecolor=black]{->}{A}{C}
\ncurve[angleA=45,angleB=135,linecolor=red]{->}{A}{D}

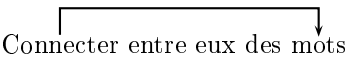
% et celles qui partent vers le bas
\ncurve[angleA=-45,angleB=-135,linecolor=blue]{->}{B}{C}
\ncurve[angleA=-45,angleB=-135,linecolor=vert]{->}{B}{D}
```

## 8.2 Pourcentages et coefficients multiplicateurs

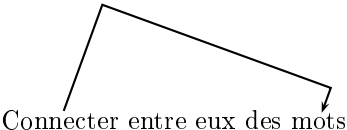
Autre utilisation des nœuds et des arcs, les pourcentages et les coefficients multiplicateurs ; on va voir comment construire un diagramme comme celui-ci :



Pour écrire au travers d'une flèche, on sait faire : il suffit d'utiliser `\mput*` (voir page 38). Mais les flèches de ce diagramme n'ont été tracées ni avec `\ncarc`, ni avec `\nccurve` ; j'ai utilisé `\ncbar` qui permet de connecter des mots entre eux comme dans :

`\Rnode{A}{Connecter} entre eux des \Rnode{B}{mots}`  
`\ncbar[angle=90]{->}{A}{B}`  Connecter entre eux des mots

Il suffit de donner une autre valeur que 90 à la variable `angle` pour vite comprendre à quoi elle correspond :

`\Rnode{A}{Connecter} entre eux des \Rnode{B}{mots}`  
`\ncbar[angle=70]{->}{A}{B}`  Connecter entre eux des mots

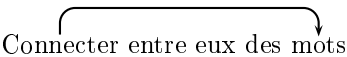
Affeux !

La variable `angle` est la valeur que le segment de départ fait avec l'horizontale ; c'est aussi (malheureusement) la valeur que le segment d'arrivée fait avec l'horizontale.

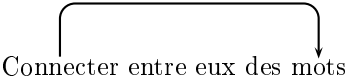
On ne peut pas définir pour `\ncbar` les deux valeurs `angleA` (départ) et `angleB` (arrivée).

Je m'en tiendrai donc à 90 comme valeur, ou -90 pour que la flèche parte vers le bas.

On arrondit les angles au moyen de la variable `linear` :

`\Rnode{A}{Connecter} entre eux des \Rnode{B}{mots}`  
`\ncbar[angle=90,linear=0.2]{->}{A}{B}`  Connecter entre eux des mots

Le segment de départ de la flèche a une longueur définie par la variable `arm` qui vaut 10 points par défaut ; on peut naturellement modifier cette longueur :

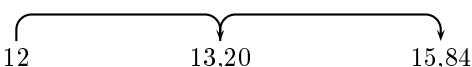
`\Rnode{A}{Connecter} entre eux des \Rnode{B}{mots}`  
`\ncbar[angle=90,linear=0.2,arm=20pt]{->}{A}{B}`  Connecter entre eux des mots

On a dans les mains tous les outils pour construire le diagramme que l'on cherche à tracer.

On va définir trois nœuds correspondant aux trois nombres 12, 13,20 et 15,84, appelés respectivement A, B et C, puis on va tracer la flèche allant de A vers B, et celle allant de B vers C :

```
\Rnode{A}{12} \hskip 2cm \Rnode{B}{13,20} \hskip 2cm \Rnode{C}{15,84}
\psset{angle=90,linear=0.2,nodesep=3pt} % paramètres
\ncbar{->}{A}{B} % flèche de A vers B
\ncbar{->}{B}{C} % flèche de B vers A
```

ce qui donne :



Les flèches se confondent au dessus de 13,20, ce qui n'est pas très beau.

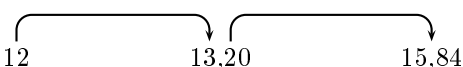
Ce sont les variables `offsetA` (relative au premier sommet) et `offsetB` (relative au second) qu'il faut modifier.

La variable `offsetA` a pour valeur 0 par défaut et il faut la modifier pour faire démarrer la flèche à un autre endroit que le milieu du nœud ; c'est ce que l'on veut pour la flèche allant de 13,20 vers 15,84. De même on va modifier la valeur de `offsetB` pour faire arriver la flèche avant le milieu du nœud ; c'est ce qu'il faut pour la flèche qui va de 12 vers 13,20.

En entrant :

```
\Rnode{A}{12} \hskip 2cm \Rnode{B}{13,20}\hskip 2cm \Rnode{C}{15,84}
\psset{angle=90,lineararc=0.2,nodesep=3pt}
\ncbar[offsetB=-4pt]{->}{A}{B}
\ncbar[offsetA=-4pt]{->}{B}{C}
```

on obtient :



Voici le code complet donnant le diagramme du début du paragraphe :

```
\begin{pspicture}(-4,-1.5)(6.5,1.5)
\Rnode{A}{12} \hskip 2cm \Rnode{B}{13,20}%
\hskip 2cm \Rnode{C}{15,84}
\psset{angle=90,lineararc=0.2,nodesep=3pt,linecolor=red}
\ncbar[offsetB=-4pt]{->}{A}{B} \mput*{\small \red $+10\$, \%$}
\ncbar[offsetA=-4pt]{->}{B}{C} \mput*{\small \red $+20\$, \%$}
\ncbar[arm=25pt,nodesep=4pt]{A}{C} \mput*{\small \red $+32\$, \%$}
\psset{angle=-90,linecolor=blue}
\ncbar[offsetB=4pt]{->}{A}{B} \mput*{\small \blue $\times 1,10$}
\ncbar[offsetA=4pt]{->}{B}{C} \mput*{\small \blue $\times 1,20$}
\ncbar[arm=25pt]{A}{C} \mput*{\small \blue $\times 1,32$}
\end{pspicture}
```

Remarquez que `offsetA` et `offsetB` ont des valeurs négatives si `angle` vaut 90, tandis qu'elles doivent être positives si `angle` vaut -90.

## 8.3 Fonctions composées

On peut également utiliser les outils des graphes pour représenter la composition de fonctions :

$$\begin{array}{ccc}
 h : & x & \longrightarrow & h(x) \\
 g : & & & X \longrightarrow g(X) \\
 f = g \circ h : & x & \longrightarrow & g(h(x))
 \end{array}$$

Pour créer ce diagramme, j'ai utilisé un tableau de type `array`, et j'ai défini des nœuds que j'ai reliés entre eux par des flèches.

Il faut quatre colonnes au tableau, et on va séparer la deuxième et la troisième, puis la troisième et la quatrième par un espace de 50 points ; pour cela il suffit d'écrire comme séparateur de colonne `@{\hspace{50pt}}`. Donc en entrant le signe `&` pour changer de colonne, c'est un espace de 50 points qui sera créé.

Sur la première ligne, on va définir  $x$  et  $h(x)$  comme des nœuds et tracer une flèche entre eux.

On fera de même pour la deuxième ligne avec  $X$  et  $g(X)$ , et sur la troisième avec  $x$  et  $g(h(x))$ .

Voici le code de ce diagramme :

```
\psset{nodesep=4pt,arrowsize=2pt 3} % paramètres
\[ % début du mode mathématique hors ligne
\begin{array}{r c @{\hspace{50pt}} c @{\hspace{50pt}} l}
% définition du tableau
h: & \Rnode{x}{x} & & \Rnode{hx}{h\,(x)} & & \\
& \ncline[|>]{x}{hx} & % flèche de la première ligne
g: & & \Rnode{X}{X} & & \Rnode{gX}{g\,(X)} & & \\
& \ncline[|>]{X}{gX} & % flèche de la deuxième ligne
f=g \circ h: & \Rnode{x}{x} & & & \Rnode{ghx}{g\,(h\,(x))} & & \\
& \ncline[|>]{x}{ghx} & % flèche de la troisième ligne
\end{array} % fin du tableau
\] % fin du mode mathématique
```

On voit que le signe de composition des fonctions est donné par `\circ` (en mode mathématique), et que pour tracer la flèche signifiant « a pour image », on entre comme option `{|->}` où le `|` est obtenu par `Alt Gr 6`.

## 8.4 Tableau de variations

Dans la chronique 10 de la saison 1, on a dessiné des tableaux de variations, en définissant des flèches `\flb` et `\flh` qui s'ajustaient plus ou moins bien...

Avec les nouveaux outils des graphes, il suffit de définir des nœuds et de les relier par des flèches, ce qui simplifie sacrément le travail !

Voici deux exemples :

$x$	$-\infty$	$-3$	$+\infty$
$f'(x)$	$-$	$0$	$+$
$f$	$0$	$-5$	$+\infty$

$x$	$-\infty$	$-3$	$+\infty$
$f(x)$	$0$	$-5$	$+\infty$

Normalement, vous devez savoir faire!

Voici quand même le code du tableau de gauche :

```

\psset{nodesep=3pt,arrowsize=2pt 3} % paramètres
$\begin{array}{|c|*5{c}|}
\hline
x & -\infty & & -3 & & +\infty \ \rule[-8pt]{0pt}{22pt} \\
\hline
f'(x) & & \pmb{-} & & \vline\hspace{-2.7pt}0 & \pmb{+} & & \rule[-6pt]{0pt}{18pt} \\
\hline
& \Rnode{zero}{0} & & & \Rnode{plusinf}{+\infty} \ \rule{0pt}{12pt} \\
f & & & & \\
& & & \Rnode{moins5}{-5} & & \\
\ncline{->}{zero}{moins5} \\
\ncline{->}{moins5}{plusinf} \\
\hline
\end{array} $

```

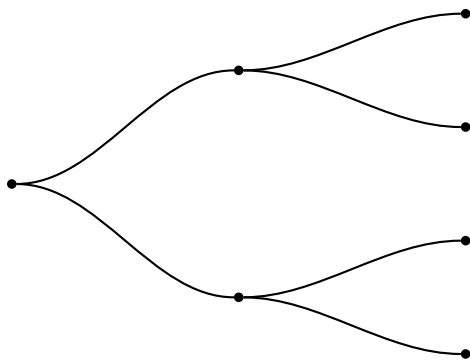
Les nœuds ont été appelés `zero`, `moins5` et `plusinf` ; pourquoi se compliquer la vie ?

## Chronique 9

# Compléments sur les arbres

### 9.1 Un peu de fantaisie

Voici un arbre binaire un peu fantaisiste :



C'est un des arbres de la page 25 dont les arêtes ont été modifiées.

Par défaut, une arête (`\edge`) est un segment, c'est-à-dire de type `\ncline`; on peut modifier globalement la forme des arêtes, ou une par une comme dans le paragraphe suivant.

Les arêtes de l'arbre de ce paragraphe ont été construites avec l'instruction `\nccurve` décrite en page 41; elles sont définies au moyen de l'instruction `\psedge` qu'il suffit de redéfinir ainsi :

```
\def\psedge{\nccurve[angleA=0,angleB=180]}
```

Autrement dit, au lieu d'avoir une arête rectiligne (`\ncline`), on a une arête courbe (`\nccurve`) dont on a défini l'angle de départ à  $0^\circ$  et l'angle d'arrivée à  $180^\circ$ .

À la place de `\def`, on aurait pu utiliser `\renewcommand`.

Et si l'on veut que cette redéfinition des formes des arêtes n'affecte pas les arbres qui vont suivre, on insérera le code de l'arbre entre deux accolades :

```
{
\def\psedge{\nccurve[angleB=180]}
\pstree[treemode=R,treesep=1.5cm,levelsep=3cm,nodesep=0pt]
{\Tdot}
{\pstree{\Tdot}{\Tdot \Tdot}}
{\pstree{\Tdot}{\Tdot \Tdot}}
}
```

Et si on veut des flèches comme arêtes, on peut redéfinir `\psedge` ainsi :

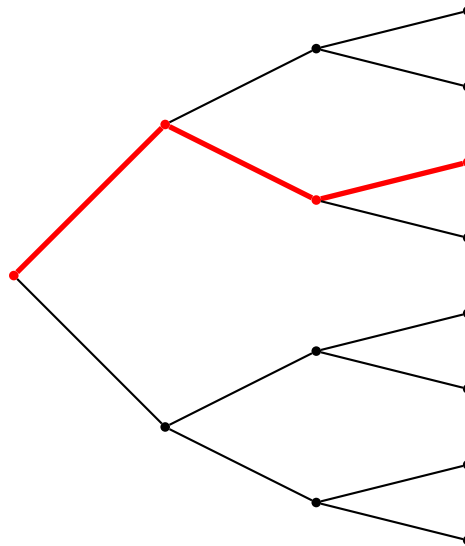
```
\def\psedge{\ncline[arrowsize=3pt 3,nodesep=3pt]{->}}
```

## 9.2 Chemin coloré

On a vu dans la première chronique consacrée aux arbres (voir page 29) comment colorer ou personnaliser un arbre ou un sous-arbre ; mais comment faire si on veut ne colorer qu'une branche d'un arbre, par exemple pour marquer un parcours ?

On ne peut pas utiliser la méthode du paragraphe précédent qui modifie toutes les arêtes de l'arbre.

Dans l'arbre ci-dessous j'ai coloré en rouge un parcours et les nœuds correspondants :



Les nœuds ont été définis par l'instruction `\Tdot`; pour les dessiner en rouge, il suffit de rajouter l'option `linecolor=red` au bon endroit.

Pour colorer une arête d'un arbre sans forcément colorer toutes les arêtes de cet arbre, il faut définir un nouveau type d'arête.

On va définir le type `\ARouge` qui sera un segment (`\ncline`) rouge d'épaisseur 2 points :

$$\text{\def\ARouge{\ncolor{red}\linewidth=2pt}}$$

L'instruction `\def` pourrait ici être remplacée par `\newcommand`.

Une fois qu'on a défini cette arête d'un nouveau type, il suffit, pour la faire afficher, d'entrer en option `edge=\ARouge`.

Il s'agit donc d'une utilisation locale de `edge`, contrairement à la redéfinition de `\psedge`.

L'arbre binaire de ce paragraphe a été dessiné en tapant :

```

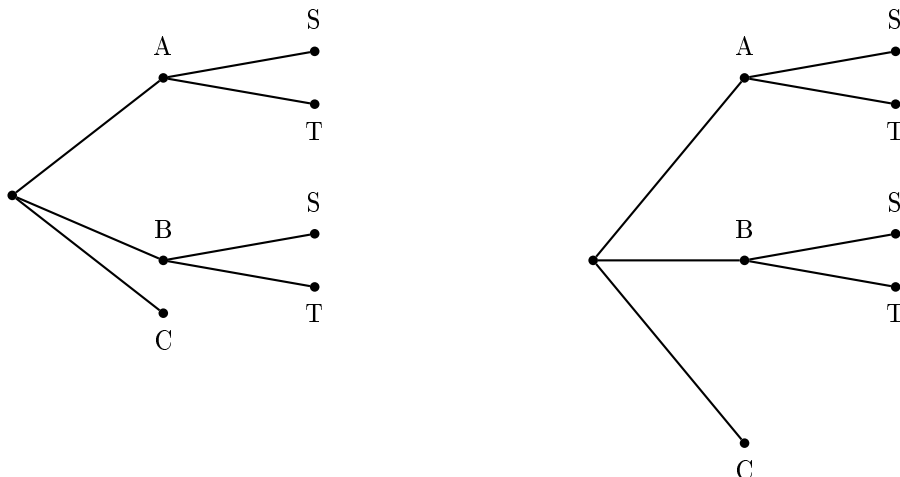
\def\ARouge{\ncline[linecolor=red,linewidth=2pt]}
\psset{treemode=R,nodesep=0mm,levelsep=20mm,treesep=10mm}
\pstree{\Tdot[linecolor=red]}
{
  \pstree{\Tdot[linecolor=red,edge=\ARouge]}
  {
    \pstree{\Tdot}{\Tdot \Tdot}
    \pstree{\Tdot[linecolor=red,edge=\ARouge]}
    {\Tdot[linecolor=red,edge=\ARouge] \Tdot}
  }
  \pstree{\Tdot}
  {
    \pstree{\Tdot}{\Tdot \Tdot}
    \pstree{\Tdot}{\Tdot \Tdot}
  }
}

```



### 9.3 Répartition des branches

Voici deux arbres représentant la même situation. Lequel est le plus joli ?



Celui de droite, bien sûr !

Dans le tracé d'un arbre, les branches sont dessinées « au mieux » en fonction des informations dont dispose  $\text{\LaTeX}$ .

L'arbre de gauche a donc été défini sans précaution particulière : trois branches partant de la racine, deux branches partant de la première branche, deux partant de la deuxième, et aucune partant de la troisième ; le résultat n'est pas très beau.

Pour tracer l'arbre de droite, j'ai tracé trois branches partant de la racine, puis deux branches partant de chacune de ces branches de première génération. Mais comme la troisième branche n'a pas de successeurs, j'ai utilisé l'instruction `\phantom` (voir chronique 8 de la saison 1) pour ne pas afficher le sous-arbre du bas.

Voici les codes utilisés ; pour l'arbre de gauche :

```
\psset{treemode=R,nodesep=0mm,levelsep=20mm,treesep=7mm}
\pstree{\Tdot}
{
  \pstree{\Tdot~[tnpos=a]{A}}
  {\Tdot~[tnpos=a]{S} \Tdot~[tnpos=b]{T}}
  \pstree{\Tdot~[tnpos=a]{B}}
  {\Tdot~[tnpos=a]{S} \Tdot~[tnpos=b]{T}}
  \Tdot~[tnpos=b]{C}
}
```

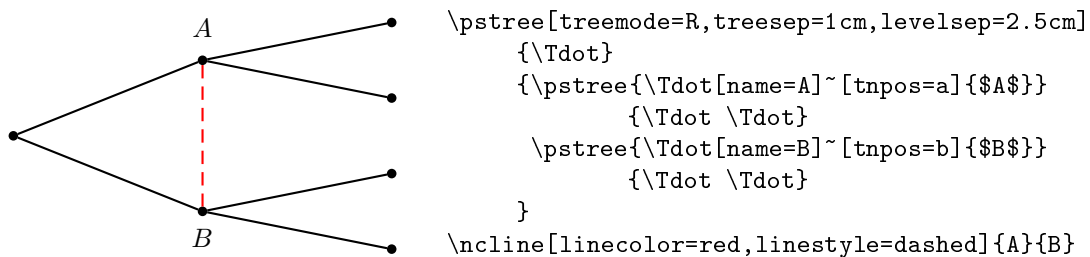
et pour l'arbre de droite :

```
\psset{treemode=R,nodesep=0mm,levelsep=20mm,treesep=7mm}
\pstree{\Tdot}
{
  \pstree{\Tdot~[tnpos=a]{A}}
  {\Tdot~[tnpos=a]{S} \Tdot~[tnpos=b]{T}}
  \pstree{\Tdot~[tnpos=a]{B}}
  {\Tdot~[tnpos=a]{S} \Tdot~[tnpos=b]{T}}
  \pstree{\Tdot~[tnpos=b]{C}}
  {\phantom{\Tdot~[tnpos=a]{S} \Tdot~[tnpos=b]{T}}}
}
```

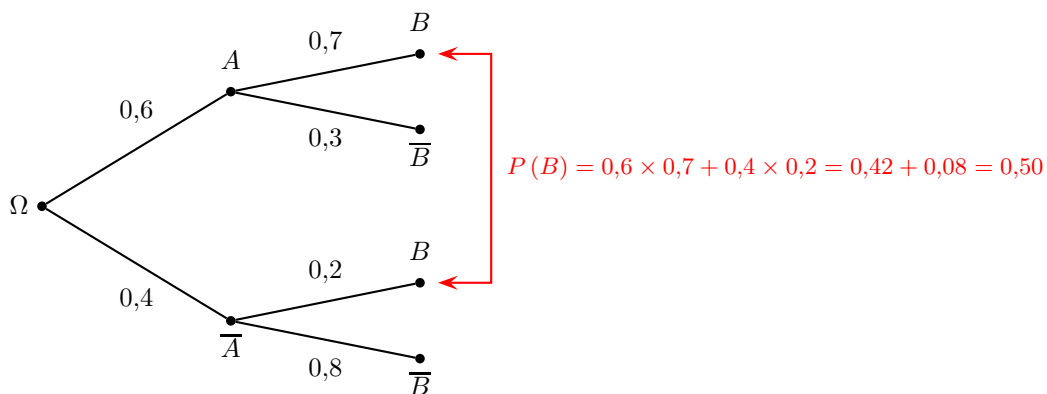
## 9.4 Nom des nœuds

On peut, dans un arbre, tracer des arcs entre des nœuds dont l'un n'est pas le successeur de l'autre ; il suffit pour cela de nommer les nœuds que l'on veut relier au moyen de l'option `name=...`, puis de tracer un segment ou un arc entre les deux nœuds qui ont été nommés.

Dans l'exemple ci-dessous on relie les nœuds au moyen d'un segment tracé avec l'instruction `\ncline` :



Voici un autre exemple, peut-être plus exploitable :



