

Mathématiques

et

L<sup>A</sup>T<sub>E</sub>X

Chroniques de François HACHE – 2013

<http://latexetmath.canalblog.com>

*J'espère que la postérité me jugera vraiment, non seulement sur les choses que j'ai expliquées, mais aussi sur celles que j'ai intentionnellement omises pour laisser aux autres le plaisir de les découvrir.*

*René Descartes*

# 1 – Quelques principes de base

## 1 Généralités

C'EST LE MATHÉMATICIEN DONALD KNUTH qui, exaspéré par les éditeurs qui publiaient ses textes mathématiques avec un rendu plus que médiocre, s'est lancé dans le projet  $\text{T}_{\text{E}}\text{X}$  et a créé ce « traitement de textes mathématiques ». Le  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  a été développé à partir du  $\text{T}_{\text{E}}\text{X}$  par LESLIE LAMPORT.

Un document  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  (on prononce  $\text{T}_{\text{E}}\text{X}$  comme *teck*) n'est en fait qu'un texte que l'on peut écrire avec n'importe quel éditeur de textes ; on l'enregistre avec l'extension `.tex`. Ce texte est ensuite **compilé** par un logiciel spécialisé qui le transforme en fichier *DVI* – *DeVice Independant* – puis le plus souvent en un fichier *PS* – *PostScript* – pour impression. Enfin, on peut le transformer en *PDF* – *Portable Document Format* – pour le rendre lisible et imprimable par tout le monde. On peut aussi compiler directement le texte en *PDF*.

Lorsqu'on a compilé un document appelé *Essai*, plusieurs fichiers sont créés : le fichier texte `Essai.tex` mais aussi les fichiers `Essai.aux`, `Essai.dvi`, `Essai.log` et `Essai.ps`. Si le document contient une table des matières ou un index, les fichiers `Essai.toc` ou `Essai.idx` sont ajoutés ; une liste des tableaux crée le fichier `Essai.lot` et une liste des figures, le fichier `Essai.lof`. Moralité : pour s'y retrouver, il vaut mieux placer chaque nouveau fichier dans un répertoire séparé.

Le système que j'ai utilisé sur PC est  $\text{USB}_{\text{TE}}\text{X}$  ; lors de son implantation, ce système installait tous les logiciels utiles et les chemins (*path*) nécessaires au bon fonctionnement de l'ensemble.  $\text{USB}_{\text{TE}}\text{X}$  a, en fait, été conçu pour fonctionner sur une clé USB, mais il fonctionne très bien installé à demeure sur un ordinateur.

Hélas, en ce début de 2013, on ne peut plus télécharger  $\text{USB}_{\text{TE}}\text{X}$ .

Comme  $\text{USB}_{\text{TE}}\text{X}$  installait *TexMaker*, je travaille avec ce très bon éditeur créé par Pascal Brachet (merci à lui !); je vous conseille de vous rendre sur son site pour l'installer, puis d'installer les autres composants, dont *MikTeX*, en suivant les instructions données à l'adresse :

<http://www.xm1math.net/doculatex/index.html>

## 2 Notion d'environnement

La notion essentielle en  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  est la notion d'environnement ; un environnement commence par un `\begin{...}` et se termine par `\end{...}` :

$$\left[ \begin{array}{l} \backslash\text{begin}\{\text{environnement}\} \\ \dots \\ \backslash\text{end}\{\text{environnement}\} \end{array} \right.$$

On peut imbriquer deux environnements à condition de terminer le second avant le premier :

$$\left[ \begin{array}{l} \backslash\text{begin}\{\text{environnement 1}\} \\ \dots \\ \left[ \begin{array}{l} \backslash\text{begin}\{\text{environnement 2}\} \\ \dots \\ \backslash\text{end}\{\text{environnement 2}\} \end{array} \right. \\ \dots \\ \backslash\text{end}\{\text{environnement 1}\} \end{array} \right.$$

Quelques environnements disponibles :

- les justifications de texte (`flushleft`; `center`; `flushright`);
- les listes (`itemize`; `enumerate`; `description`; `list`);
- les tabulations et tableaux (`tabbing`; `tabular`);
- les citations (`quote`; `quotation`; `verse`);
- le multicolonnage (`multicols`);
- les modes mathématiques (`math`; `displaymath`);
- plus tous les environnements que l'utilisateur peut créer lui-même.