```

\pstree[treemode=R,treesep=1cm,levelsep=2.5cm]
{\Tdot~[tnpos=1]{$\Omega$}}
{
\pstree{\Tdot~[tnpos=a]{$A$}\taput{$0,6$}}
{\Tdot[name=b1]~[tnpos=a]{$B$}\taput{$0,7$}
\Tdot~[tnpos=b]{$\overline{B}$}\tbput{$0,3$}}
\pstree{\Tdot~[tnpos=b]{$\overline{A}$}\tbput{$0,4$}}
{\Tdot[name=b2]~[tnpos=a]{$B$}\taput{$0,2$}
\Tdot~[tnpos=b]{$\overline{B}$}\tbput{$0,8$}}
}
\psset{arrowsize=3pt 3}
\ncbar[arm=20pt,nodesep=5pt,linecolor=red]{<->}{b1}{b2}
\Aput{\red \small $P(B) = 0,6 \times 0,7 + 0,4 \times 0,2 = 0,42 + 0,08 = 0,50$}

```

Les explications de `\ncbar` et `arm` se trouvent à la page 44, et on parle la première fois de `nodesep` page 36.

À la place de : `\ncbar[arm=20pt,nodesep=5pt,linecolor=red]{<->}{b1}{b2}`  
on peut mettre : `\nccurve[nodesep=5pt,angleA=-30,angleB=30,linecolor=red]{<->}{b1}{b2}`

L'utilisation de `\nccurve` est expliquée en page 41.

## Chronique 10

# Multido

Dans la chronique 6 j'ai un peu parlé de l'instruction `\multido` qui nécessite l'extension `multido` ou l'extension `pstricks-add`.

Je vais un peu préciser les choses dans cette chronique-ci.

La documentation de l'extension se trouve dans le fichier `multido-doc.pdf` disponible à l'adresse :  
<http://www.ctan.org/tex-archive/macros/generic/multido>

### 10.1 Rappel

La syntaxe de l'instruction `\multido` est : `\multido{var=début+pas}{nombre}{à répéter}` comme dans : `\multido{\i=0+5}{7}{\i\ }` (voir plus bas).

On peut également répéter quelque chose sans utiliser de variable.

Par exemple si on veut laisser de la place pour écrire une réponse comme dans \_\_\_\_\_ il suffira d'entrer `\multido{}{10}{\_}` qui trace dix caractères `_` à la suite.

Mais il y a une chose importante à savoir quand on utilise l'instruction `\multido` avec des variables, c'est que ces variables sont typées : la première lettre de la variable détermine son type.

### 10.2 Variable de type integer

Si la variable a un nom qui commence par `i` ou par `I`, elle sera de type entier (`integer`).

Pour que le `\multido` soit valide, il faut que le nombre de départ et le pas de la variable utilisées soient des nombres entiers. Voyons quelques exemples :

	La séquence	donne
1	<code>\multido{\i=0+5}{7}{\i\ }</code>	0 5 10 15 20 25 30
2	<code>\multido{\i=0+5}{5}{(\i,\i)\ }</code>	(0,0) (5,5) (10,10) (15,15) (20,20)
3	<code>\multido{\i=0+5,\I=1+3}{5}{(\i,\I)\ }</code>	(0,1) (5,4) (10,7) (15,10) (20,13)
4	<code>\multido{\i=0+-5}{7}{\i\ }</code>	0 -5 -10 -15 -20 -25 -30
5	<code>\$\multido{\i=0+-5}{7}{\i\ }\$</code>	0 - 5 - 10 - 15 - 20 - 25 - 30
6	<code>\multido{\i=-5+1}{7}{\i\ }</code>	-5 -4 -3 -2 -1 0 1
7	<code>\multido{\i=0+0.5}{7}{\red \i\ }</code>	.5 0 0 0 0 0 0
8	<code>\multido{\i=0+5}{5}{\red 2*\i\ }</code>	2*0 2*5 2*10 2*15 2*20

J'ai appelé `\i` la variable utilisée dans chaque exemple ; on aurait pu l'appeler `\I` ou `\ia` ou `\Ia` ou ... (je n'ai pas essayé toutes les combinaisons de lettres possibles). Mais on ne peut pas l'appeler `\i1` car cela donne un résultat surprenant (mais compréhensible).

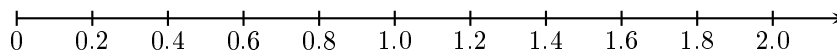
Cette variable `\i` est une variable locale qui n'a d'existence que dans la boucle.

On voit dans les exemples que :

- ce que j'ai appelé **nombre** dans la syntaxe de l'instruction `\multido` donne le nombre de résultats obtenus après l'exécution de `\multido` (1) ;
- on peut faire afficher des couples en utilisant une seule variable (2) ou deux (3). On peut de même utiliser des couples pour placer des points ou tracer des segments (on le verra dans une future chronique) ;
- un pas ne se soustrait pas mais peut être négatif ; la séquence `\i=0-5` produit une erreur et doit être remplacée par `\i=0+-5` (4) ;
- le nombre de départ peut être négatif (6) ;
- l'instruction `\multido` peut être utilisée en mode mathématique (5), mais il vaut mieux ne mettre en mode mathématique que ce que l'on affiche (6) ;
- si le pas est non entier, il n'y a pas de message d'erreur mais le résultat est faux (7). Il y a même quelque chose de bizarre au niveau de la couleur ;
- enfin on ne peut pas faire des calculs avec la variable utilisée (8).

## 10.3 Variable de type number

Comment faire si l'on veut graduer un axe tous les centimètres de 0,2 en 0,2 ?



Il ne faut pas utiliser une variable de type `integer` puisqu'on a vu qu'on ne pouvait pas y ajouter un nombre non entier ; on va utiliser une variable de type `number` dont la valeur de début peut être un nombre non entier, tout comme le pas.

Pour tracer le graphique du dessus, il faut deux variables : `\i` (de type `integer`) qui va permettre de tracer les graduations tous les centimètres (avec `\psline`), et une variable de type `number` pour écrire 0, 0.2, ..., 2.0 ; cette variable partira de 0 et aura pour incrément 0.2.

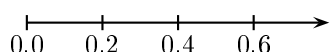
Une variable de type `number` doit avoir un nom commençant par `n` ou `N` comme `\n`, `\N`, `\nx`, `\Ny`, etc. et ne pas contenir de chiffres dans son nom.

Les deux variables `\i` et `\n` doivent prendre le même nombre de valeurs.

Le code du graphique est :

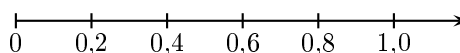
```
\psset{unit=1cm,arrowsize=2pt 3}% paramètres
\begin{pspicture}(-1,-0.5)(11,0.2)
\psline{->}(0,0)(11,0)% tracé de l'axe
\multido{\i=0+1,\n=0+0.2}% définition des variables
    {11}% nombre de valeurs
    {
        \psline(\i,-0.1)(\i,0.1)% tracé de la graduation
        \uput[d](\i,0){\n}% légendes
    }
\end{pspicture}
```

On obtient :



en définissant `\n` ainsi : `\n=0.0+0.2`.

On verra plus tard comment obtenir :



avec des nombres décimaux écrits avec des virgules et pas des points.

Quelques exemples de boucles traitées avec une variable de type `number` :

	La séquence	donne
1	<code>\multido{\n=10+5}{5}{\n\ }</code>	10 15 20 25 30
2	<code>\multido{\n=10+0.05}{5}{\n\ }</code>	10 10.05 10.10 10.15 10.20
3	<code>\multido{\n=10.1+1.5}{5}{\n\ }</code>	10.1 11.6 13.1 14.6 16.1
4	<code>\multido{\n=10.1+0.05}{5}{\red \n\ }</code>	10.1 10.06 10.11 10.16 10.21
5	<code>\multido{\n=10.01+0.5}{5}{\red \n\ }</code>	10.01 10.6 11.1 11.6 12.1
6	<code>\multido{\n=10.01+0.50}{5}{\n\ }</code>	10.01 10.51 11.01 11.51 12.01
7	<code>\multido{\n=10+5}{5}{\red 2+\n\ }</code>	2+10 2+15 2+20 2+25 2+30

Quelques remarques que l'on peut tirer de ces exemples :

- pas de problème si on emploie `\n` comme un entier (1) ;
- pas de problème non plus si la valeur de départ est entière, et l'incrément non entier (2) ;
- toujours pas de problème si la valeur de départ contient une décimale, tout comme l'incrément (3) ;
- si le nombre de départ a une décimale et l'incrément deux (4), ou le contraire (5), il n'y a pas de message d'erreur mais les résultats sont faux ;
- ça redevient juste si le nombre de départ et l'incrément ont chacun deux décimales (6) ;
- les calculs avec la variable ne sont toujours pas possibles (7).

Moralité ?

Si on veut obtenir un résultat conforme aux espérances avec une variable de type `number`, on s'arrangera pour que le nombre de départ et l'incrément aient le même nombre de chiffres après la virgule ; c'est toujours possible, il suffit de rajouter quelques 0 là où il faut.

Si le nombre de départ est entier, ça fonctionnera toujours.

On peut donc faire un peu ce que l'on veut avec le type `number`.

## 10.4 Variable de type real

Il existe un autre type de variable, le type `real` dont le nom doit commencer par `r` ou `R`. Le `\multido` donnera un résultat correct s'il y a moins de 4 chiffres de chaque côté du point décimal. Hélas, le mode d'emploi lui-même signale qu'il peut y avoir « occasional small errors » ; en effet, si on tape : `\multido{\r=0.10+0.05}{6}{\r\ }`

on obtient : 0.1 0.15001 0.20001 0.25002 0.30002 0.35002

C'est plutôt moins bien qu'avec le type `number` ; à oublier, donc.

## 10.5 Variable de type dimension

Je signale pour mémoire le type `dimension` dont je n'ai pas bien vu l'intérêt.

L'initiale du nom de la variable doit être `d` ou `D`, on s'en doutait !

On ajoute des centimètres et des points et ça donne des `sp` (scaled points) :

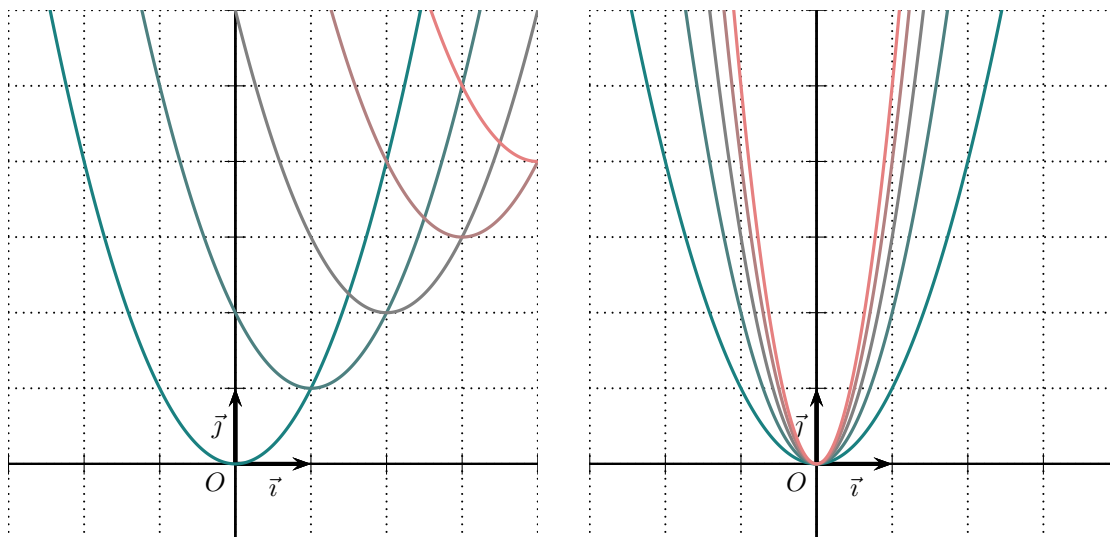
`\multido{\d=1cm+10pt}{4}{\d\ }` donne 1864679sp 2520039sp 3175399sp 3830759sp

Il faut 65536 `sp` pour faire 1 point.

Les curieux iront voir dans la documentation (dans laquelle il n'y a pas grand-chose!!!).

## 10.6 Applications graphiques

On peut utiliser `\multido` pour tracer des familles de fonctions.  
Voyons deux variations sur la fonction carré :



Voici le code du graphique de gauche :

```
\psset{unit=1cm,algebraic}
\def\xmin{-3} \def\xmax{4}
\def\ymin{-1} \def\ymax{6}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=1,griddots=10,gridlabels=0]
\psaxes[labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[d1](0,0){$0$}
\psaxes[linewidth=1.5pt,arrowsize=2pt 2]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$}
\uput[l](0,0.5){$\vec{\jmath}$}
\psset{linewidth=1.2pt,plotpoints=1000}
\multido{\i=0+1,\n=0.1+0.2}
{5}
{
\definecolor{couleur}{rgb}{\n,0.5,0.5}
\psplot[linecolor=couleur]{\xmin}{\xmax}{(x-\i)^2+\i}
}
\end{pspicture*}
```

On peut voir que l'on a utilisé la variable `\i` dans des calculs.

La variable `\n` sert à modifier la couleur des tracés ; voir les explications de `\definecolor` dans la chronique 1 de la saison 2 (page 4). L'instruction `\definecolor` est située à l'intérieur de la boucle pour que la couleur change à chaque tracé de courbe.

La fonction doit être écrite en notation infixée : il faut activer l'option `algebraic` dans `\psset` en écrivant `algebraic=true` ou `algebraic` tout court.

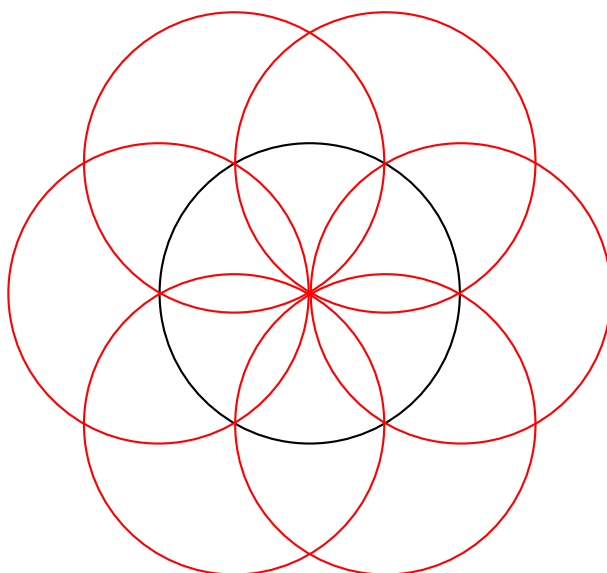
Le deuxième graphique s'obtient en remplaçant la ligne `\psplot...` par :

```
\psplot[linecolor=couleur]{\xmin}{\xmax}{\i*x^2}
```

et en remplaçant `\i=0+1` par `\i=1+1`.

## 10.7 Rosace

À ce stade, on sait presque réaliser ce dessin :



Il y a un cercle en noir de rayon 2, tracé avec `\pscircle` ; pas de problème.

Il y a six cercles en rouge de même rayon que le cercle noir, dont les centres sont situés aux sommets d'un hexagone régulier inscrit dans le premier cercle.

Pour désigner les centres de ces cercles, on va utiliser le repérage polaire.

En `pstricks`, quand dans un couple  $(x,y)$  les nombres sont séparés par une virgule, il s'agit de coordonnées cartésiennes.

Quand les nombres sont séparés par un point virgule, on passe en repérage polaire (avec un angle exprimé en degrés).

Les centres des cercles sont donc repérés en polaire par les couples  $(2;0)$ ,  $(2;60)$ ,  $(2;120)$ , ...,  $(2;300)$  ; on définira une variable entière représentant l'angle, qui partira de 0 et qui ira de 60 en 60 en prenant 6 valeurs.

Voici le code :

```
\psset{unit=1cm}
\def\xmin{-4.5} \def\xmax{4.5}
\def\ymin{-4.5} \def\ymax{4.5}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pscircle(0,0){2}%           cercle en noir
\psset{linecolor=red}%      on passe en rouge
\multido{\i=0+60}%          définition de la variable
        {6}%                nombre de valeurs prises
        {\pscircle(2;\i){2}}% cercles en rouge
\end{pspicture}
```

### Petite remarque

Ce n'est pas que pour faire joli et pour montrer ce que l'on peut faire avec `\multido` que j'ai dessiné cette rosace ; on peut aussi l'utiliser en classe.

En sixième, on peut faire reproduire ce dessin pour travailler l'apprentissage du compas.

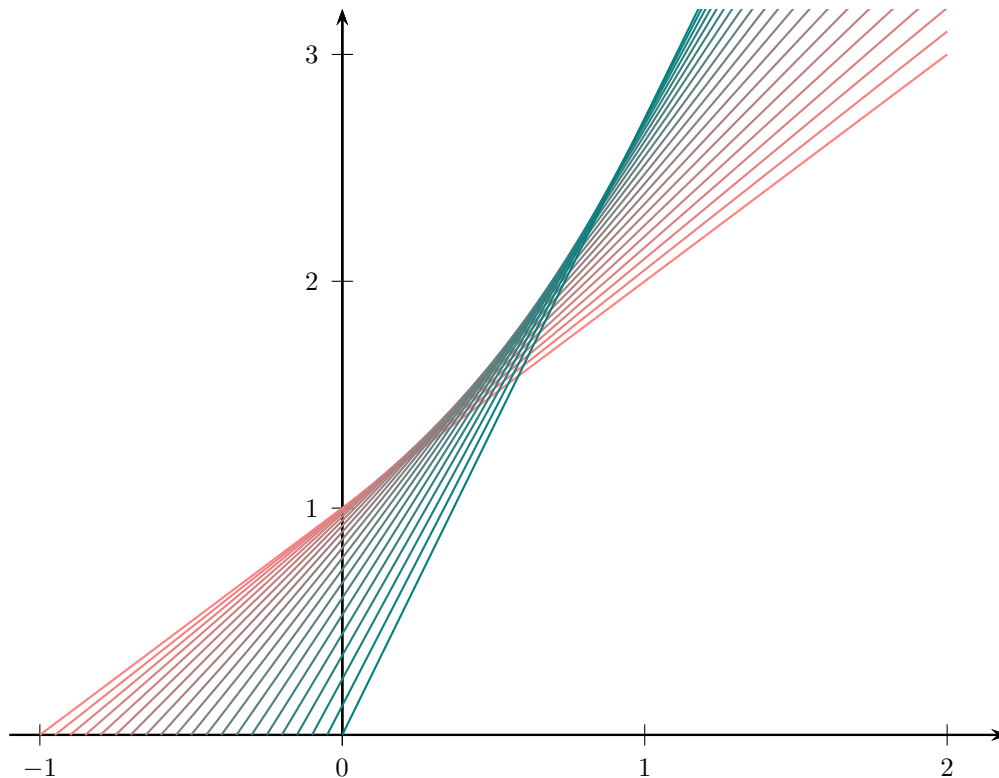
Plus tard dans la scolarité on peut faire décrire cette figure aux élèves, voire même leur faire écrire un texte mathématique qui permettrait à d'autres élèves de tracer cette rosace.

## 10.8 Bouquet final

Je ne peux pas terminer cette chronique sans vous présenter ce graphique créé par JEAN-ÉRIC VISCA et qui m'a inspiré pour tracer les graphiques du paragraphe 10.6.

J'ai un peu modifié le code (notamment en simplifiant l'équation de la droite) ; on trouve l'original à l'adresse :

<http://calque.pagesperso-orange.fr/latex/latexmultido.html>



Il s'agit de tracer les tangentes à la courbe de la fonction exponentielle pour 21 points de cette courbe dont les abscisses sont comprises entre 0 et 1.

Chaque tangente est construite avec une couleur différente qui est définie dans la boucle.

Voici le code de ce beau graphique (qui nécessite l'extension `pstricks-add`) :

```
\psset{xunit=4,yunit=3,algebraic}
\begin{pspicture}(-1,0)(1,3)
\psaxes{->}(0,0)(-1,0)(1.1,3.2)
\multido
  {\n=-1+0.05}
  {21}
  {
    \definecolor{couleur}{rgb}{-\n,0.5,0.5}
    \psplot[linecolor=couleur]{\n}{1}{2.718^{(\n + 1)*(x-\n)}}
  }
\end{pspicture}
```

Les matheux vérifieront les calculs !



## Chronique 11

# Fonction partie entière

Pourquoi consacrer une chronique à la fonction partie entière ?

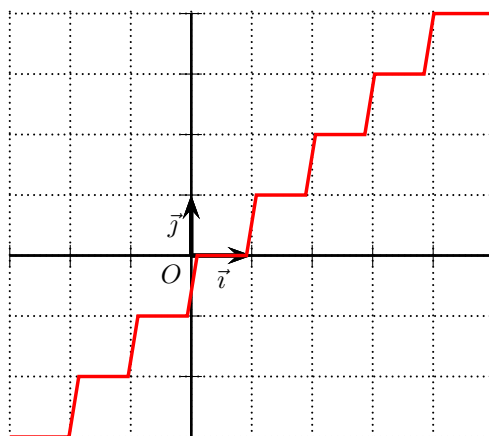
Parce que cette fonction permet de voir des petites choses nouvelles et de travailler un peu l'instruction `\multido` détaillée dans la chronique précédente.

### 11.1 Première version

La fonction partie entière s'appelle `floor` comme dans GeoGebra.

On peut donc essayer de tracer sa représentation graphique au moyen d'un `\psplot` :

```
\psset{unit=0.8cm}
\def\xmin{-3} \def\xmax{5}
\def\ymin{-3} \def\ymax{4}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=1,griddots=10,gridlabels=0]
\psaxes[labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[d1](0,0){$0$}
\psaxes[linewidth=1.5pt]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$}
\uput[l](0,0.5){$\vec{\jmath}$}
\psset{linecolor=red,linewidth=1.3pt}% ligne qui sera modifiée
\psplot{\xmin}{4.99}{x floor}% ligne qui sera modifiée
\end{pspicture}
```

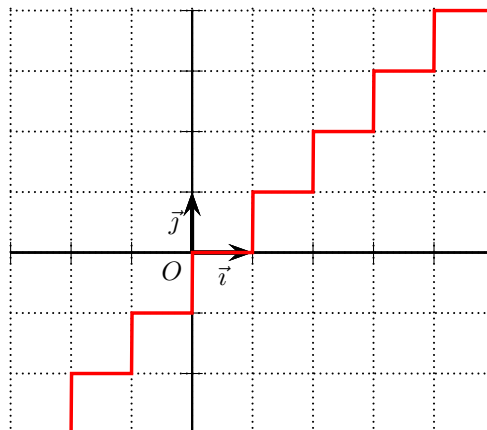


Ce n'est vraiment pas fameux !

Un début de solution consiste à augmenter le nombre de points par un `plotpoints=1000` comme option dans `\psplot`.

C'est mieux, mais ce n'est toujours pas acceptable : les points sont reliés entre eux, même quand il y a discontinuité, c'est-à-dire pour chaque valeur entière de la variable  $x$ .

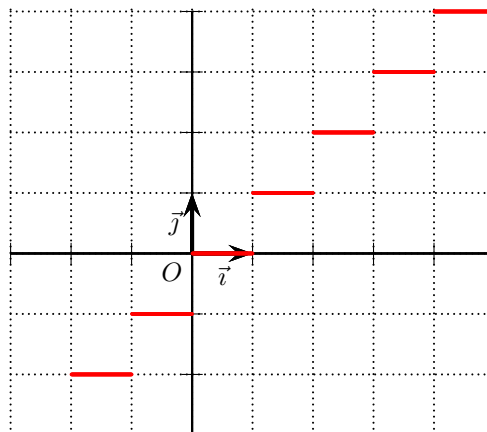
Un peu comme sur les calculatrices.



## 11.2 Deuxième version

Comme on peut le faire sur les calculatrices, on va tracer la fonction point par point sans relier les points entre eux ; pour cela on va entrer comme option `plotstyle=dots` dans `\psplot`.

Il a fallu quand même réduire la taille des points ; pour cela il a suffi de modifier la variable `dotsscale` dans `\psset`.



Le résultat est assez satisfaisant.

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,dotsscale=0.4}
\psplot[plotpoints=1000,plotstyle=dots]{\xmin}{4.99}{x floor}
```

Au passage, on remarque que l'on s'arrête à 4.99 (comme dans le graphique du paragraphe 11.1) et pas à `\xmax` pour éviter un point (ou un segment) disgracieux en dehors du quadrillage.

Et si on veut marquer les points de coordonnées  $(-3, -3)$ ,  $(-2, -2)$ , etc. ?

On va utiliser `\multido` qui va servir également à tracer les segments.

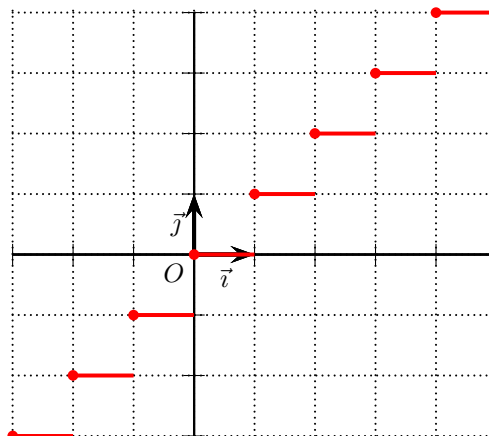
## 11.3 Troisième version

Que faut-il comme variables dans une boucle `\multido` pour placer les points et tracer les segments qui vont représenter la fonction partie entière ?

- Pour tracer les points, il faut une variable de type entier (appelée `\i`) qui prendra toutes les valeurs entières entre  $-3$  et  $4$  soit 8 valeurs. On utilisera `\psdots` pour placer ces points.
- On tracera ensuite 8 segments de  $(-3, -3)$  à  $(-2, -3)$ , de  $(-2, -2)$  à  $(-1, -2)$ , etc., et de  $(4,4)$  à  $(5,4)$  ; il faut donc une deuxième variable de type entier (appelée `\I`) qui prendra 8 valeurs entières entre  $-2$  et  $5$ . Les segments seront tracés avec `\psline`.

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,linewidth=1.5pt}
\multido{\i=\xmin+1,\I=-2+1}
{8}
{
\psdots[linewidth=1pt](\i,\i)
\psline(\i,\i)(\I,\i)
}
```



On trace les points de coordonnées  $(\i,\i)$  et les segments qui vont de  $(\i,\i)$  à  $(\I,\i)$ .  
Pas mal ! Mais on peut faire mieux en n'utilisant qu'une seule variable.

## 11.4 Quatrième version

En fait, dans le tracé de la fonction partie entière, les segments sont tous les mêmes ; par rapport au point, on avance d'une unité en abscisse et on garde la même ordonnée.

Ce serait bien si `\pstricks` autorisait les déplacements relatifs.

C'est évidemment possible ! Sinon je n'en parlerais pas...

On va utiliser l'instruction `\rlineto` qui va tracer un segment relativement au point courant, comme s'il s'agissait d'un vecteur de coordonnées  $(1,0)$  ; on place le point courant au bon endroit en utilisant l'instruction `\moveto`.

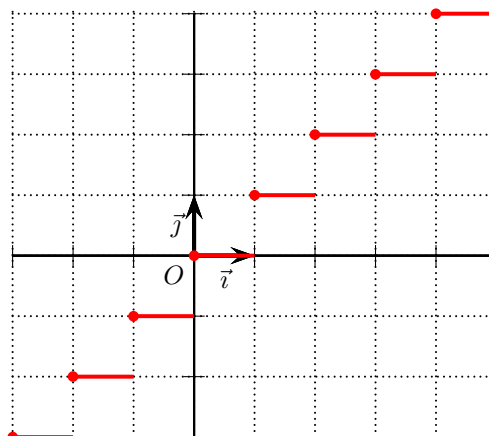
Et quand je vous aurai dit qu'il faut placer le tout dans un environnement personnalisé de type `\pscustom` (déjà vu en saison 1 lors du tracé d'aires sous une courbe), vous saurez tout !

Petit détail qui m'a agacé : l'instruction `\psdots` qui place les points doit être écrite en dehors de l'environnement `\pscustom` ; ne me demandez pas pourquoi !

Une seule variable de type `integer` suffit alors dans la boucle `\multido`.

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,linewidth=1.5pt}
\multido{\i=\xmin+1}
{8}
{
\psdots[linewidth=1pt](\i,\i)
\pscustom{
\moveto(\i,\i)
\rlineto(1,0)
}
}
```



Le résultat est évidemment le même que dans la troisième version.

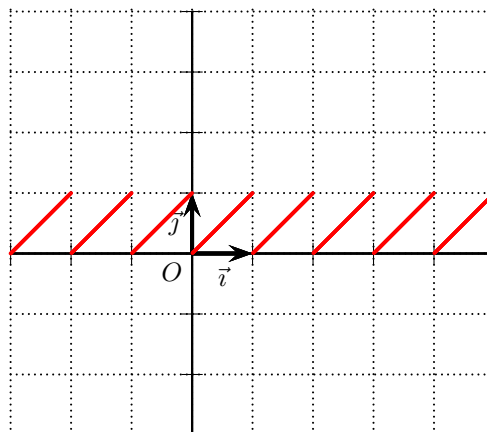
## 11.5 Application

Pour réemployer les méthodes que l'on vient de voir, on va tracer la représentation graphique de la fonction  $x \mapsto x - E(x)$  où  $E$  désigne la fonction partie entière; on va le faire de deux façons : point par point puis en utilisant `\rlineto`.

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,dotscale=0.4}
\psplot[plotpoints=1000,plotstyle=dots]
  {\xmin}{4.99}{x x floor sub}
```

En entrant l'option `algebraic=true` dans `\psset`, on peut écrire `{x-floor(x)}` à la place de `{x x floor sub}`.

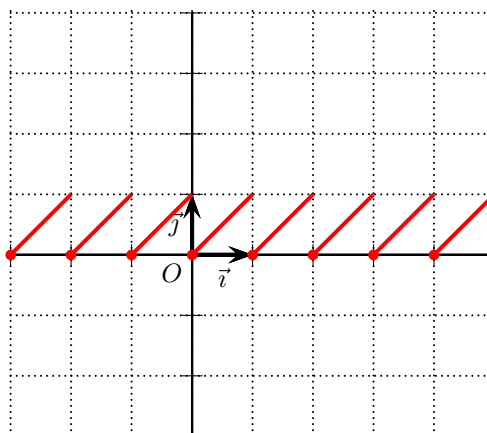


Pour aller du début d'un segment à son extrémité, on augmente l'abscisse de 1 et l'ordonnée de 1 : c'est donc facile d'utiliser l'instruction `\rlineto`.

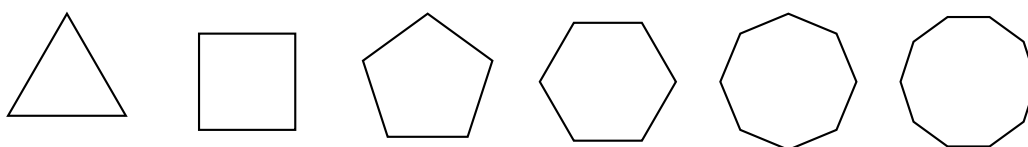
On place des points en  $(-3,0)$ ,  $(-2,0)$ , ...,  $(4,0)$  et à partir de chacun de ces points on trace au moyen de `\rlineto` un vecteur (sans flèche) de coordonnées  $(1,1)$ .

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,linewidth=1.5pt}
\multido{\i=\xmin+1}
  {8}
  {
    \psdots[linewidth=1pt](\i,0)
    \pscustom{
      \moveto(\i,0)
      \rlineto(1,1)
    }
  }
```



On verra d'autres applications de `\multido` dans une future chronique consacrée aux tracés de polygones réguliers.



## Chronique 12

# Ajustement affine

Petite chronique présentant un problème d'ajustement affine tel qu'on peut en rencontrer dans certaines séries de lycée; elle permet de faire le point sur les repères cartésiens et les unités à utiliser dans ce genre d'exercice.

### 12.1 Le problème

Une étude sur une chaîne de magasins a été menée pour mesurer l'impact d'une campagne publicitaire sur les ventes.

Le tableau ci-dessous présente les montants des dépenses publicitaires, en milliers d'euros, et des ventes, en millions d'euros.

Dépenses publicitaires $x_i$ en milliers d'euros	26	27	29	31	32	35
Ventes $y_i$ en millions d'euros	4,5	4,8	4,95	5,1	5,25	5,4

On veut représenter cette série double par un nuage de points, placer le point moyen et tracer la droite d'ajustement de  $y$  en  $x$  obtenue par la méthode des moindres carrés; les calculs seront effectués à la calculatrice.

Enfin on voudrait déterminer graphiquement sur quel montant de ventes on pourrait compter avec des dépenses publicitaires de 28 et de 37 milliers d'euros.

Les calculs donnent  $G(30, 5)$  comme point moyen et  $y = 0,09375x + 2,1875$  pour équation de la droite d'ajustement.

En remplaçant  $x$  par 28 dans  $y = 0,09375x + 2,1875$ , on trouve  $y = 4,8125$  et si on remplace  $x$  par 37, on trouve  $y = 6,65625$ .

Voir la représentation graphique attendue page suivante.

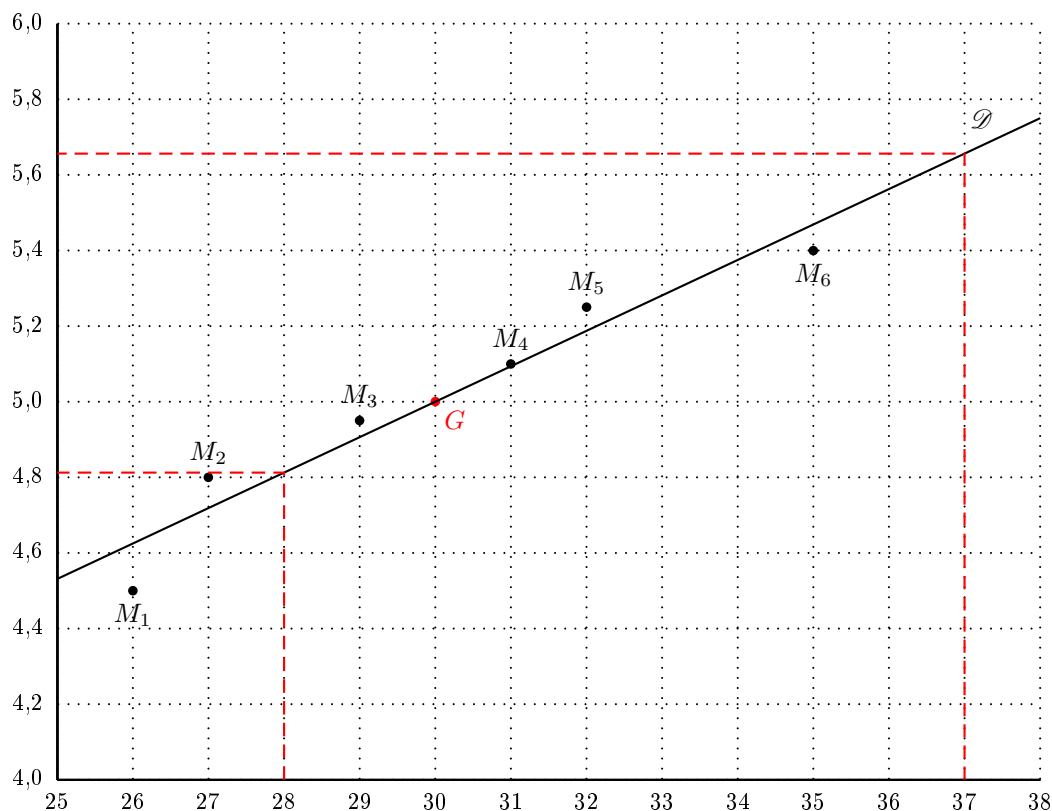
### 12.2 Le repère et les axes

Compte tenu des données du problème, on prendra des valeurs de  $x$  entre 25 et 38 et des valeurs de  $y$  entre 4 et 6. On prend 1 cm comme unité en abscisse et 5 cm en ordonnée.

Les axes seront tracés de `\xmin` à `\xmax` horizontalement, et de `\ymin` à `\ymax` verticalement.

On placera l'origine en  $(25, 4)$ , c'est-à-dire en  $(\texttt{\textbackslash xmin}, \texttt{\textbackslash ymin})$ .

Enfin on retirera les marques d'unités (`ticks=none`) et les légendes automatiques (`labels=none`) sur les axes.



Ce qui donne comme squelette de code :

```
\psset{xunit=1cm,yunit=5cm}
\def\xmin{25} \def\xmax{38}
\def\ymin{4} \def\ymax{6}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\psaxes[labels=none,ticks=none](\xmin,\ymin)(\xmax,\ymax)
...
\end{pspicture}
```

## 12.3 Le quadrillage et les légendes

On ne pourra pas utiliser l'instruction `\psgrid` pour dessiner le quadrillage de la figure ; en effet cette instruction utilise `xunit` et `yunit` pour déterminer l'espacement des divisions principales. Il faudrait alors diviser par 5 l'espacement en ordonnée, sans diviser l'espacement en abscisse, ce qui ne semble pas possible.

On va donc tracer « à la main » le quadrillage en utilisant l'instruction `\multido` deux fois : une fois pour les traits verticaux, une fois pour les traits horizontaux puisqu'il n'y a pas le même nombre de traits dans les deux directions.

### Verticalement

On va tracer 14 segments en pointillés entre les points (25,4) et (25,6), puis entre (26,4) et (26,6) ,... , puis entre (38,4) et (38,6). Et comme on a paramétré le graphique, on utilisera `\ymin` à la place de 4, `\ymax` à la place de 6, et `\xmin` à la place de 25.

Dans la même boucle, on écrira les légendes en utilisant `\uput`.

On aura donc besoin d'une variable de type `integer` qui partira de 25 (`\xmin`) et augmentera de 1 en prenant en tout 14 valeurs.

On aurait pu se contenter de 13 valeurs en partant de 26 pour tracer les droites verticales, mais il faut quand même écrire 25 en dessous de l'origine donc j'ai choisi de démarrer à 25. Voici le code de la boucle `\multido` :

```
\multido{\i=\xmin+1}{14}
{
  \uput[d](\i,\ymin){\small \i}
  \psline[linestyle=dotted](\i,\ymin)(\i,\ymax)
}
```

### Horizontalement

On va tracer des segments en pointillés de (4;25) à (4;38), puis de (4,2;25) à (4,2;38), ..., puis de (6;25) à (6;38).

Au passage on écrira les légendes en utilisant `\uput`.

Comme compteur de `\multido`, il faudra une variable de type `number` qui partira de 4 en allant jusqu'à 6 par pas de 0,2; il faudra donc 11 valeurs.

On va également faire en sorte d'écrire 4,2, 4,4, etc. avec une virgule à la place du point; il faut pour cela charger l'extension `numprint` qui modifie l'écriture décimale en reconnaissant que l'on écrit en français.

Ce package chargé, on utilisera la fonction `\nombre` : ainsi `\nombre{5.4}` donnera 5,4.

On peut également charger l'extension avec l'option `np` pour utiliser le raccourci `\np` à la place de `\nombre`; on charge donc l'extension en tapant :

```
\usepackage[np]{numprint}
```

Voici le code de la boucle :

```
\multido{\n=\ymin+0.2}{11}
{
  \uput[1](\xmin,\n){\small \np{\n}}
  \psline[linestyle=dotted](\xmin,\n)(\xmax,\n)
}
```

L'extension `\numprint` permet aussi de regrouper 3 par 3 les chiffres des nombres qui dépassent trois chiffres; ainsi le nombre 1234567 peut être écrit 1 234 567 en utilisant `\np{1234567}`.

C'est aussi valable pour les chiffres à droite de la virgule; on obtient 3,141 592 653 5 en entrant `\np{3,1415926535}`. On retient les premières décimales de  $\pi$  ( $\pi$ ) en comptant les lettres des mots de la strophe bien connue :

```

| Que j'aime à faire connaître ce nombre utile aux sages
| Immortel Archimède, artiste ingénieur
| ...
```

## 12.4 Les points

Il s'agit maintenant de placer les points du nuage dans le graphique; comme les unités sont adaptées au problème, il n'y a aucune précaution particulière à prendre.