### 3 Structure d'un document

Un document  $\text{\LaTeX}$  doit débiter par la déclaration de la classe de document que l'on va écrire :

```
\documentclass[options]{Type de document}
```

Beaucoup d'instructions en  $\text{\LaTeX}$  ont besoin de paramètres : ces paramètres seront écrits entre deux accolades `{}` et `}` (entre accolades car les parenthèses ont une signification particulière en mathématiques). Dans l'instruction `\documentclass`, c'est obligatoire de donner le `Type de document` que l'on veut créer.

Le `Type de document` peut être :

- `article` pour écrire un texte de quelques pages;
- `report` pour écrire un rapport un peu plus long qu'un article;
- `letter` pour une lettre;
- `book` pour un livre complet;
- `slides` pour créer des transparents.

Le mode `article` suffit pour écrire des textes mathématiques de quelques pages.

En  $\text{\LaTeX}$ , tout ce qui est optionnel s'écrit entre crochets `[ ]`; on peut donc ne rien mettre entre les crochets pour commencer. On verra néanmoins des options obligatoires qui doivent être mises entre crochets.

Un texte s'écrit entre deux balises, l'une de début de document, l'autre de fin de document : `\begin{document}` et `\end{document}`.

Un premier texte écrit en  $\text{\LaTeX}$  peut ressembler à :

```
\documentclass{article}
\begin{document}
Voici mon premier texte.
\end{document}
```

En général, on donne comme option la police de caractères et la taille du papier; on écrira donc :

```
\documentclass[10pt, a4paper]{article}
\begin{document}
Voici mon premier texte.
\end{document}
```

Mais si l'on entre :

```
\documentclass[10pt, a4paper]{article}
\begin{document}
Voici un texte sans intérêt!
\end{document}
```

on verra écrit : Voici un texte sans intrt!

Les accents ont disparu car  $\text{\LaTeX}$  ne connaît pas les caractères accentués ; il faut donc charger des extensions (*packages* en anglais) qui vont dire à  $\text{\LaTeX}$  de reconnaître les caractères accentués. On écrira donc :

```
\documentclass[10pt, a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[français]{babel}
\begin{document}
Voici un texte sans intérêt!
\end{document}
```

Un texte en  $\text{\LaTeX}$  a donc une structure de la forme :

<i>Définition de classe</i>		<code>\documentclass[...]{...}</code>
<i>Chargement des extensions</i>		[ <code>\usepackage[...]{...}</code> <code>\usepackage[...]{...}</code> ...
<i>Définitions personnelles</i>		[ <code>\newcommand{...}{...}</code> <code>\renewcommand{...}{...}</code> ...
<i>Corps du texte</i>		[ <code>\begin{document}</code> ... ... <code>\end{document}</code>

## 4 Conseil pour démarrer

Après avoir installé un système  $\text{\LaTeX}$  sur votre ordinateur, je vous conseille d'essayer de récupérer un fichier `tex` d'un utilisateur averti (ou plus averti que vous), de le copier et de le compiler sur votre ordinateur.

Il faut, lors de cette première compilation, que votre ordinateur soit connecté à Internet ; en effet, s'il manque des extensions, le système (bien installé) ira automatiquement les chercher sur le net et les installera sur votre ordinateur.

Cela veut dire aussi qu'une première compilation peut être longue.

Ensuite, quand on a un système stable (c'est-à-dire avec les extensions nécessaires), il n'est plus utile d'être connecté pour travailler en  $\text{\LaTeX}$ .

Bonnes rédactions en  $\text{\LaTeX}$ , et n'hésitez pas à partager les « beaux » textes mathématiques que vous produirez.