J'ai déjà parlé d'une petite commande personnelle appelée `\point` qui place le point et marque son nom en même temps :

```
\newcommand*\point[4]{%
\psdots(#1,#2)
\uput[#3](#1,#2){#4}}
% permet de placer un point et de marquer son nom en même temps
% 4 paramètres : abscisse, ordonnée, emplacement et nom
```

On traite de façon spéciale le point  $G$  que l'on veut tracer en rouge.

## 12.5 La droite

Pas de problème pour tracer la droite d'équation  $y = 0,09375x + 2,1875$  avec `\psplot`; on entrera comme définition de la fonction `{x 0.09375 mul 2.1875 add}`.

Enfin on tracera des segments en mode tirets (`dashed`) pour faire l'interpolation en  $x = 28$  et l'extrapolation en  $x = 37$ .

## 12.6 Le code complet

Voici le code complet du graphique :

```
\psset{xunit=1cm, yunit=5cm}
\def\xmin{25} \def\xmax{38}
\def\ymin{4.0} \def\ymax{6.0}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
%\psgrid[subgriddivx=5]
\psaxes[labels=none,ticks=none](\xmin,\ymin)(\xmax,\ymax)

\multido{\i=\xmin+1}
{14}%           nombre de traits
{
\uput[d](\i,\ymin){\small \i}%           écrit 25 26 27 etc.
\psline[linestyle=dotted](\i,\ymin)(\i,\ymax)% pointillés verticaux
}

\multido{\n=\ymin+0.2}
{11}%           nombre de traits
{
\uput[l](\xmin,\n){\small \np{\n}}%           écrit 4,0 4,2 etc.
\psline[linestyle=dotted](\xmin,\n)(\xmax,\n)% pointillés horizontaux
}

\point{26}{4.5}{d}{\$M_1$}
\point{27}{4.8}{u}{\$M_2$}
\point{29}{4.95}{u}{\$M_3$}%           points du nuage
\point{31}{5.1}{u}{\$M_4$}
\point{32}{5.25}{u}{\$M_5$}
\point{35}{5.4}{d}{\$M_6$}

\psdots[linecolor=red](30,5)%           point G
\uput[dr](30,5){\red $G$}

\uput[ul](37.5,5.7){$\mathscr{D}$}%           nom de la droite

\psplot{\xmin}{\xmax}{x 0.09375 mul 2.1875 add}% tracé de la droite

\psset{linestyle=dashed, linecolor=red}%           en rouge et en pointillés
\psline(28,\ymin)(28,4.8125)(\xmin,4.8125)$           pour x=28
\psline(37,\ymin)(37,5.65625)(\xmin,5.65625)$           pour x=37

\end{pspicture}
```



## Chronique 13

# Polygones et autres dessins

Dans cette chronique, on va dessiner des polygones réguliers puis des frises circulaires, et on reviendra un peu sur les rosaces.

On va commencer par détailler la construction d'un polygone régulier dans un cas plus intéressant que le triangle ou le carré.

### 13.1 Pentagone

Si on veut dessiner un pentagone régulier inscrit dans un cercle par exemple de rayon 2, ce n'est pas très facile d'utiliser les coordonnées cartésiennes pour déterminer les sommets du pentagone ! On va donc utiliser un repérage polaire et prendre pour sommets les points repérés par  $(2;0)$ ,  $(2;72)$ ,  $\dots(2;288)$ . Je rappelle que les angles sont exprimés en degrés (voir page 55).

Que faut-il faire pour tracer un pentagone régulier ?

Se placer au premier sommet avec l'instruction `\moveto`, puis tracer un segment de ce premier sommet au deuxième avec l'instruction `\lineto` et ainsi de suite jusqu'au dernier sommet.

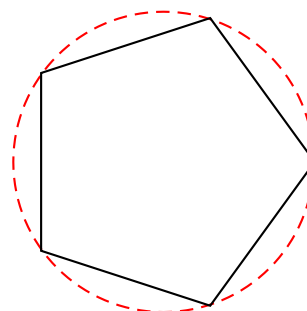
Enfin, il faut refermer la figure et revenir au premier sommet.

Naturellement on a besoin d'une boucle définie par `\multido` qui prendra autant de valeurs que le nombre de côtés du polygone, c'est-à-dire 5. La variable servant de compteur peut être de type `integer`.

Enfin, on a déjà vu qu'utiliser l'instruction `\moveto` nécessite d'être dans un environnement `\pscustom`. Et dans cet environnement, on revient au point de départ en fermant le chemin par l'instruction `\closepath`.

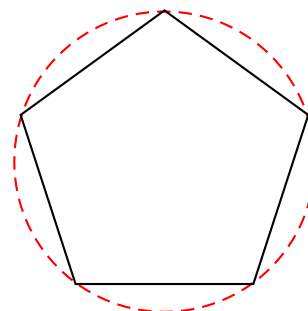
Voici le code du polygone et le résultat obtenu :

```
\psset{unit=1cm}
\def\xmin{-2} \def\xmax{2} \def\ymin{-2} \def\ymax{2}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pscircle[linecolor=red,linestyle=dashed](0,0){2}
\pscustom{
  \moveto(2;0)%      premier sommet
  \multido{\i=0+72}% variable
    {5}%            nombre de côtés
    {\lineto(2;\i)}% tracé du côté
  \closepath%       retour au départ
}%                 fin du \pscustom
\end{pspicture}
```



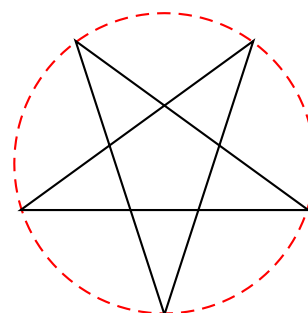
Le sommet de départ est en  $(2;0)$  ; on pourrait le mettre à n'importe quel endroit sur le cercle. On va pour cela créer une variable `\deb` qui va désigner l'angle repérant le sommet initial :

```
\psset{unit=1cm}
\def\xmin{-2} \def\xmax{2} \def\ymin{-2} \def\ymax{2}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pscircle[linecolor=red,linestyle=dashed](0,0){2}
\def\deb{18}
\pscustom{\moveto(2;\deb)
\multido{\i=\deb+72}{5}
{\lineto(2;\i)}}
\closepath
\end{pspicture}
```



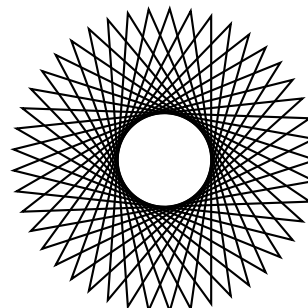
En donnant à `\deb` la valeur  $-18$  on aura un pentagone avec la pointe en bas. Un pentagone régulier peut aussi être croisé ; il suffit de relier les sommets de 2 en 2 (puisque 5 et 2 sont premiers entre eux) :

```
\psset{unit=1cm}
\def\xmin{-2} \def\xmax{2} \def\ymin{-2} \def\ymax{2}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pscircle[linecolor=red,linestyle=dashed](0,0){2}
\def\deb{-18}
\pscustom{\moveto(2;\deb)
\multido{\i=\deb+144}{5}
{\lineto(2;\i)}}
\closepath
\end{pspicture}
```



Enfin on va faire tourner cette étoile pour créer un joli dessin : on va donc insérer le code du tracé du pentagone croisé dans une boucle `\multido` qui prendra 9 valeurs par pas de 8. On se passera du cercle et la variable `\deb` sera remplacée par le compteur `\I` de cette nouvelle boucle :

```
\psset{unit=1cm}
\def\xmin{-2} \def\xmax{2} \def\ymin{-2} \def\ymax{2}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\multido{\I=0+8}{9}
{
\pscustom{\moveto(2;\I)
\multido{\i=\I+144}{5}
{\lineto(2;\i)}}
\closepath
}
\end{pspicture}
```

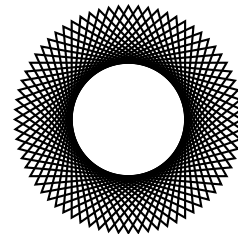
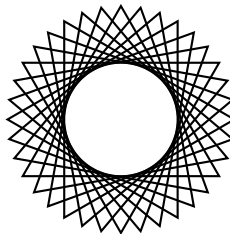
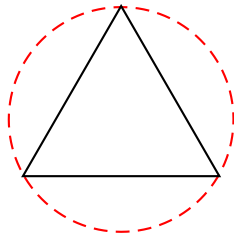
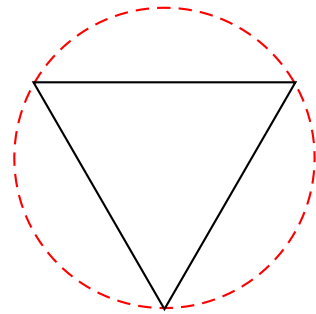


Joli, non ?

## 13.2 Triangle

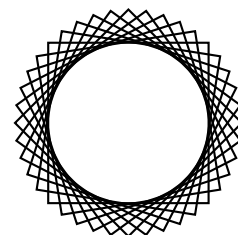
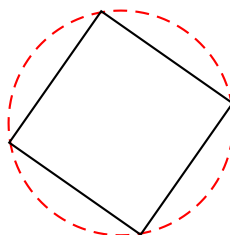
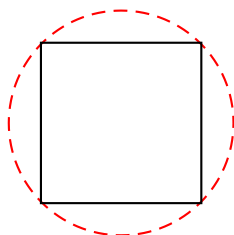
Ce que l'on a fait pour le pentagone régulier, on peut le faire pour le triangle équilatéral :

```
\psset{unit=1cm}
\def\xmin{-2} \def\xmax{2} \def\ymin{-2} \def\ymax{2}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pscircle[linecolor=red,linestyle=dashed](0,0){2}
\def\deb{30}
\pscustom{
\moveto(2;\deb)
\multido{\i=\deb+120}{3}
{\lineto(2;\i)}
\closepath
}
\end{pspicture}
```



## 13.3 Carré

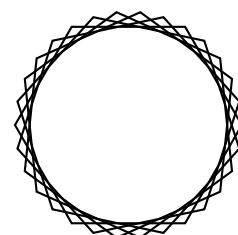
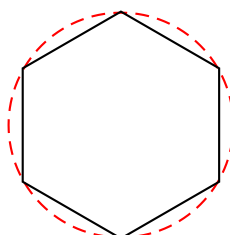
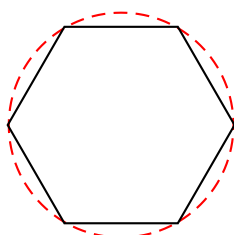
Pour le carré, on modifie le nombre de côtés et l'angle de décalage :



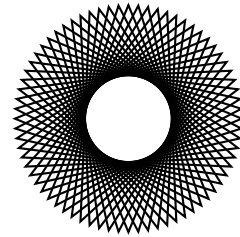
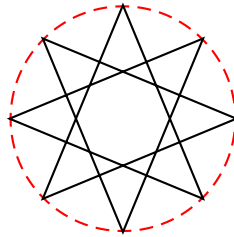
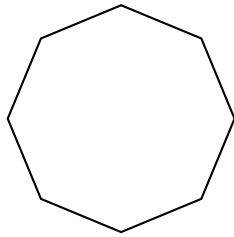
## 13.4 Autres polygones réguliers

On peut décliner le procédé en changeant le nombre de côtés.

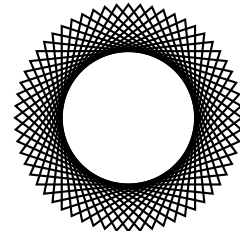
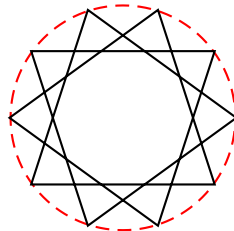
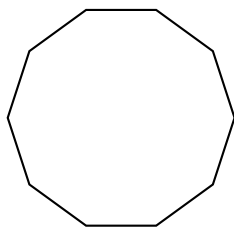
### 13.4.1 Hexagone



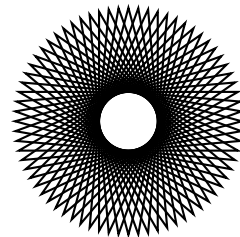
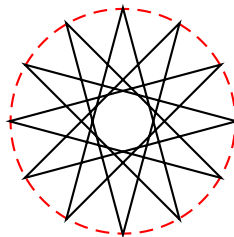
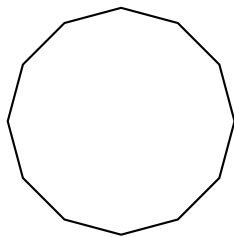
### 13.4.2 Octogone



### 13.4.3 Décagone

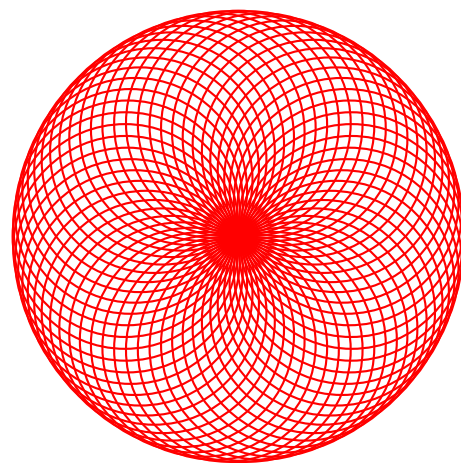
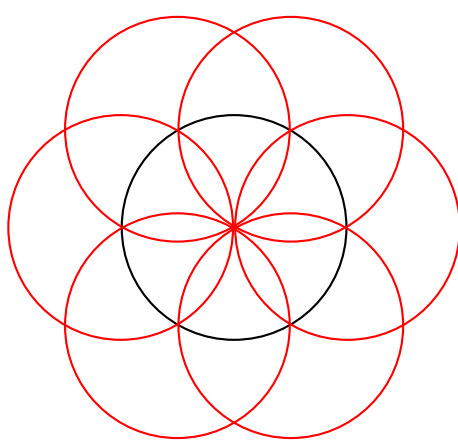


### 13.4.4 Dodécagone



## 13.5 Rosaces

On a déjà vu la figure de gauche à la page 55, et il y a peu de changements à effectuer pour obtenir la figure de droite :



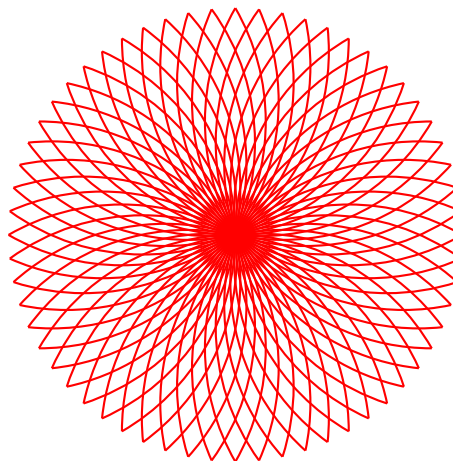
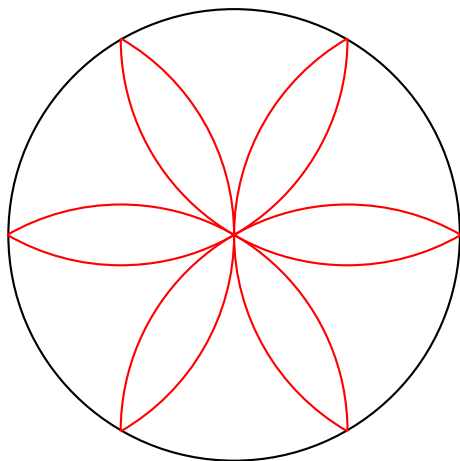
Voici les deux codes pour vous en persuader :

```
\psset{unit=0.5cm}
\def\xmin{-6.5} \def\xmax{6.5}
\def\ymin{-6.1} \def\ymax{6.1}
\begin{pspicture}
  (\xmin,\ymin)(\xmax,\ymax)
\pscircle(0,0){3}
\multido{\i=0+60}
  {6}
  {\pscircle[linecolor=red](3;\i){3}}
\end{pspicture}
```

```
\psset{unit=0.5cm}
\def\xmin{-6.5} \def\xmax{6.5}
\def\ymin{-6.5} \def\ymax{6.1}
\begin{pspicture}
  (\xmin,\ymin)(\xmax,\ymax)
%\pscircle(0,0){3}% pas de cercle noir
\multido{\i=0+6}
  {60}
  {\pscircle[linecolor=red](3;\i){3}}
\end{pspicture}
```

Au lieu de tracer 6 cercles dont les centres sont décalés de  $60^\circ$ , on en trace 60 dont les centres sont décalés de  $6^\circ$ .

Il faut un peu plus de travail pour obtenir ces deux dessins :



Dans la figure de gauche, on ne trace plus des cercles (`\pscircle`) mais des arcs de cercles (`\psarc`) ; mais pour cette dernière instruction, il faut donner l'angle de départ de l'arc, et l'angle d'arrivée. Et ces deux angles varient en fonction de la position du centre de l'arc sur le cercle noir.

Il faut également savoir que les arcs de cercles se tracent dans le sens trigonométrique : l'instruction `\psarc(abs,ord){rayon}{ang début}{ang fin}` va tracer un arc de cercle de rayon `rayon`, centré sur le point de coordonnées `(abs,ord)` et allant dans le sens trigonométrique de `ang début` à `ang fin`.

Au lieu des coordonnées cartésiennes, on utilise ici aussi un repérage polaire pour déterminer le centre ; le centre sera donc repéré par le couple `(rayon ; angle du centre)`.

Le tableau suivant donne l'angle de départ et l'angle d'arrivée pour chaque position du centre :

angle du centre	angle début	angle fin
$0^\circ$	$120^\circ$	$240^\circ$
$60^\circ$	$180^\circ$	$300^\circ$
$120^\circ$	$240^\circ$	$360^\circ$
$180^\circ$	$300^\circ$	$420^\circ$
$240^\circ$	$360^\circ$	$480^\circ$
$300^\circ$	$420^\circ$	$540^\circ$

Il faut donc trois variables entières, une pour l'angle du centre, une pour l'angle de départ de l'arc, une pour l'angle d'arrivée; ces trois variables prenant le même nombre de valeurs, une seule boucle suffit.

Le code du dessin de gauche est :

```
\psset{unit=1cm}
\def\r{3}%                rayon
\def\xmin{-\r} \def\xmax{\r}
\def\ymin{-\r} \def\ymax{\r}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pscircle(0,0){\r}%       cercle noir
\multido{\i=0+60,\ix=120+60,\iy=240+60}%  variables
    {6}%                  nombre de cercles
    {\psarc[linecolor=red](\r;\i){\r}{\ix}{\iy}}% arc
\end{pspicture}
```

Pour obtenir le dessin de droite, il suffit de remplacer la boucle `\multido` par celle-ci :

```
\multido{\i=0+6,\ix=120+6,\iy=240+6}
    {60}
    {\psarc[linecolor=red](\r;\i){\r}{\ix}{\iy}}
```

Au lieu de 6 arcs de cercle décalés de 60°, on dessine 60 arcs de cercle décalés de 6°.

## Chronique 14

# Document prof-élève

Il peut être intéressant d'avoir des documents mixtes prof-élève, c'est-à-dire une version pour le professeur et une version à distribuer aux élèves. On va voir comment intégrer une correction dans un document, et faire afficher deux versions, avec et sans correction, bref comment on écrit du « texte conditionnel ».

### 14.1 Premier exemple

On écrit le texte d'un exercice en y intégrant les réponses qu'on aimerait ne pas montrer aux élèves. Voici la version élève :

Soit  $f$  la fonction définie sur  $\mathbf{R}$  par  $f(x) = 2x^3 - 3x^2 - 12x - 2$ .

1. Déterminer la fonction  $f'$  dérivée de la fonction  $f$ .
2. Déterminer le signe de  $f'$  ; en déduire les variations de la fonction  $f$ .
3. ...

Et la version professeur :

Soit  $f$  la fonction définie sur  $\mathbf{R}$  par  $f(x) = 2x^3 - 3x^2 - 12x - 2$ .

1. Déterminer la fonction  $f'$  dérivée de la fonction  $f$ .  
 $f'(x) = 6x^2 - 6x - 12$
2. Déterminer le signe de  $f'$  ; en déduire les variations de la fonction  $f$ .  
 $f'(x) = 6(x^2 - x - 2) = 6(x + 1)(x - 2)$   
etc.
3. ...

Il faut définir une variable qu'on va appeler `\prof` qui tient lieu de booléen ; si cette variable est activée, on affiche le corrigé pour le professeur, sinon on n'affiche que le texte pour les élèves.