## 2 – Coordonnées

Il y a plusieurs façons d'écrire les coordonnées d'un point (ou d'un vecteur) ; les deux plus courantes sont  $A(3, -2)$  et  $A(3; -2)$ . Les anglo-saxons qui utilisent le point comme séparateur décimal se contentent de la virgule entre l'abscisse et l'ordonnée ; nous sommes obligés d'utiliser le point-virgule pour séparer des nombres qui contiennent déjà une virgule :  $B(2,5;3)$ .

Mais il peut être pratique pour plus de lisibilité, et pour faciliter les calculs entre les abscisses et les ordonnées (voire les cotes), d'écrire les coordonnées « en colonnes » :  $A\begin{pmatrix} 3 \\ -2 \end{pmatrix}$ .

Certains préféreront un trait vertical :  $A\left| \begin{array}{c} 3 \\ -2 \end{array} \right.$ .

Il existe en L<sup>A</sup>T<sub>E</sub>X différents environnements qui permettent de faire ce que l'on veut.

### 1 Première méthode avec `pmatrix`

Une écriture comme  $A\begin{pmatrix} 3 \\ -2 \end{pmatrix}$  s'obtient en utilisant l'environnement `pmatrix` :

```
$A
\begin{pmatrix}
3 \\
-2
\end{pmatrix}$
```

On peut retenir que le `p` de `pmatrix` signifie « parenthèses ».

L'environnement `pmatrix` n'est valide qu'en mode mathématique, d'où la présence du `$` avant d'appeler l'environnement, et la présence du même `$` quand on en sort.

On crée en fait une matrice d'une colonne et de deux lignes, le passage d'une ligne à une autre se faisant par `\\`.

Pas de problème dans un repère de l'espace où l'on aura  $A\begin{pmatrix} 3 \\ -2 \\ 7 \end{pmatrix}$  en rajoutant une troisième ligne.

### 2 Deuxième méthode avec `array`

Une écriture comme  $A\left| \begin{array}{c} 3 \\ -2 \end{array} \right.$  s'obtient en utilisant l'environnement `array`.

Cet environnement permet d'écrire des tableaux en mode mathématique.

Si on veut écrire du texte présenté sous forme de tableau, on utilisera l'environnement `tabular`.

Voici ce qu'il faut entrer :

```
$A
\begin{array}{|c}
3 \\
-2
\end{array}$
```

Le `c` en paramètre signifie que les nombres seront centrés ; on peut les aligner à gauche (en entrant `l` comme `left` à la place du `c`) ou à droite (avec `r` comme `right`).

Le `|` devant le `c` permet de tracer le trait vertical à gauche : si on veut également un trait à droite, on entrera `{|c|}`.

En fait, si on entre exactement ce qui est au dessus, on obtient  $A \begin{vmatrix} 3 \\ -2 \end{vmatrix}$  ce qui n'est pas très beau!

Il faut donc séparer le nom du point (ou du vecteur) et le trait vertical par une espace (en matière d'imprimerie, le mot « espace » est du genre féminin).

Il existe plusieurs espacements possibles en mode mathématique; en voici trois, du plus petit au plus grand : `\`, `\;`, `\l` (attention : le `\l` signifie « appui sur la barre d'espacement » !)

Donc en tapant	<pre>\$A\; \begin{array}{ c} 3 \\ -2 \end{array}\$</pre>	on obtient	$A \begin{vmatrix} 3 \\ -2 \end{vmatrix}$
----------------	--	------------	---

### 3 Autres environnements

Il existe d'autres environnements matriciels qui peuvent être intéressants en d'autres circonstances :

En tapant	<pre>\$\$\begin{matrix} 3 \\ -2 \end{matrix}\$\$</pre>	on obtient	$\begin{matrix} 3 \\ -2 \end{matrix}$
-----------	--	------------	---------------------------------------

En tapant	<pre>\$\$\begin{bmatrix} 3 \\ -2 \end{bmatrix}\$\$</pre>	on obtient	$\begin{bmatrix} 3 \\ -2 \end{bmatrix}$
-----------	--	------------	---

En tapant	<pre>\$\$\begin{vmatrix} 3 \\ -2 \end{vmatrix}\$\$</pre>	on obtient	$\begin{vmatrix} 3 \\ -2 \end{vmatrix}$
-----------	--	------------	---

En tapant	<pre>\$\$\begin{Vmatrix} 3 \\ -2 \end{Vmatrix}\$\$</pre>	on obtient	$\begin{Vmatrix} 3 \\ -2 \end{Vmatrix}$
-----------	--	------------	---

Par exemple pour écrire	$\begin{vmatrix} 3 & -5 & 9 \\ -2 & 8 & 7 \\ 1 & -6 & 11 \end{vmatrix}$	on entrera	<pre>\$\$\begin{vmatrix} 3 &amp; -5 &amp; 9 \\ -2 &amp; 8 &amp; 7 \\ 1 &amp; -6 &amp; 11 \end{vmatrix}\$\$</pre>
-------------------------	---	------------	--

Au passage, on voit que `&` est le séparateur de colonnes dans un tableau ou une matrice.

### 4 Automatisation

C'est bien joli tout ça, mais s'il faut taper des choses du style

```
$A
\begin{pmatrix}
3 \\
-2
\end{pmatrix}$
```



chaque fois qu'on veut écrire les coordonnées d'un point, on n'a pas fini (et on regrette de s'être mis à  $\text{\LaTeX}$ !!!).

Mais non!

Justement  $\text{\LaTeX}$  permet de créer des commandes qui automatisent ce que l'on écrit souvent.

On va donc créer une commande que l'on va appeler `coo`, qui va nécessiter deux paramètres (l'abscisse et l'ordonnée) et qui va automatiquement écrire les coordonnées comme on le souhaite. Il faut donc écrire dans le préambule (avant le `\begin{document}`) :

```
\newcommand{\coo}[2]
{
\begin{pmatrix}
#1 \\
#2
\end{pmatrix}
}
```

On a ainsi défini une nouvelle commande `coo` à deux paramètres (c'est ce que veut dire `[2]`), qui s'appellent `#1` et `#2`; on affectera ces paramètres en mettant les valeurs souhaitées entre accolades. On utilise cette commande en écrivant par exemple `\$A\coo{3}{-2}\$`, ce qui donne le résultat espéré  $A \begin{pmatrix} 3 \\ -2 \end{pmatrix}$ .

Pour l'espace, on va définir une nouvelle commande (à trois paramètres) `cooe` ainsi :

```
\newcommand{\cooe}[3]
{
\begin{pmatrix}
#1 \\
#2 \\
#3
\end{pmatrix}
}
```

qui permettra d'obtenir  $A \begin{pmatrix} 3 \\ -2 \\ 7 \end{pmatrix}$  en tapant `\$A\cooe{3}{-2}{7}\$`.

La définition d'une commande peut se faire sur une seule ligne

```
\newcommand{\cooe}[3]{\begin{pmatrix} #1 \\ #2 \\ #3 \end{pmatrix} }
```

mais c'est nettement moins lisible.

On peut créer des commandes sans paramètre, comme la commande `\R` pour écrire rapidement  $\mathbb{R}$  :

```
\newcommand{\R}{\mathbb R}
```

Personnellement, dans mes textes, j'écris **R** (en gras) pour l'ensemble des réels, réservant la double barre à l'écriture manuscrite; ma commande `\R` est donc :

```
\newcommand{\R}{\mathbf R}
```

Enfin il faut savoir que, dans une nouvelle commande, on ne peut pas passer plus de 9 paramètres, mais on travaille rarement dans un espace de dimension supérieure à 9...



## 3 – Matrices

### 1 Sans fioritures

Une façon d'écrire des matrices avec des parenthèses, est d'utiliser l'environnement `pmatrix`; on change de colonne avec `&`, et de ligne avec `\\`, le tout en mode mathématique :

```


$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$


```

L'environnement `pmatrix` n'a pas besoin, contrairement aux environnements `tabular` et `array`, que l'on précise au départ le nombre de colonnes que l'on va utiliser.

Le résultat obtenu est tout à fait correct.