On définit la variable par `\def\prof` ou de manière équivalente par `\newcommand{\prof}{};` ; le mieux est de placer cette ligne avant `\begin{document}`.

Pour désactiver la définition de la variable `\prof`, on met le caractère `%` au début de la ligne contenant la définition de cette variable.

Le « si... alors... fin du si » se programme ainsi :

```
\ifdefined\prof
...
\fi
```

Le texte complet donnant l'exemple précédent est :

```

Soit  $f$  la fonction définie sur  $\mathbf{R}$  par  $f(x)=2x^3-3x^2-12x-2$ .
\begin{enumerate}
\item Déterminer la fonction  $f'$  dérivée de la fonction  $f$ .
    \ifdefined\prof
    {\red
       $f'(x)=6x^2-6x-12$ }
    \fi
\item Déterminer le signe de  $f'$ ;
    en déduire les variations de la fonction  $f$ .
    \ifdefined\prof
    {\red
       $f'(x)=6(x^2-x-2)=6(x+1)(x-2)$ 
      etc.}
    \fi
\item  $\cdots$ 
\end{enumerate}

```

Dans cet exemple, `\R` est une commande personnelle qui affiche **R**, ensemble des réels :

```
\newcommand{\R}{\textbf{R}}
```

## 14.2 Autre version

On peut aussi vouloir, dans la version professeur, ne pas afficher le texte donné aux élèves :

Soit  $f$  la fonction définie sur  $\mathbf{R}$  par  $f(x) = 2x^3 - 3x^2 - 12x - 2$ .

1.  $f'(x) = 6x^2 - 6x - 12$
2.  $f'(x) = 6(x^2 - x - 2) = 6(x+1)(x-2)$   
etc.
3. ...

Il faut, pour cela, compléter le `si... alors...` par un `sinon`, autrement dit par `\else`, qui va contenir le texte pour les élèves ; ce texte ne sera pas affiché quand la variable `\prof` sera activée.

Voici le nouveau code :

```

Soit  $f$  la fonction définie sur  $\mathbf{R}$  par  $f(x)=2x^3-3x^2-12x-2$ .
\begin{enumerate}
\item \ifdefined\prof
    {\red
       $f'(x)=6x^2-6x-12$ }
    \else Déterminer la fonction  $f'$  dérivée de la fonction  $f$ .
    \fi
\item \ifdefined\prof
    {\red
       $f'(x)=6(x^2-x-2)=6(x+1)(x-2)$ 
      etc.}
    \else Déterminer le signe de  $f'$ ;
    en déduire les variations de la fonction  $f$ .
    \fi
\item  $\cdots$ 
\end{enumerate}

```



### 14.3 Deuxième exemple

On peut vouloir faire remplir aux élèves un tableau de calculs à effectuer à la calculatrice, et avoir sous la main le même tableau avec les réponses.

Version élève :

$2,5 + 3,4 \times 1,8$	$\sqrt{6,25} + 3,01$
...	...

Et version professeur :

$2,5 + 3,4 \times 1,8$	8,62	$\sqrt{6,25} + 3,01$	5,51
...	...	...	...

On va écrire la réponse dans le tableau et jouer sur la couleur de cette réponse; on l'écrit en rouge dans la version professeur, et en blanc dans la version élève : elle est alors invisible.

On va donc définir une couleur `hhhhhh` qui sera rouge ou blanche :

```
\ifdefined\prof
\newrgbcolor{hhhhh}{1 0 0}%    rouge (pour professeur)
\else
\newrgbcolor{hhhhh}{1 1 1}%    blanc (pour élève)
\fi
```

La couleur s'appelle `hhhhh` (sans `\`) mais on l'utilise par l'instruction `\hhhhh` (avec `\`).

On peut donner n'importe quel nom à cette couleur, à condition qu'il n'existe pas déjà!

Pour la définition de `\newrgbcolor`, voir page 7 de la chronique 1.

On écrira un résultat dans le tableau ainsi : `\hhhhh 8,62` ou même sans accolades.

Voici le code du tableau :

```
{%          début d'environnement
\renewcommand{\arraystretch}{1.3}%  on élargit les lignes
$\begin{array}{|cc|cc|}%          4 colonnes
\hline
2,5+3,4\times 1,8 & \hspace*{1cm} & \hhhhh 8,62\hspace*{1cm} & \\
& & & \displaystyle\sqrt{6,25}+ 3,01 \\
& & \hspace*{1cm} & \hhhhh 5,51 \hspace*{1cm}\\
\hline
\cdots & \hhhhh \cdots & \cdots & \hhhhh \cdots\\
\hline
\end{array}$
}%          fin de la redéfinition de \arraystretch
```

Les plus vigilants d'entre vous auront vu la présence de `\displaystyle` dans le code du tableau; cette commande mérite une chronique séparée...



## Chronique 15

# Displaystyle

Tout ce qui va être dit dans cette chronique nécessite d'avoir chargé l'extension `amsmath` en entrant l'instruction `\usepackage{amsmath}` dans le préambule du document.

### 15.1 Définition

Un document qu'il faut absolument posséder s'intitule « LISTE DES COMMANDES LATEX », et on peut le télécharger sur le site de l'ENS :

[http://www.math.ens.fr/~millien/telatex/liste\\_commandes.pdf](http://www.math.ens.fr/~millien/telatex/liste_commandes.pdf)

Voici ce qui est donné dans ce document comme définition de `\displaystyle` :

Force le mode mathématique courant à être comme à l'intérieur d'un environnement mathématique mis en évidence par `\[... \]`.  
Ne s'utilise qu'à l'intérieur du mode mathématique.

En  $\text{\LaTeX}$  il existe deux modes d'écriture de formules mathématiques : le mode en ligne (`\[... \]`) et le mode hors ligne (`\[... \]`) dans lequel les formules sont détachées du texte et centrées.

Dans ces deux modes, les formules ne se présentent pas tout à fait sous la même forme ; la commande `\displaystyle` sert à donner aux formules le même aspect dans le mode en ligne que dans le mode hors ligne.

Mais comme on peut trouver que taper `\displaystyle` est un peu long, on va gagner du temps en créant un raccourci ; il suffit d'écrire dans le préambule du document :

`\newcommand{\ds}{\displaystyle}`

On tapera donc `\ds` chaque fois qu'on voudra taper `\displaystyle`.

### 15.2 Exemples

#### 15.2.1 Fraction

L'instruction qui permet d'écrire des fractions est `\frac` : elle nécessite deux paramètres, le numérateur et le dénominateur.

Si on tape (en hors ligne) `\[ \frac{x+1}{x^2+2} \]`, on obtient :

$$\frac{x+1}{x^2+2}$$

Et si l'on tape (en ligne) `\frac{x+1}{x^2+2}`, on obtient :  $\frac{x+1}{x^2+2}$

Vous avez certainement compris à quoi servait `\displaystyle` et son raccourci `\ds` :

il suffit de taper `\ds\frac{x+1}{x^2+2}` pour obtenir  $\frac{x+1}{x^2+2}$

Certes, me direz-vous, il existe l'instruction `\dfraction` qui écrit correctement les fractions en ligne ; et bien oui, cette instruction n'est qu'un raccourci de l'instruction `\displaystyle\frac` comme le précise la documentation du package `amsmath` ; cette documentation est disponible à l'adresse :  
<ftp://ftp.ams.org/ams/doc/amsmath/amslatex/amsldoc.pdf>

### 15.2.2 Racine

En écrivant `\sqrt{13}`, on obtient  $\sqrt{13}$ , et en écrivant `\[\sqrt{13}\]`, on obtient

$$\sqrt{13}$$

Il n'y a guère de différence dans les graphismes de ces deux formules ; essayons avec autre chose...  
 En écrivant `\sqrt{6,25}`, on obtient  $\sqrt{6,25}$ , et en écrivant `\[\sqrt{6,25}\]`, on obtient

$$\sqrt{6,25}$$

Là, on voit mieux la différence et l'écriture hors ligne est bien plus lisible que l'écriture en ligne car la barre du radical est plus éloignée du nombre.

Pour écrire en ligne  $\sqrt{6,25}$ , on entrera donc `\ds\sqrt{6,25}`.

Et pourquoi ne pas faire avec la racine ce qui existe avec la fraction ?

On va créer une nouvelle commande `\dsqrt` de la façon suivante :

```
\newcommand{\dsqrt}{\displaystyle\sqrt}
```

On tapera donc `\dsqrt{6,25}` pour obtenir  $\sqrt{6,25}$  et `\dsqrt[3]{6,25}` pour obtenir  $\sqrt[3]{6,25}$ .

Remarque : quand je crée une nouvelle commande, j'évite d'inclure dans la nouvelle définition une commande déjà redéfinie ; autrement dit, j'écris `\newcommand{\dsqrt}{\displaystyle\sqrt}` et pas `\newcommand{\dsqrt}{\ds\sqrt}`. Comme ça si la commande `\ds` saute pour une raison ou pour une autre, la définition de `\dsqrt` reste valide.

### 15.2.3 Intégrale

Pour écrire une intégrale, on utilise l'instruction `\int`. Attention, la lettre `d` indiquant la différentiation doit être écrite en romain et pas en italiques ; on va donc commencer par créer une commande qui fait cela :

```
\renewcommand{\d}{\,\text{d}}
```

On utilise `\renewcommand` et pas `\newcommand` car la commande `\d` existe déjà en L<sup>A</sup>T<sub>E</sub>X (elle place un point en dessous de la lettre qui suit).

Remarque : si on utilise le package `hyperref` pour créer des liens dans le fichier pdf, il faut redéfinir la commande `\d` après le chargement de cette extension.

Une intégrale en ligne définie par `\int_a^b f(t) \d t` donne  $\int_a^b f(t) dt$ , tandis que l'intégrale hors ligne définie par `\[\int_a^b f(t) \d t\]` donne

$$\int_a^b f(t) dt$$

On va donc créer la commande

```
\newcommand{\dint}{\displaystyle\int}
```

qui permettra d'écrire en ligne de belles intégrales :

On obtiendra la formule  $I = \int_a^b f(t) dt$  en tapant `\dint_a^b f(t) \d t`.

### 15.2.4 Somme et produit

La somme et le produit seront traités de la même façon ; on obtient deux résultats assez décevants en ligne  $\sum_{k=1}^{10} k^2$  et  $\prod_{k=1}^{10} k$  en entrant `\sum_{k=1}^{10} k^2` et `\prod_{k=1}^{10} k`.

Les résultats hors ligne sont ceux que l'on attend :

$$\sum_{k=1}^{10} k^2 \text{ et } \prod_{k=1}^{10} k$$

Deux commandes sont donc à créer :

```
\newcommand{\ds}\displaystyle\sum
\newcommand{\dprod}\displaystyle\prod
```

La somme  $\sum_{k=1}^{10} k^2$  et le produit  $\prod_{k=1}^{10} k$  écrits en ligne en utilisant `\ds` à la place de `sum`, et `\dprod` à la place de `prod`, donnent alors de beaux résultats.

### 15.2.5 Limite

Un autre exemple va concerner l'écriture des limites.

Quand on écrit `\lim_{x \to +\infty} f(x)`, on obtient  $\lim_{x \rightarrow +\infty} f(x)$ .

Vraiment moche !

Nouvelle utilisation de `\displaystyle` dans une nouvelle commande :

```
\newcommand{\dlim}\displaystyle\lim
```

Et maintenant quand on écrit `\dlim_{x \to +\infty} f(x)`, on obtient  $\lim_{x \rightarrow +\infty} f(x)$ .

C'est quand même mieux !

### 15.2.6 Nombre de combinaisons

Le dernier exemple va concerner le nombre de combinaisons de  $p$  parmi  $n$  qui s'écrit `\binom{n}{p}` et qui donne  $\binom{n}{p}$ .

Là aussi, en entrant `\ds\binom{n}{p}`, on obtient  $\binom{n}{p}$ .

Mais il n'est pas utile de définir une nouvelle commande car la commande `\dbinom` existe déjà dans le package `amsmath` ; en entrant `\dbinom{n}{p}`, on obtient l'affichage souhaité  $\binom{n}{p}$ .

## 15.3 Raccourcis

En résumé, voici tous les raccourcis définis dans cette chronique :

<code>\newcommand{\ds}\displaystyle%</code>	<code>displaystyle</code>
<code>\newcommand{\dsqrt}\displaystyle\sqrt%</code>	<code>racine</code>
<code>\renewcommand{\d}\{\,\text{d}\}%</code>	<code>le d de différentiation</code>
<code>\newcommand{\ds}\displaystyle\sum%</code>	<code>somme</code>
<code>\newcommand{\dprod}\displaystyle\prod%</code>	<code>produit</code>
<code>\newcommand{\dlim}\displaystyle\lim%</code>	<code>limite</code>



## Chronique 16

# Diagrammes

### 16.1 Matriochkas

Quand on charge une extension, il se peut que celle-ci charge aussi une ou plusieurs autres extensions qui elles-mêmes peuvent charger d'autres extensions, etc.

Par exemple, l'extension `pst-all` appelle différentes extensions dont on a déjà parlé (ou pas!) : `pstricks`, `pst-plot`, `pst-node`, `pst-tree`, `pst-grad`, `pst-coil`, `pst-text`, `pst-3d`, `pst-eps`, `pst-fill`, `pstricks-add` et `multido`.

Quand on charge `pstricks-add`, on charge en plus `pstricks` et `pst-math`, puis on recharge `pst-plot`, `pst-node`, `pst-3d` et `multido`.

Et quand on charge...

Bref, vous avez compris.

Donc en appelant l'extension `pst-all` par `\usepackage{pst-all}`, on charge la plupart des extensions dont on a besoin pour travailler en `psTricks`; à partir de maintenant, on pourra donc se contenter de charger cette extension en laissant tomber les `\usepackage{pstricks-add}` et autres `\usepackage{pst-tree}`.

Au passage, l'extension `xcolor` est chargée par `pstricks`; on peut donc également supprimer `\usepackage{xcolor}`.

Comment vérifier tout ça ?

Un « package » est un fichier ayant pour extension `sty`. En éditant le fichier `pst-all.sty`, par exemple avec l'éditeur  $\text{\LaTeX}$  que vous utilisez, vous verrez au début du fichier :

- la ligne `\RequirePackage{pstricks}` qui charge le package `pstricks`;
- la ligne `\RequirePackage{pst-plot}` qui charge le package `pst-plot`;
- etc.

### 16.2 Diagramme à barres

Voici un tableau donnant la répartition des notes à un test :

Notes	$x_i$	0	1	2	3	4	5	6	7	8	9	10
Effectifs	$y_i$	0	1	2	3	5	7	6	0	5	2	1

On veut représenter cette série par un diagramme en barres.

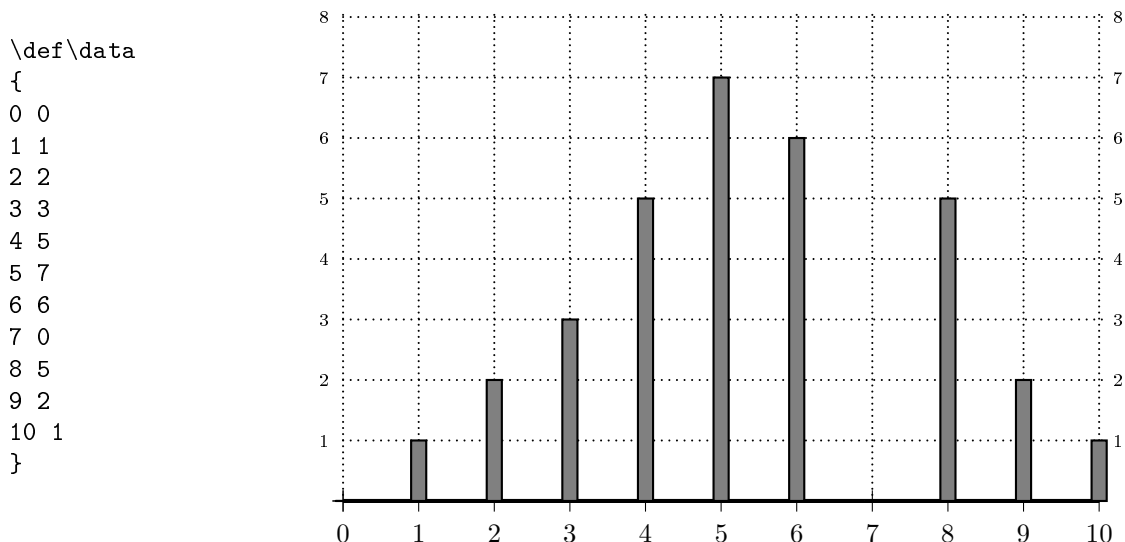
Pour cela on va utiliser l'instruction `\listplot` dont on va définir les paramètres :

```
plotstyle = bar      pour tracer des barres;
barwidth  = 0.2cm    pour définir la largeur des barres;
fillcolor = black    pour définir la couleur de remplissage des barres;
fillstyle = solid     pour définir le style de remplissage des barres.
```

À part `solid`, les valeurs qu'on peut donner à `fillstyle` sont `crosshatch`, `vlines` et `hlines`, et leurs versions étoilées `crosshatch*`, `vlines*` et `hlines*`. À essayer !

Ensuite il faut définir les données, le plus simple étant de les placer dans une variable. Les données sont entrées sous la forme d'une liste  $x_1$   $y_1$   $x_2$   $y_2$  etc. (abscisses et ordonnées séparées par un espace).

On peut entrer les notes dans un ordre quelconque (ce que je déconseille) et il vaut mieux entrer les notes ayant pour effectif 0. Et pour une meilleure lisibilité, on entrera les données en colonnes. Voici comment on définit les données appelées `\data` et le graphique obtenu :



Le code complet du graphique est :

```
\psset{xunit=1cm,yunit=0.8cm}
\begin{pspicture}(-1,-1)(10,8)
\psgrid[subgriddiv=1,griddots=10,gridlabels=0](0,0)(10,8)% quadrillage
\psaxes[linewidth=1.5pt](0,0)(10,0)% axe horizontal seulement
\multido{\n=1+1}
{8}% nombre de graduations
{
\uput[l](0,\n){\scriptsize \n}% graduations côté gauche
\uput[r](10,\n){\scriptsize \n}% graduations côté droit
}
\listplot[plotstyle=bar,barwidth=0.2cm,%
fillcolor=gray,fillstyle=solid]{\data}% tracés des barres
\end{pspicture}
```

En changeant les paramètres de `\listplot`, on peut avoir des effets plus originaux.

Voici deux diagrammes basés sur la même série de données ; seule la ligne `\listplot` a été modifiée.

Pour le premier diagramme, on a entré :

```
\listplot[shadow=true,plotstyle=bar,%
barwidth=0.4cm,fillcolor=red!50,fillstyle=solid]{\data}
```

Deux explications sur les paramètres : `shadow=true` trace une ombre derrière chaque rectangle, et `fillcolor=red!50` remplit de rouge à 50 %.