Mais si on écrit des nombres sous forme de fractions, et qu'on ne fait rien de particulier, le résultat n'est pas terrible !

```


$$\begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{2}{1} & \frac{3}{1} & \frac{4}{1} \\ \frac{5}{1} & \frac{6}{1} & \frac{7}{1} \\ \frac{1}{8} & \frac{1}{9} & \frac{1}{2} \end{pmatrix}$$


```

Voici trois méthodes qui vont arranger les choses...

### 2 Première méthode – Simple et efficace

La hauteur des lignes dans un tableau (`tabular`, `array`, `pmatrix`...) est contrôlée par une commande `\arraystretch` qui gère un facteur d'espacement égal à 1 par défaut; on peut le redéfinir avant de déclarer un tableau :

```
\renewcommand{\arraystretch}{2}
```

Pour ne pas modifier les tableaux qui apparaîtraient dans la suite du document, on remet ce facteur à 1 après avoir quitté l'environnement `pmatrix` :

```
\renewcommand{\arraystretch}{1}
```

```


$$\begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{2}{1} & \frac{3}{1} & \frac{4}{1} \\ \frac{5}{1} & \frac{6}{1} & \frac{7}{1} \\ \frac{1}{8} & \frac{1}{9} & \frac{1}{2} \end{pmatrix}$$


```

Ce facteur n'est pas forcément un nombre entier; on peut tout à fait entrer :

```
\renewcommand{\arraystretch}{1.5}
```



## 4 – Courbes en PsTricks

On peut, en  $\text{\LaTeX}$ , tracer des courbes au moyen d'extensions qu'il suffit de charger dans le préambule du document.

J'utilise depuis mes débuts l'extension `PsTricks` qui permet de faire de belles choses ; on entre donc dans le préambule :

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

### 1 Quelques préliminaires

L'extension `pstricks` qui maintenant s'appelle `pstricks-add` (on lui a rajouté des commandes), permet de définir un environnement `pspicture` qui délimite une zone dans laquelle on va pouvoir :

- tracer des axes avec `\psaxes` ;
- tracer un quadrillage avec `\psgrid` ;
- placer des points avec `\psdots` et écrire leurs noms avec `\uput` ou `\rput` ;
- tracer des segments avec `\psline` ou des vecteurs en rajoutant une option à cette commande ;
- tracer des cercles avec `\pscircle` ou des arcs de cercle avec `\psarc` ;
- tracer des courbes avec `\psplot` ou avec `\pscurve` ;
- etc.

La liste est longue.

Mais on n'est pas obligé d'être dans un environnement `pspicture` pour utiliser des commandes `PsTricks`.

Si, par exemple, on veut écrire « Tourner la feuille » et dessiner une belle flèche en bas de la première page d'un devoir, comme ici :

Tourner la feuille  $\longrightarrow$

il suffit d'entrer :

```
\begin{flushright}
Tourner la feuille \hspace{2cm}\psline[arrowsize=3pt 3]{->}(-2,0.1)(0,0.1)
\end{flushright}
```

L'environnement `flushright` sert à aligner sur la droite la suite de caractères « Tourner la page » suivie d'un espacement horizontal de 2 cm, défini par `\hspace{2cm}`.

`\psline` est la commande qui sert à tracer un trait, et on adjoint `{->}` pour tracer un vecteur ; ce vecteur partira du point de coordonnées  $(-2, 0.1)$  et s'arrêtera au point de coordonnées  $(0, 0.1)$ .

Le plupart du temps, les options facultatives sont entrées entre crochets : `[arrowsize=3pt 3]` permet d'agrandir un peu la taille des flèches. Si on voulait changer la couleur de la flèche pour la dessiner en rouge, on entrerait `linecolor=red` entre les crochets, à la suite de `arrowsize` ; et il faudrait mettre une virgule comme séparateur : `[arrowsize=3pt 3, linecolor=red]`.

Quand il n'y a pas de repère défini par l'environnement `pspicture`, l'emplacement où l'on écrit la commande est de coordonnées  $(0, 0)$ .

L'unité par défaut étant le cm, cette flèche mesurera 2 cm, ce qui explique le `\hspace{2cm}` qui réserve la place nécessaire à droite du texte pour tracer la flèche.

Quant à l'ordonnée 0.1, c'est juste pour remonter un peu la flèche qui serait trop basse sinon (et ce serait moins joli!).

Enfin si on veut être sûr que le « Tourner la feuille » soit le plus bas possible dans la page, on fera précéder les instructions par un `\vfill` qui poussera la phrase le plus bas possible, et on les fera suivre d'un `\newpage` pour passer à la page suivante :

```
\vfill
\begin{flushright}
Tourner la feuille \hspace{2cm}\psline[arrowsize=3pt 3]{->}(-2,0.1)(0,0.1)
\end{flushright}
\newpage
```

## 2 Création du repère

Que faut-il pour tracer une courbe ? D'abord un repère, c'est-à-dire :

- une origine  $O$  ;
- deux droites sécantes en  $O$ , qui seront perpendiculaires dans le cas d'un repère orthogonal. Ces axes sont dessinés avec des flèches au bout dans le cas d'un repère  $(O, I, J)$ , sans flèche dans le cas de repère  $(O; \vec{i}, \vec{j})$  ou  $(O; \vec{u}, \vec{v})$  ;
- des points  $I$  et  $J$ , ou des vecteurs  $\vec{i}$  et  $\vec{j}$ , ou  $\vec{u}$  et  $\vec{v}$ , pour les unités. En plus du dessin des points ou des vecteurs, il faut écrire leurs noms ;
- éventuellement un quadrillage, voire du papier millimétré.

On détaille les différentes instructions nécessaires.

`\psset{xunit=1cm, yunit=1cm, runit=1cm}`  
définit les unités du repère que l'on va créer : c'est assez explicite pour `xunit` et `yunit` ; quant à `runit`, c'est l'unité pour tracer des cercles.

Si toutes les unités sont égales à 1 cm, on peut écrire `unit=1cm`.

Il faut définir les unités avant de définir la zone graphique.

`\begin{pspicture}(-3,-5)(4,6)`

définit une zone graphique rectangulaire qui va du point de coordonnées  $(-3, -5)$  au point de coordonnées  $(4, 6)$  ; ça correspond à des abscisses entre  $-3$  et  $4$ , et des ordonnées entre  $-5$  et  $6$ .

On terminera cet environnement en entrant `\end{pspicture}`.

`\psgrid[subgriddiv=2, gridlabels=0, gridcolor=gray, subgridcolor=orange]`

trace un quadrillage dont la couleur sera grise (`gridcolor=gray`) ; ce quadrillage sera divisé en deux (`subgriddiv=2`). Pour du papier millimétré, on prendra des unités à 1 cm et on entrera `subgriddiv=10`. La couleur de cette subdivision est gérée par `subgridcolor`, orange ici.

Enfin pour ne pas que les nombres de  $-3$  à  $4$  sur l'axe des abscisses, et de  $-5$  à  $6$  sur l'axe des ordonnées, s'affichent, on entre `gridlabels=0`.

Quand je trace une grille, je la trace toujours avant les axes pour que le dessin des axes se fasse par dessus celui de la grille (et non le contraire).

`\psaxes[arrowsize=3pt 3, ticksize=-2pt 2pt, labels=none]{->}(0,0)(-3,-5)(4,6)`

`\psaxes[ticksize=-2pt 2pt, labels=none](0,0)(-3,-5)(4,6)`

permettent de tracer des axes, avec ou sans flèches (présence de `{->}` ou non).

Le  $(0,0)$  indique à quel endroit les axes doivent se couper, et on remet les coordonnées du rectangle défini par `pspicture`. Il m'est arrivé, pour des raisons esthétiques, de « tricher » un peu et de mettre des axes un peu plus longs que le rectangle définie au départ. À voir au cas par cas.

L'égalité `ticksize=-2pt 2pt` définit la dimension du petit trait qui marque les unités entières : 2 points en dessous de la ligne, 2 points au dessus.

Et `labels=none` empêche la numérotation sur les axes ; on le retirera si on veut que cette numérotation soit présente.