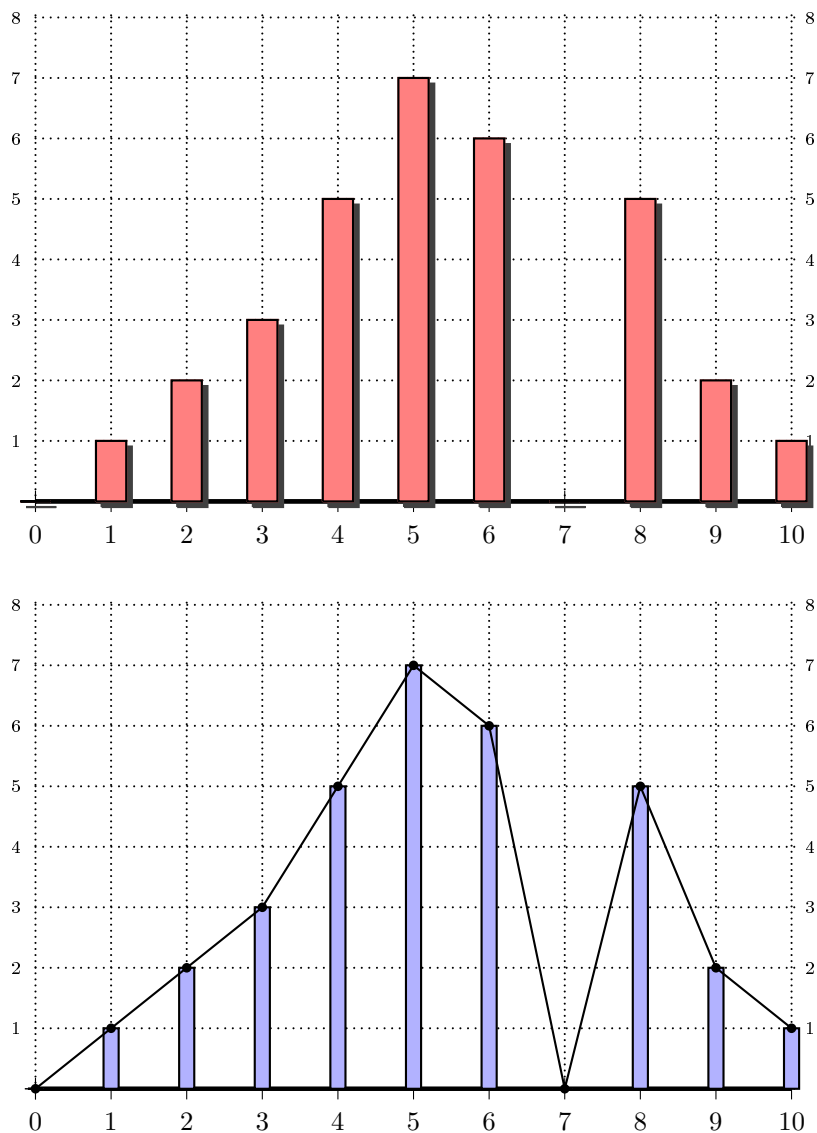
Pour le deuxième diagramme, on a entré :

```
\listplot[plotstyle=bar,barwidth=0.2cm,%
fillcolor=blue!30,fillstyle=solid]{\data}
\listplot[showpoints=true]{\data}
```



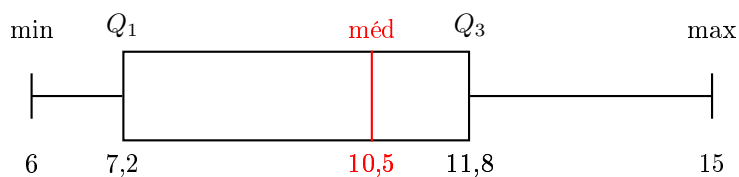
Le deuxième `\listplot` permet de tracer une courbe qui relie les sommets des rectangles, avec affichage des points défini par `showpoints=true`.

Refaites ce même graphique en intervertissant les données dans la définition de `\data` pour voir !



### 16.3 Diagramme en boîte

Voilà un diagramme en boîte qui représente une série dans laquelle le minimum est 6, le premier quartile 7,2, la médiane 10,5, le troisième quartile 11,8 et le maximum 15 :



Ce diagramme est obtenu par l'instruction :

```
\diagboite{6}{7.2}{10.5}{11.8}{15}{1cm}
```

dans laquelle `\diagboite` est une nouvelle commande que l'on va décrire.

Cette commande `\diagboite` a besoin de six paramètres : le minimum de la série, son premier quartile, sa médiane, son troisième quartile et son maximum, qu'il faut entrer impérativement dans cet ordre sous peine de catastrophe ! Le dernier paramètre est l'unité en  $x$  qu'il faut ajuster en fonction des données que l'on veut représenter.

L'unité en  $y$  est fixée à 0,3 cm.

On remarque que l'on rentre comme paramètre le nombre 10.5 avec un point comme séparateur décimal, et qu'à l'affichage on a 10,5 avec une virgule : c'est le package `numprint` qui fait ce travail. Il faut donc avoir entré dans le préambule : `\usepackage[np]{numprint}`.

Le code de cette commande est :

```
\newcommand{\diagboite}[6]
{
%%%%\diagboite{min}{Q1}{méd}{Q3}{max}{xunit}
\psset{xunit=#6,yunit=0.3cm}%          unités
\begin{pspicture}(\#1,-5)(\#5,5)%      zone de dessin
%%%% dessin du diagramme
\psline(\#1,-1)(\#1,1)%                petit trait vertical de début
\psline(\#1,0)(\#2,0)%                 moustache gauche de min à Q1
\psframe(\#2,-2)(\#4,2)%               boîte de Q1 à Q3
\psline[linecolor=red](\#3,-2)(\#3,2)% trait vertical rouge de médiane
\psline(\#4,0)(\#5,0)%                 moustache droite de Q3 à max
\psline(\#5,-1)(\#5,1)%                petit trait vertical de fin
%%%% affichage des valeurs
\uput[d](\#1,-2){\np{\#1}}%            minimum
\uput[d](\#2,-2){\np{\#2}}%            Q1
\uput[d](\#3,-2){\red \np{\#3}}%      médiane
\uput[d](\#4,-2){\np{\#4}}%            Q3
\uput[d](\#5,-2){\np{\#5}}%            maximum
%%%% labels
\uput[u](\#1,2){min}%                  min
\uput[u](\#2,2){$Q_1$}%                Q1
\uput[u](\#3,2){\red méd}%            méd
\uput[u](\#4,2){$Q_3$}%                Q3
\uput[u](\#5,2){max}%                 max
\end{pspicture}
}
```

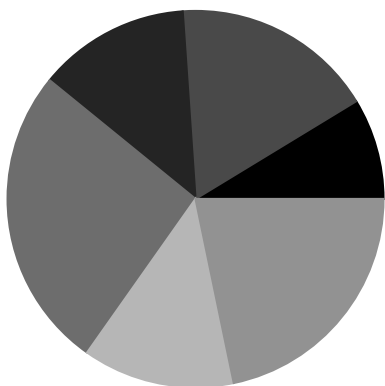
## 16.4 Diagramme circulaire

Autre diagramme qu'il est très facile de construire en L<sup>A</sup>T<sub>E</sub>X, le diagramme circulaire ; pas besoin de créer une nouvelle instruction, celle qu'il faut existe : c'est `\psChart` (attention au C majuscule).

Admettons que l'on ait à représenter par un diagramme circulaire la série suivante :

A	B	C	D	E	F
10	20	15	30	15	25

Voici un diagramme circulaire (obtenu sans aucune précaution) et son code (assez simple) :



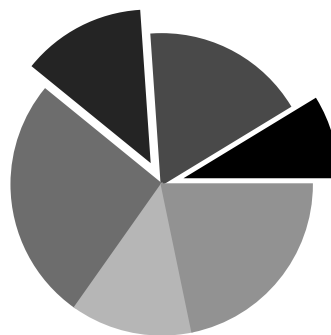
```
\psset{unit=1cm}
\begin{pspicture*}(-2,-2)(2,3)
\psChart{10,20,15,30,15,25}{}{2.5}
\end{pspicture*}
```

On voit que `\psChart` a besoin de trois paramètres :

- le premier est la liste des valeurs, séparées par une virgule ;
- le troisième est le rayon du cercle qui sera tracé ;
- le deuxième est la liste des secteurs qui vont être éclatés.

Exemple de diagramme avec des secteurs éclatés (1 et 3) :

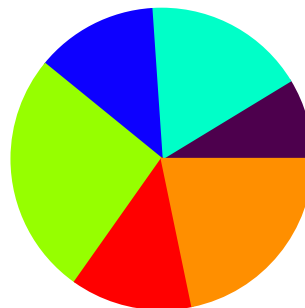
```
\psset{unit=1cm}
\begin{pspicture*}(-3,-3)(3,3)
\psChart{10,20,15,30,15,25}{1,3}{2}
\end{pspicture*}
```



L'éclatement peut être modifié avec la variable `chartSep` qui, normalement, est à 10 points.

Par défaut, le diagramme circulaire est en niveaux de gris ; on peut naturellement demander des couleurs avec l'option `chartColor=color`.

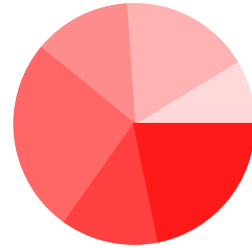
```
\psset{unit=1cm}
\begin{pspicture*}(-3,-3)(3,3)
\psChart[chartColor=color]{10,20,15,30,15,25}{}{2}
\end{pspicture*}
```



Les couleurs sont prédéfinies, mais on peut les changer si on le souhaite ; il suffit d'utiliser la variable `userColor`.

Pour tracer des secteurs en niveaux de rouge, on entrera :

```
\psset{unit=1cm}
\begin{pspicture*}(-2,-2)(2,2)
\psChart[chartColor=color,%
  userColor={red!15,red!30,red!45,red!60,red!75,red}]%
  {10,20,15,30,15,25}{{1.6}}
\end{pspicture*}
```

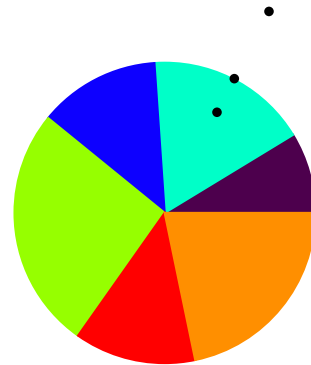


Il suffit d'entrer la liste des couleurs souhaitées, une par secteur.

Enfin, il est très facile de marquer des labels correspondant aux secteurs circulaires ; la commande `\psChart` définit automatiquement des points pour chaque secteur.

Dans le diagramme ci-contre de rayon  $r$ , on a défini trois points pour le secteur 2, situés sur la bissectrice de l'angle :

- le point intérieur est situé à une distance de  $0,75r$  du centre et a pour coordonnées `psChartI2`, où I signifie In et où 2 est le numéro du secteur ;
- le point situé sur le cercle a pour coordonnées `psChart2` ;
- le point extérieur est situé à une distance de  $1,5r$  du centre et a pour coordonnées `psChartO2`, la lettre O signifiant Out.

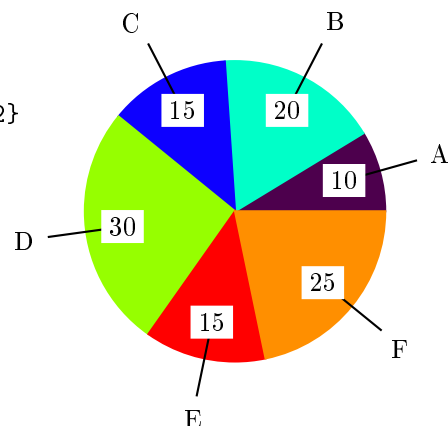


Ces trois points sont tracés par l'instruction

```
\psdots(psChartI2)(psChart2)(psChartO2).
```

Ce qui permet de réaliser un diagramme de ce type :

```
\psset{unit=1cm}
\begin{pspicture*}(-3,-3)(3,3)
\psset{chartNode0=1.25}% 1.5 par défaut
\psChart[chartColor=color]{10,20,15,30,15,25}{{2}}
\multido{\i=1+1}{6}
  {\psline(psChartI\i)(psChartO\i)}
\uput[15](psChartO1){A} \rput*(psChartI1){10}
\uput[60](psChartO2){B} \rput*(psChartI2){20}
\uput[130](psChartO3){C} \rput*(psChartI3){15}
\uput[190](psChartO4){D} \rput*(psChartI4){30}
\uput[260](psChartO5){E} \rput*(psChartI5){15}
\uput[-45](psChartO6){F} \rput*(psChartI6){25}
\end{pspicture*}
```



Au passage, on peut voir comment rapprocher le point extérieur en modifiant la variable `chartNode0` qui détermine la distance par rapport au centre ; par défaut cette distance est de 1,5 fois le rayon et on l'a définie ici à 1,25 fois le rayon. On pourrait de même modifier la distance entre le point intérieur et le centre en modifiant la variable `chartNodeI`.

C'est `\rput*` – employé à la place de `\rput` – qui permet d'écrire les nombres sur fond blanc.

## Chronique 17

# Notes de marges

Dans le style de ce qui a été fait dans la chronique 14 (voir page 71), on peut vouloir écrire dans des devoirs le barème que l'on cache (ou pas) aux élèves.  
Mais où va-t-on écrire le barème ? Dans la marge, bien sûr !

### 17.1 Principes généraux

L'instruction qui permet d'écrire dans la marge est `\marginpar` ; on entre

```
\marginpar{\red note}
```

note

pour que le mot « note » s'écrive en rouge dans la marge.

De quel côté ? Bonne question !

Si on écrit en mode recto seulement, le texte sera écrit par défaut dans la marge de droite. Si on est en mode recto-verso (`twoside`), le texte sera écrit dans la marge droite des pages impaires, et dans la marge gauche des pages paires ; c'est normal, c'est toujours le plus loin de la reliure.

La largeur de la zone de marge dans laquelle on peut écrire dépend de plusieurs paramètres, notamment la classe du document ; je me suis rendu compte qu'en mode recto-verso, la largeur de la zone de droite n'était pas la même que la largeur de la zone de gauche. Je conseille donc de fixer la largeur des deux zones (par exemple à 50 points) en entrant dans le préambule :

```
\setlength{\marginparwidth}{50pt}
```

Si on veut modifier la distance entre le bord du texte et ce que l'on va écrire dans la marge, on peut modifier la variable `\marginparsep` :

```
\setlength{\marginparsep}{7pt}
```

Attention : il faut que la somme de `\marginparwidth` et de `\marginparsep` soit inférieure à la taille de la marge du document (définie dans le préambule).

### 17.2 Barème

Pour afficher le barème dans la marge, on va définir une variable `\bareme` qui va écrire le barème si elle est active ; on entre dans le préambule du document : `\def\bareme` et on met un signe % devant la ligne pour désactiver la variable.

Sur chaque ligne où l'on voudra écrire un élément du barème, on entrera :

```
\ifdefined\bareme \marginpar{\red 2 pt} \fi
```

2 pt

On peut trouver que le 2 pt est écrit un peu bas par rapport à la ligne ; on utilise alors `\vspace*` avec un argument négatif pour le remonter :

```
\ifdefined\bareme \marginpar{\vspace*{-7pt} \red 2 pt} \fi
```

## 17.3 Compléments

### 17.3.1 Changement de côté

En mode recto-verso, le texte d'une note de marge est par défaut vers l'extérieur de la feuille; en mode recto, ce texte est toujours situé à droite. On peut changer le côté où l'on veut écrire la note de marge en entrant l'instruction `\reversemarginpar` avant d'appeler l'instruction `\marginpar`.

note 1

On va écrire **note 1** (en rouge) comme note de marge : on entre `\marginpar{\red note 1}`.

La page courante ayant un numéro pair (86), la note est du côté gauche.

On inverse le côté par défaut des notes de marge en utilisant `\reversemarginpar`.

Ainsi en entrant :

```
\reversemarginpar
\marginpar{\red note 2}
```

note 2

la **note 2** s'écrit dans la marge droite.

On remet les marges dans l'ordre normal en entrant : `\normalmarginpar`

Merci à Arnaud Gazagnes pour cette info que je n'ai pas trouvée ailleurs. La version du 10 février 2014 de son excellent « L<sup>A</sup>T<sub>E</sub>X... pour le prof de math » est téléchargeable à l'adresse

<http://math.univ-lyon1.fr/irem/IMG/pdf/LatexPourProfMaths.pdf>

### 17.3.2 Page paire - impaire

Dans le cas d'un mode recto-verso, on peut différencier le texte que l'on écrira en fonction de la parité de la page; si on entre `\marginpar[gauche]{droite}`, le mot « gauche » sera écrit dans la marge gauche de la page si l'instruction est appelée depuis une page paire, et le mot « droite » dans la marge droite si la page est impaire.

### 17.3.3 Boîte

note un  
peu longue

La note de marge est située dans une boîte; donc si le texte dépasse la largeur de la zone d'écriture, il passe automatiquement à la ligne; on peut voir dans la marge gauche le résultat obtenu par l'instruction :

```
\marginpar{\red note un peu longue}
```

Au passage, on constate que le texte est justifié dans la boîte et que la note de marge s'écrit bien du côté gauche, ce qui signifie que l'instruction `\normalmarginpar` a bien fait son travail.

Comme la note de marge est insérée dans une boîte, on peut tourner le texte que l'on va mettre dans la marge en utilisant `\rotatebox` : `\marginpar{\rotatebox{90}{\blue marge}}`

marge

### 17.3.4 Distance entre deux notes

Pour être exhaustif sur le sujet, je signale qu'il existe une variable définissant la distance entre deux notes de marge, c'est `\marginparpush` qui, par défaut, vaut 5 points.

On peut la mettre à 10 points en entrant l'instruction `\setlength{\marginparpush}{10pt}`.

### 17.3.5 En guise de conclusion

On peut naturellement insérer dans la marge un symbole quelconque pour attirer l'attention du lecteur; voici un exemple dans lequel on a créé un triangle contenant un point d'exclamation au moyen de `\newcommand` :

```
\newcommand{\attention}
{\psset{unit=1cm} \marginpar
{\pspolygon[linecolor=red,linewidth=1.5pt](0,0)(1,0)(1;60)
\rput(0.45,0.35){\Large \bf !}}}
```

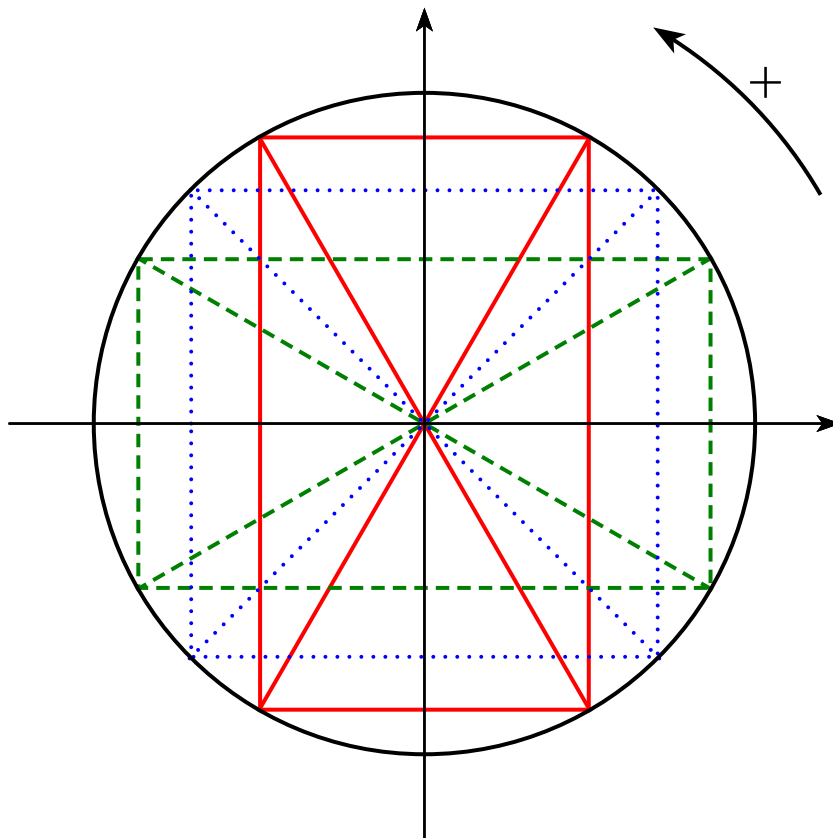
On insère tout simplement cet avertissement dans la marge en entrant `\attention`.



# Cercle trigonométrique

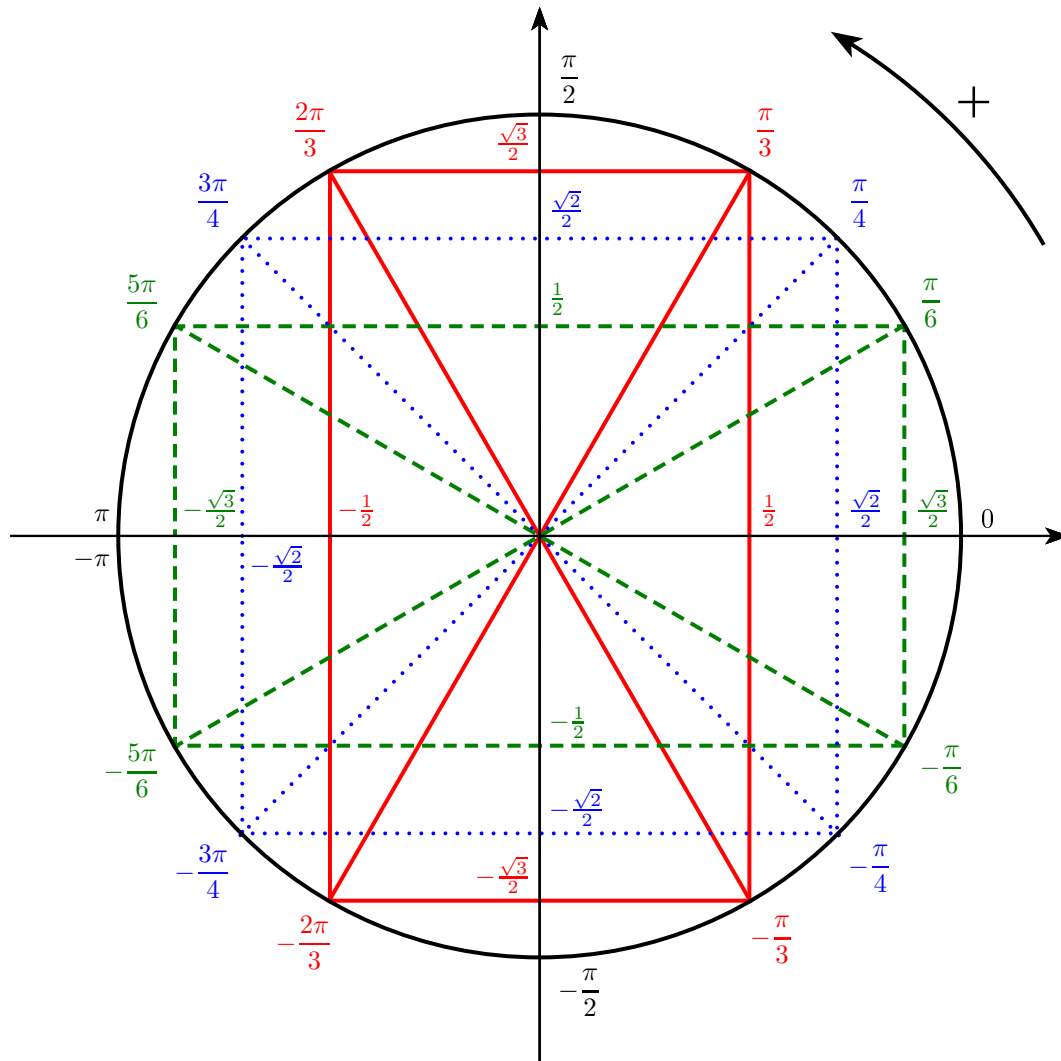
On va le dessiner en deux versions : avec ou sans les valeurs des angles et de leurs images par les fonctions cosinus et sinus.

Voici la version muette :



## 18.2 Cercle complet

Et voici la version complète :



### 18.3 Quelques explications

Il n'y a naturellement qu'un seul code pour ces deux cercles ; la différence vient d'une variable `\corrigé` qui, activée, donne le cercle complet. Quand cette variable n'est pas activée, les informations sont écrites en blanc, donc n'apparaissent pas. Voir chronique 14 de la saison 2.

Il a d'abord fallu définir les couleurs avec des variables : `\cr` pour **c**oefficient de **r**ouge, `\cv` et `\cb` respectivement pour le **v**ert et le **b**leu, que l'on peut modifier à sa guise.

Mais pour définir des couleurs avec des variables, il faut utiliser l'instruction `\definecolor` et pas l'instruction `\newrgbcolor`.

Chaque triplet `\cr,\cv,\cb` définit deux couleurs ; par exemple pour le rectangle correspondant à  $\pi/3$ , ce sont les couleurs `aaaaaa` et `aaaa` ; cette deuxième couleur `aaaa` est soit la même que `aaaaaa`, soit le blanc si la variable `\corrigé` n'est pas activée.

Pour tout savoir sur les définitions de couleurs, reportez-vous à la première chronique de cette saison.



## 18.4 Le code

Le code de l'ensemble est un peu long mais il est assez structuré et commenté.

```
{\def\corrige{}}% accolades indispensables
% début du code du cercle trigonométrique

\psset{unit=1.4cm}
\begin{pspicture}(-5,-5)(5.5,5.5)

\psset{linewidth=1.5pt,arrowsize=4pt 3,arrowinset=0.25}% paramètres

% orientation du cercle
\psarc{->}(0,0){5.5}{30}{60}      \psdots[dotstyle=+,dotscale=2](5.8;45)

{% rectangle correspondant à pi/3

% définition de la couleur
\def\cr{1} \def\cv{0} \def\cb{0}% pour modifier les couleurs
% \cr : coefficient de rouge entre 0 et 1
% \cv : coefficient de vert entre 0 et 1
% \cb : coefficient de bleu entre 0 et 1
\definecolor{aaaaaa}{rgb}{\cr,\cv,\cb}
\ifdefined \corrige
    \definecolor{aaaa}{rgb}{\cr,\cv,\cb}% même couleur que aaaaaa
\else
    \definecolor{aaaa}{rgb}{1,1,1}% ou blanc
\fi

\psset{linecolor=aaaaaa}% coloration des segments

\psframe(4;60)(4;240)% rectangle
\psline(4;60)(4;240)% diagonale
\psline(4;120)(4;-60)% diagonale

\color{aaaa}

% indications sur le cercle
\uput[60](4;60){$\dfrac{\pi}{3}$}      \uput[-60](4;-60){$-\dfrac{\pi}{3}$}
\uput[120](4;120){$\dfrac{2\pi}{3}$}  \uput[-120](4;-120){$-\dfrac{2\pi}{3}$}

% valeurs des cosinus et sinus
\uput[45](2,0){$\frac{1}{2}$}      \uput[45](-2,0){$-\frac{1}{2}$}
\uput[135](0,3.464){$\frac{\sqrt{3}}{2}$}  \uput[135](0,-3.464){$-\frac{\sqrt{3}}{2}$}

}% fin de rectangle correspondant à pi/3

{% rectangle correspondant à pi/6

\def\cr{0} \def\cv{0.5} \def\cb{0}
\definecolor{bbbbbb}{rgb}{\cr,\cv,\cb}
\ifdefined\corrige \definecolor{bbbb}{rgb}{\cr,\cv,\cb}
\else
    \definecolor{bbbb}{rgb}{1,1,1}
\fi

\psset{linestyle=dashed,linecolor=bbbbbb}

\psframe(4;30)(4;210) \psline(4;30)(4;210) \psline(4;-30)(4;150)
```

```

\color{bbbb}

\uput[30](4;30){$\dfrac{\pi}{6}$} \uput[-30](4;-30){$-\dfrac{\pi}{6}$}
\uput[150](4;150){$\dfrac{5\pi}{6}$} \uput[210](4;210){$-\dfrac{5\pi}{6}$}

\uput[45](3.464,0){$\frac{\sqrt{3}}{2}$} \uput[45](0,2){$\frac{1}{2}$}
\uput[45](-3.464,0){$-\frac{\sqrt{3}}{2}$} \uput[45](0,-2){$-\frac{1}{2}$}

}% fin de rectangle correspondant à pi/6

{% carré correspondant à pi/4

\def\cr{0} \def\cv{0} \def\cb{1}
\definecolor{cccccc}{rgb}{\cr,\cv,\cb}
\ifdefined\corrigé \definecolor{cccc}{rgb}{\cr,\cv,\cb}
\else \definecolor{cccc}{rgb}{1,1,1}
\fi

\psset{linestyle=dotted,linecolor=cccccc}

\psframe(4;45)(4;225) \psline(4;45)(4;225) \psline(4;135)(4;-45)

\color{cccc}

\uput[45](4;45){$\dfrac{\pi}{4}$} \uput[-135](4;-135){$-\dfrac{3\pi}{4}$}
\uput[-45](4;-45){$-\dfrac{\pi}{4}$} \uput[135](4;135){$\dfrac{3\pi}{4}$}

\uput[45](0,-2.828){$-\frac{\sqrt{2}}{2}$} \uput[45](0,2.828){$\frac{\sqrt{2}}{2}$}
\uput[-45](-2.828,0){$-\frac{\sqrt{2}}{2}$} \uput[45](2.828,0){$\frac{\sqrt{2}}{2}$}

}% fin de carré correspondant à pi/4

% cercle
\pscircle[linewidth=1.5pt](0,0){4}

\psaxes[linewidth=0.8pt,labels=none,ticks=0]{->}(0,0)(-5,-5)(5,5)

{
\ifdefined\corrigé \newrgbcolor{dddd}{0 0 0}% noir
\else \newrgbcolor{dddd}{1 1 1}% blanc
\fi

\color{dddd}

\uput[45](4,0){0} \uput[45](0,4){$\dfrac{\pi}{2}$}
\uput[-45](0,-4){$-\dfrac{\pi}{2}$} \uput{8pt}[ul](-4,0){$\pi$}
\uput{8pt}[dl](-4,0){$-\pi$}
}

\end{pspicture}
% fin du code du cercle trigo
}% fin de la définition de la variable \corrigé

```

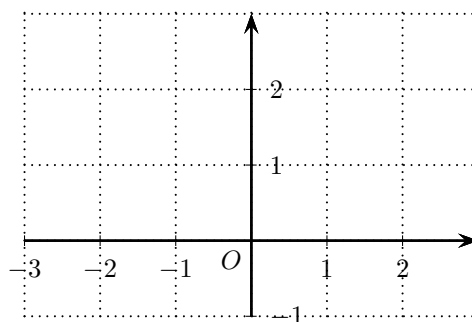
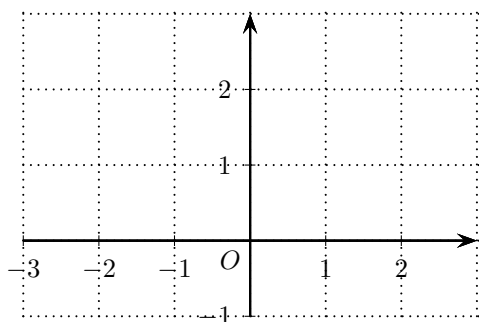
Ouf !

## Chronique 19

# Compléments sur les repères

### 19.1 Gauche ou droite, haut ou bas

Voyez-vous la différence entre ces deux repères ?



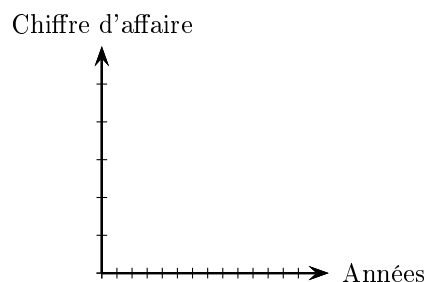
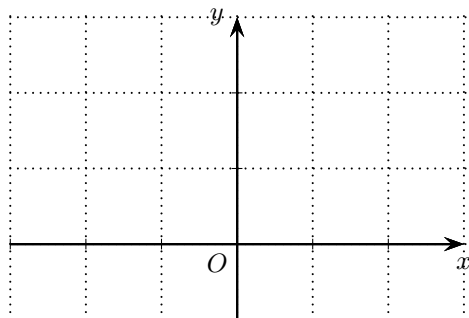
Bien sûr ! Dans celui de droite, les graduations sur l'axe des ordonnées sont à droite de l'axe. Cette position est gérée par la variable `ylabelPos` (attention à la majuscule sur le P) qui, par défaut, est à `left` ; il suffit donc d'entrer dans `\psaxes` l'option `ylabelPos=right` pour écrire les graduations sur la droite.

Vous vous doutez qu'on peut faire pareil en abscisse avec la variable `xlabelPos` qui peut prendre deux valeurs : `bottom` (par défaut) et `up`.

### 19.2 Au bout des axes

#### 19.2.1 Principe

Il peut être intéressant de présenter des graphiques avec des libellés au bout des axes :



On peut, naturellement, aller écrire « à la main » ces textes au moyen de `\uput` ou de `\rput`.

Mais il existe une option de l'instruction `\psaxes` qui permet de faire ça très facilement ; après avoir défini tout ce qu'il faut, on peut rajouter deux options correspondant à des textes qui seront respectivement écrits au bout de l'axe des abscisses et au bout de l'axe des ordonnées.

Dans l'exemple de gauche, on rajoutera comme option `[$x$,d][$y$,l]` ; le `d` signifie `down` et peut être remplacé par `-90` ou tout autre valeur désignant un angle en degrés. Vous avez compris que le caractère après `$y$` est la lettre  $\ell$  qui signifie `left`.

La ligne `\psaxes` ressemble donc à :

```
\psaxes[...]{->}(0,0)(\xmin,\ymin)(\xmax,\ymax)[$x$,d][$y$,l]
```

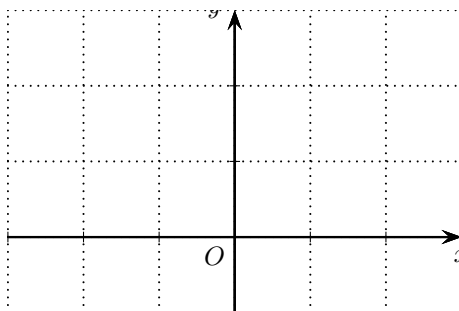
Dans l'exemple de droite on a rajouté en bout de ligne `[Années,r][Chiffre d'affaire,u]`, avec `r` pour `right` et `u` pour `up`.

Si on ne souhaite un libellé que sur l'axe des  $x$ , on entrera quelque chose comme `[$x$,d][{},l]` ; pour un libellé uniquement sur l'axe des  $y$ , ce sera `[{},d][$y$,l]`.

### 19.2.2 Précaution

Il arrive souvent que l'on utilise `\pspicture*` à la place de `\pspicture` pour limiter ce que l'on trace au rectangle défini par `\pspicture*`.

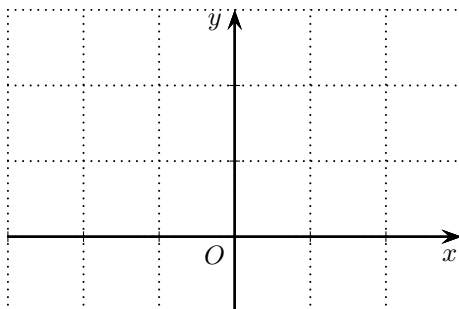
Voilà le résultat de ce que l'on obtient avec les options `[$x$,d][$y$,l]` en utilisant `\pspicture*` :



Les lettres  $x$  et  $y$  sont coupées ; il faut donc les déplacer.

Au lieu de mettre pour  $x$  la valeur `-90` (qui correspond à `down`), on mettra `-120`, et au lieu de mettre pour  $y$  la valeur `180` (qui correspond à `left`), on mettra `210`.

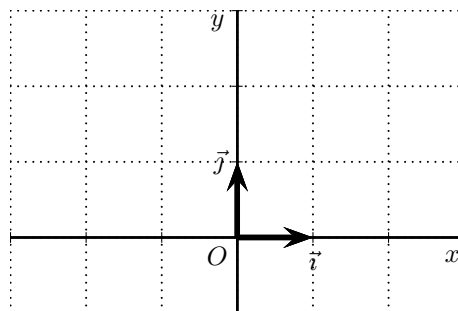
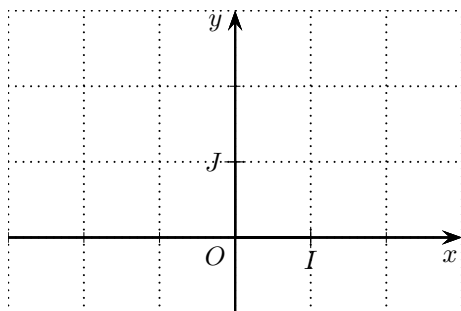
On entrera donc comme options `[$x$,-120][$y$,210]`, ce qui donnera :



Nettement mieux !

### 19.2.3 Prolongements

Ce que l'on a fait pour les axes, on peut le faire pour écrire  $I$  et  $J$  (en seconde) ou  $\vec{i}$  et  $\vec{j}$  (en première et terminale).



Dans le graphique de gauche, j'ai rajouté la ligne :

```
\psaxes[labels=none](0,0)(1,1)[$I$,d][$J$,l]
```

et dans celui de droite :

```
\psaxes[linewidth=2pt]{->}(0,0)(1,1)[$\vec{i}$,d][$\vec{j}$,l]
```

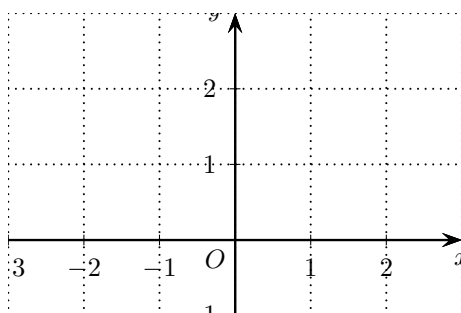
Trois remarques pour terminer ce paragraphe.

- Dans le graphique de gauche, il faut écrire `labels=none` en option pour ne pas que des 0 et des 1 viennent perturber le repère ; c'est inutile dans le graphique de droite à cause des flèches demandées par `{->}`.
- À propos du graphique de droite, quand on écrit les vecteurs unitaires  $\vec{i}$  et  $\vec{j}$ , on ne met plus de flèches au bout des axes.
- Ces options supplémentaires de `\psaxes` ne permettent pas d'écrire  $x'$  au bout gauche de l'axe des abscisses, ni  $y'$  en bas de l'axe des ordonnées.

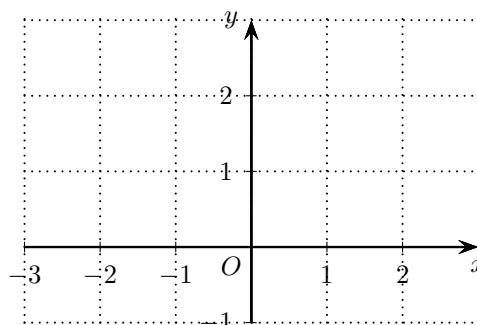
## 19.3 Étoile ou pas

Tout le monde connaît la différence entre `\pspicture` et `\pspicture*` ; ces deux instructions définissent un environnement graphique limité par deux points qui sont les extrémités d'un rectangle. Et dans le cas de la version étoilée, les tracés de tous les objets graphiques définis dans l'environnement seront limités au rectangle.

Évidemment la version étoilée évite d'avoir des représentations graphiques de fonctions qui débordent du rectangle défini par `\pspicture` ; mais si on gradue les axes (en supprimant l'option `labels=none`), la version étoilée pose quelques problèmes d'affichage des graduations :



Avec `\pspicture*`

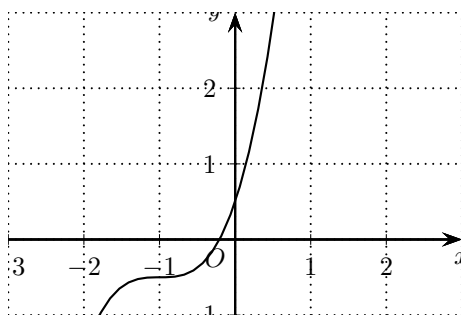
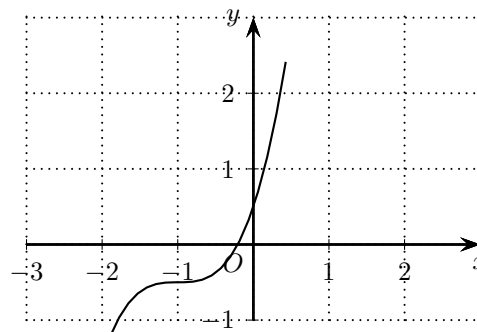


Avec `\pspicture`

L'idéal serait d'avoir un `\pspicture` qui limite les tracés au rectangle défini.

C'est presque possible : on peut définir dans l'environnement deux variables `yMaxValue` et `yMinValue` qui seront respectivement les valeurs maximale et minimale de la représentation graphique d'une fonction tracée avec `\psplot`.

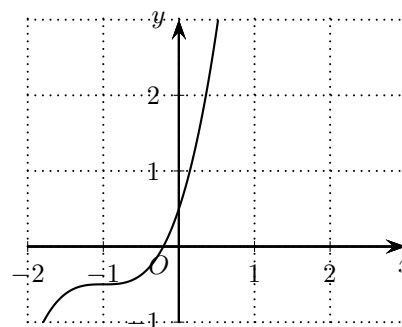
Attention, si on trace un cercle avec un rayon trop grand, son tracé pourra sortir du rectangle défini par `\pspicture`.

Avec `\pspicture*`Avec `\pspicture`

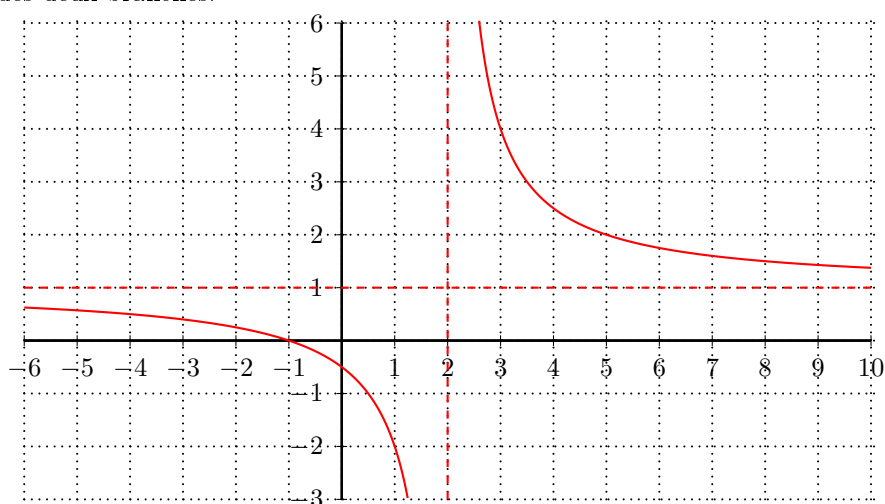
C'est presque bien avec `\pspicture` mais la courbe se trace en dessous de `\ymin` et n'arrive pas tout à fait à `\ymax`. La solution ? Il suffit de définir un nombre de points du tracé assez important en rajoutant comme option `plotpoints=3000` (que l'on peut encore augmenter si nécessaire).

Voici le code et la courbe obtenue :

```
\psset{unit=1cm}
\def\xmin {-2} \def\xmax {3}
\def\ymin {-1} \def\ymax {3}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \psset{yMaxValue=\ymax,yMinValue=\ymin,%
    plotpoints=3000}
  \psset{arrowsize=3pt 3,ticks=-2pt 2pt}
  \psgrid[subgriddiv=1,griddots=10,gridlabels=0]
  \psaxes{->}(0,0)(\xmin,\ymin)(\xmax,\ymax)%
  [$$,d][$y$,1]
  \uput[d1](0,0){$0$}
  \psplot[\xmin]{\xmax}{x 1 add 3 exp 0.5 sub}
\end{pspicture}
```



Autre (énorme) avantage de la méthode : si on veut tracer une hyperbole, on n'a pas besoin de s'occuper des deux branches.



Ce tracé de la fonction est obtenu par `\psplot[linecolor=red]{\xmin}{\xmax}{\f}` où la fonction `\f` est définie par `\def\f{3 x 2 sub div 1 add}`.

Un jour on verra comment tracer la fonction tangente...

## Chronique 20

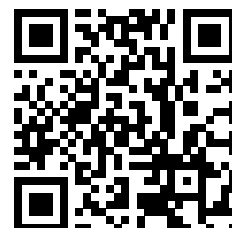
# Courte bibliographie

Il y a une très abondante littérature sur  $\text{\LaTeX}$ ; voici une liste de documents intéressants (il m'a fallu faire une sélection sévère!).

1. Pour écrire en  $\text{\LaTeX}$ , il faut avoir chargé quelques packages. Tous les packages sont documentés (ou presque) sur le site de CTAN. C'est évidemment une source inépuisable de renseignements (pas toujours faciles à interpréter, d'ailleurs!), en anglais le plus souvent.  
Lien : <http://ctan.org/>
2. Un des premiers documents que j'ai eu en ma possession est « Une courte (?) introduction à  $\text{\LaTeX}$  2<sub>ε</sub> » de TOBIAS OETIKER. Ce document est téléchargeable en version 5.03 en anglais ou en version 5.01 en français.  
Version anglaise : <http://tobi.oetiker.ch/lshort/lshort.pdf>  
Version française : <http://www.tex.ac.uk/ctan/info/lshort/french/lshort-fr.pdf>
3. Un excellent document régulièrement remis à jour et corrigé par son auteur ARNAUD GAZAGNES : «  $\text{\LaTeX}$ ... pour le prof de maths! ».   
Lien : <http://math.univ-lyon1.fr/irem/IMG/pdf/LatexPourProfMaths.pdf>
4. Autre très riche document écrit par VINCENT LOZANO et qui s'intitule « Tout ce que vous avez toujours voulu savoir sur  $\text{\LaTeX}$  sans jamais oser le demander ». On peut le télécharger gratuitement mais on peut aussi l'acheter pour la modique somme de 15 €.   
Lien : [http://www.framabook.org/docs/latex/framabook5\\_latex\\_v1\\_art-libre.pdf](http://www.framabook.org/docs/latex/framabook5_latex_v1_art-libre.pdf)  
Lien : <http://www.apmep.asso.fr/IMG/pdf/TraductionfinalePSTricks.pdf>
5. Un petit aide-mémoire que j'ai imprimé et que je consulte très souvent : « Dessin géométrique en  $\text{\LaTeX}$  avec PSTricks ».   
Lien : <http://www.mathforu.com/pdf/dessin-pstricks.pdf>
6. Une liste – exhaustive? – des commandes  $\text{\LaTeX}$  avec renvois hypertextes au sein de la liste est proposée par L'ENS. Très bon outil.   
Lien : [http://www.math.ens.fr/~millien/tdlatex/liste\\_commandes.pdf](http://www.math.ens.fr/~millien/tdlatex/liste_commandes.pdf)
7. Autre document que j'ai pas mal consulté et qui est très bien construit : « How to » par SÉBASTIEN COMBÉFIS.   
Lien : <http://www.latex-howto.be/files/LaTeX-HowTo-full.pdf>
8. L'auteur de Texmaker, PASCAL BRACHET, a mis une bonne documentation sur  $\text{\LaTeX}$  et sur Texmaker sur son site.   
Lien : [http://www.xm1math.net/texmaker/doc\\_fr.html](http://www.xm1math.net/texmaker/doc_fr.html)

9. Et pour retrouver ces chroniques, même sur votre smartphone :

Lien : <http://latexetmath.canalblog.com/>



# À SUIVRE. . .



# Index

## A

<code>\addcontentsline</code> .....	21
affine (ajustement) .....	61
ajustement affine .....	61
amsmath (package) .....	31
angle .....	13, 44
angleA .....	37, 41
angleB .....	37, 41
<code>\Aput</code> .....	36
arbre .....	23
arcangle .....	36

## B

bar (plotstyle) .....	79
barème .....	85
barres (diagramme) .....	79
barwidth .....	79
bibliographie .....	95
<code>\binom</code> .....	77
boîte (diagramme) .....	81
<code>\boldmath</code> .....	31
<code>\boldsymbol</code> .....	31
boundingbox .....	14
<code>\Bput</code> .....	36

## C

caption (package) .....	13
<code>\captionof</code> .....	13
carré .....	67
cases (environnement) .....	31
cercle trigonométrique .....	88
cercle trigonométrique muet .....	87
chartColor ( <code>\psChart</code> ) .....	83
chartNodeI ( <code>\psChart</code> ) .....	84
chartNodeO ( <code>\psChart</code> ) .....	84
chartSep ( <code>\psChart</code> ) .....	83
<code>\circ</code> .....	46
circulaire (diagramme) .....	82
<code>\closepath</code> .....	65
<code>\Cnode</code> .....	40
<code>\cnode</code> .....	40
<code>\Cnode*</code> .....	40

<code>\cnode*</code> .....	40
<code>\cnodeput</code> .....	40
<code>\cnodeput*</code> .....	40
<code>\color</code> .....	4
color (package) .....	3
<code>\colorbox</code> .....	6
combinaison .....	77
compilation (mode de) .....	11
conditionnel (texte) .....	71
<code>\contentsname</code> .....	20
couleurs prédéfinies .....	4

## D

<code>\dbinom</code> .....	77
décagone .....	68
<code>\DecimalMathComma</code> .....	32
<code>\def</code> .....	47, 48
definecolor .....	88
<code>\definecolor</code> .....	4
<code>\dfrac</code> .....	75
diagramme circulaire .....	82
diagramme en barres .....	79
diagramme en boîte .....	81
dimension (multido) .....	53
<code>\displaystyle</code> .....	75
document	
esclave .....	9
maître .....	10
dodécagone .....	68
dotscale .....	58
dvips .....	11

## E

edge .....	47, 48
<code>\else</code> ( <code>\ifdefined</code> ) .....	72
entrée forcée (table des matières) .....	21
eps (format) .....	11
équilatéral (triangle) .....	67
esclave (document) .....	9

## F

<code>\fcolorbox</code> .....	6
-------------------------------	---

<code>\fi</code> ( <code>\ifdefined</code> )	71
<code>fillcolor</code>	79
<code>fillstyle</code>	79
<code>floor</code>	57
flottant (objet)	11
fonctions composées	45
<code>\frac</code>	75
<code>fraction</code>	75

## G

GeoGebra	12
graphe	35
graphe probabiliste	35
<code>graphicx</code> (package)	12
gras (mode mathématique)	31

## H

<code>height</code>	12
hexagone régulier	67
<code>\hskip</code>	36
hyperbole (tracé)	94

## I

<code>\ifdefined</code>	71
image (insertion)	11
<code>\includegraphics</code>	12
insertion d'image	11
inkscape (logiciel)	15
<code>\int</code>	76
integer (multido)	51
intégrale	76

## K

Königsberg (ponts de)	40
-----------------------	----

## L

légende (d'une figure)	13
<code>levelsep</code>	24
<code>\lim</code>	77
limite	77
<code>linearc</code>	44
<code>\lineto</code>	65
<code>\listplot</code>	79

## M

maître (document)	10
marge (note de)	85
<code>\marginpar</code>	85
<code>\marginparpush</code>	86

<code>\marginparsep</code>	85
<code>\marginparwidth</code>	85
<code>\moveto</code>	59, 65
<code>\mput*</code>	38, 44
<code>\multido</code>	65
<code>\multido</code>	33, 51, 58
<code>multido</code> (package)	51

## N

<code>nab</code> (shortcut)	30, 39
<code>name</code>	50
<code>\ncarc</code>	36
<code>\ncbar</code>	44
<code>\nccircle</code>	37
<code>\nccurve</code>	41, 43, 47
<code>\nccurve*</code>	41
<code>\ncline</code>	40, 47
<code>\ncput*</code>	29
<code>\newcmykcolor</code>	7
<code>\newgray</code>	7
<code>\newhsbcolor</code>	7
<code>\newrgbcolor</code>	7
<code>nodesep</code>	24, 36
<code>nœud</code>	35
<code>\nombre</code>	63
<code>\normalmarginpar</code>	86
note de marge	85
<code>\np</code>	63
<code>np</code> (numprint)	82
number (multido)	52
numprint (package)	63, 82

## O

octogone	68
<code>offsetA</code>	45
<code>offsetB</code>	45
<code>origin</code>	13

## P

<code>\pagecolor</code>	4
partie entière	57
pentagone régulier	65
<code>\phantom</code>	49
<code>\pi</code>	63
<code>plotpoints</code>	94
<code>plotstyle</code>	58, 79
<code>\pmb</code>	31
polaire (repérage)	55
polygone régulier	65
<code>\prod</code>	77
produit	77

profondeur (table des matières) .....	18
ps2pdf .....	11
\psarc .....	69
\psaxes (options) .....	92
\psChart .....	82
psChart (\psChart) .....	84
psChartI (\psChart) .....	84
psChartO (\psChart) .....	84
\pscustom .....	59, 65
\psnode .....	38
\pspicture .....	93
\pspicture* .....	93
pst-all (package) .....	79
pst-tree .....	23
\pstree .....	24
pstricks-add (package) .....	6

## R

racine .....	76
radius .....	24, 27
real (multido) .....	53
\RequirePackage .....	79
\reversemarginpar .....	86
\rlineto .....	59
\Rnode .....	35
\rnode .....	35
rosace .....	55, 68
\rput* .....	84

## S

scale .....	13
\setcounter .....	18
shadow .....	80
shortput .....	30, 39
somme .....	77
sommet d'un graphe .....	35
\sqrt .....	76
\StandardMathComma .....	32
structure (éléments) .....	17
\sum .....	77

## T

tab (shortput) .....	30
table des matières .....	17
tablr (shortput) .....	30
\taput .....	27
\tbput .....	27
\TC .....	24
\TC* .....	24
\TCircle .....	27

\TCircle .....	27
\Tdot .....	24
\textcolor .....	4
texte conditionnel .....	71
\Tf .....	24
tilde .....	26
\tlput .....	27
\tnpos .....	26
tocdepth .....	18
\Tp .....	24
\Tr .....	26
treemode .....	24
treeseq .....	24
triangle équilatéral .....	67
trigonométrie (cercle) .....	88
\trput .....	27

## U

userColor (\psChart) .....	84
----------------------------	----

## V

virgule (mode mathématique) .....	32
-----------------------------------	----

## W

width .....	12
-------------	----

## X

xlabelPos .....	91
-----------------	----

## Y

ylabelPos .....	91
yMaxValue .....	93
yMinValue .....	93



# Sommaire

<b>1</b>	<b>Couleurs en L<sup>A</sup>T<sub>E</sub>X</b>	<b>3</b>
1.1	Un peu de théorie . . . . .	3
1.1.1	Synthèse additive . . . . .	3
1.1.2	Synthèse soustractive . . . . .	3
1.1.3	Couleurs primaires . . . . .	3
1.2	Package <code>color</code> . . . . .	3
1.2.1	Couleurs prédéfinies . . . . .	4
1.2.2	Instructions . . . . .	4
1.3	Package <code>pstricks-add</code> . . . . .	6
1.3.1	Couleurs prédéfinies et utilisation . . . . .	7
1.3.2	Définition de nouvelles couleurs . . . . .	7
1.4	Noir et blanc . . . . .	8
<b>2</b>	<b>Maître et esclave</b>	<b>9</b>
2.1	Principe . . . . .	9
2.2	Un document esclave . . . . .	9
2.3	Le document maître . . . . .	10
2.4	La table des matières . . . . .	10
<b>3</b>	<b>Insertion d'image</b>	<b>11</b>
3.1	Mode de compilation . . . . .	11
3.2	Flottant ou pas . . . . .	11
3.3	Format EPS . . . . .	11
3.3.1	Création d'une image EPS avec GeoGebra . . . . .	12
3.3.2	Insertion de l'image . . . . .	12
3.3.3	Options . . . . .	12
3.3.4	Légende . . . . .	13
3.4	Format JPG . . . . .	14
3.4.1	Insertion de l'image . . . . .	14
3.4.2	Conversion de format . . . . .	15
<b>4</b>	<b>Table des matières</b>	<b>17</b>
4.1	Classe <code>book</code> . . . . .	17
4.2	Principe . . . . .	17
4.3	Niveau de profondeur . . . . .	18
4.4	Changements de nom . . . . .	20
4.4.1	Nom de la table des matières . . . . .	20
4.4.2	Nom d'entrée dans la table . . . . .	21
4.5	Entrée forcée . . . . .	21
4.6	Le point sur le i . . . . .	21

<b>5 Arbres</b>	<b>23</b>
5.1 Objectif . . . . .	23
5.2 Premiers pas . . . . .	23
5.3 Arbres plus fournis . . . . .	24
5.4 Les nœuds . . . . .	26
5.5 Autre version . . . . .	26
5.6 Les branches . . . . .	27
5.7 Nouvelle version . . . . .	27
5.8 La touche finale . . . . .	28
5.9 Le code . . . . .	28
5.10 Variante . . . . .	29
5.11 Raccourcis . . . . .	30
<b>6 Petits trucs mathématiques</b>	<b>31</b>
6.1 Gras ou très gras . . . . .	31
6.2 Environnement <code>cases</code> . . . . .	31
6.3 Virgule en mode mathématique . . . . .	32
6.4 Repère . . . . .	33
6.4.1 Premier exemple . . . . .	33
6.4.2 Deuxième exemple . . . . .	34
6.4.3 Le code du repère . . . . .	34
<b>7 Graphes</b>	<b>35</b>
7.1 Le principe . . . . .	35
7.2 Un premier graphe . . . . .	35
7.2.1 Les sommets . . . . .	35
7.2.2 Les arcs . . . . .	36
7.2.3 Les probabilités . . . . .	36
7.2.4 Les boucles . . . . .	37
7.2.5 Le code . . . . .	37
7.2.6 Variante . . . . .	38
7.2.7 Placement des diagrammes . . . . .	38
7.3 Un graphe plus compliqué . . . . .	38
7.4 Raccourcis . . . . .	39
7.5 Autre graphe . . . . .	40
7.6 Les ponts de Königsberg . . . . .	40
7.7 Tout faire avec <code>\nccurve</code> . . . . .	41
<b>8 Applications des graphes</b>	<b>43</b>
8.1 Distributivité . . . . .	43
8.2 Pourcentages . . . . .	44
8.3 Fonctions composées . . . . .	45
8.4 Tableau de variations . . . . .	46
<b>9 Compléments sur les arbres</b>	<b>47</b>
9.1 Un peu de fantaisie . . . . .	47
9.2 Chemin coloré . . . . .	48
9.3 Répartition des branches . . . . .	49
9.4 Nom des nœuds . . . . .	50

<b>10 Multido</b>	<b>51</b>
10.1 Rappel . . . . .	51
10.2 Variable de type <code>integer</code> . . . . .	51
10.3 Variable de type <code>number</code> . . . . .	52
10.4 Variable de type <code>real</code> . . . . .	53
10.5 Variable de type <code>dimension</code> . . . . .	53
10.6 Applications graphiques . . . . .	54
10.7 Rosace . . . . .	55
10.8 Bouquet final . . . . .	56
<b>11 Fonction partie entière</b>	<b>57</b>
11.1 Première version . . . . .	57
11.2 Deuxième version . . . . .	58
11.3 Troisième version . . . . .	58
11.4 Quatrième version . . . . .	59
11.5 Application . . . . .	60
<b>12 Ajustement affine</b>	<b>61</b>
12.1 Le problème . . . . .	61
12.2 Le repère et les axes . . . . .	61
12.3 Le quadrillage et les légendes . . . . .	62
12.4 Les points . . . . .	63
12.5 La droite . . . . .	64
12.6 Le code complet . . . . .	64
<b>13 Polygones et autres dessins</b>	<b>65</b>
13.1 Pentagone . . . . .	65
13.2 Triangle . . . . .	67
13.3 Carré . . . . .	67
13.4 Autres polygones réguliers . . . . .	67
13.4.1 Hexagone . . . . .	67
13.4.2 Octogone . . . . .	68
13.4.3 Décagone . . . . .	68
13.4.4 Dodécagone . . . . .	68
13.5 Rosaces . . . . .	68
<b>14 Document prof-élève</b>	<b>71</b>
14.1 Premier exemple . . . . .	71
14.2 Autre version . . . . .	72
14.3 Deuxième exemple . . . . .	73
<b>15 Displaystyle</b>	<b>75</b>
15.1 Définition . . . . .	75
15.2 Exemples . . . . .	75
15.2.1 Fraction . . . . .	75
15.2.2 Racine . . . . .	76
15.2.3 Intégrale . . . . .	76
15.2.4 Somme et produit . . . . .	77
15.2.5 Limite . . . . .	77
15.2.6 Nombre de combinaisons . . . . .	77
15.3 Raccourcis . . . . .	77

<b>16 Diagrammes</b>	<b>79</b>
16.1 Matriochkas . . . . .	79
16.2 Diagramme à barres . . . . .	79
16.3 Diagramme en boîte . . . . .	81
16.4 Diagramme circulaire . . . . .	82
<b>17 Notes de marges</b>	<b>85</b>
17.1 Principes généraux . . . . .	85
17.2 Barème . . . . .	85
17.3 Compléments . . . . .	86
17.3.1 Changement de côté . . . . .	86
17.3.2 Page paire - impaire . . . . .	86
17.3.3 Boîte . . . . .	86
17.3.4 Distance entre deux notes . . . . .	86
17.3.5 En guise de conclusion . . . . .	86
<b>18 Cercle trigonométrique</b>	<b>87</b>
18.1 Cercle muet . . . . .	87
18.2 Cercle complet . . . . .	88
18.3 Quelques explications . . . . .	88
18.4 Le code . . . . .	89
<b>19 Compléments sur les repères</b>	<b>91</b>
19.1 Gauche ou droite, haut ou bas . . . . .	91
19.2 Au bout des axes . . . . .	91
19.2.1 Principe . . . . .	91
19.2.2 Précaution . . . . .	92
19.2.3 Prolongements . . . . .	92
19.3 Étoile ou pas . . . . .	93
<b>20 Courte bibliographie</b>	<b>95</b>
<b>Index</b>	<b>97</b>
<b>Sommaire</b>	<b>101</b>