À l'intersection des axes, se trouve l'origine  $O$ . Il suffit donc d'écrire le nom du point là où il faut. Pour écrire dans un environnement `pspicture`, j'utilise `\uput` qui nécessite trois paramètres : entre crochets la position par rapport au point, entre parenthèses les coordonnées du point d'ancrage, entre accolades ce que l'on veut écrire. Pour le point  $O$ , on écrira `\uput[d1](0,0){$0$}`.

Le `d1` veut dire `down left` ou en français, `en bas à gauche`. Les positions de base sont `u` pour `up`, `d` pour `down`, `l` pour `left` et `r` pour `right`, que l'on peut combiner en `dl`, `ur`, etc.

Pour un réglage plus fin, on peut entrer la valeur d'un angle en degrés, comme sur un cercle trigonométrique : le `ur` correspond à  $45^\circ$  et le `u` correspond à  $90^\circ$  ; si on veut placer le nom d'un point entre les deux, on peut donc entrer `[60]` comme paramètre de position.

Pour terminer la création du repère, on écrit les noms des points  $I$  et  $J$ , des vecteurs  $\vec{i}$  et  $\vec{j}$ , ou  $\vec{u}$  et  $\vec{v}$  selon le cas :

<code>\uput[d](1,0){\$I\$}</code>	<code>\uput[l](0,1){\$J\$}</code>
<code>\uput[d](0.5,0){\$\vec{\imath}\$}</code>	<code>\uput[l](0,0.5){\$\vec{\jmath}\$}</code>
<code>\uput[d](0.5,0){\$\vec{u}\$}</code>	<code>\uput[l](0,0.5){\$\vec{v}\$}</code>

Quand on met une flèche sur une lettre `i`, c'est plus joli de ne pas mettre le point sur le `i` ; on utilise donc `\imath` et `\jmath` pour écrire  $\vec{i}$  et  $\vec{j}$ .

Naturellement, si on a des repères  $(O; \vec{i}, \vec{j})$  ou  $(O; \vec{u}, \vec{v})$ , il faut tracer les vecteurs ; on le fera avec l'instruction `\psaxes` de la façon suivante : `\psaxes[linewidth=1.8pt]{->}(0,0)(1,1)`

L'instruction `linewidth=1.8pt` va donner une épaisseur de 1,8 point aux dessins des deux vecteurs.

Une remarque à propos de l'écriture des noms des points ; j'ai choisi de les écrire en majuscule (c'est normal) et en italique (ça a peut-être moins). C'est un choix ! On peut les écrire en romain...

Voici donc ce qu'il faut entrer pour un repère  $(O; \vec{i}, \vec{j})$ , où  $x \in [-3, 4]$  et  $y \in [-5, 6]$ , et avec quadrillage au demi-centimètre :

```
\psset{xunit=1cm, yunit=1cm, runit=1cm}
\begin{pspicture}(-3,-5)(4,6)
\psgrid[subgriddiv=2, gridlabels=0, gridcolor=gray, subgridcolor=orange]
\psaxes[ticks=-2pt 2pt, labels=none](0,0)(-3,-5)(4,6)
\uput[d1](0,0){$0$}
\psaxes[linewidth=1.8pt]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$}
\uput[l](0,0.5){$\vec{\jmath}$}
\end{pspicture}
```

### 3 Petite amélioration

Si on modifie le rectangle de départ défini par `pspicture`, il faudra également le modifier dans `\psaxes`, et même plus tard modifier les valeurs de  $x$  dans le tracé de la fonction par `\psplot`.

On peut donc dès la création du repère entrer les valeurs extrêmes de  $x$  et de  $y$  dans des variables :

```
\psset{xunit=1cm, yunit=1cm, runit=1cm}
\def\xmin{-3} \def\xmax{4}
\def\ymin{-5} \def\ymax{6}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=2, gridlabels=0, gridcolor=gray, subgridcolor=orange]
\psaxes[ticks=-2pt 2pt, labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[d1](0,0){$0$}
\psaxes[linewidth=1.8pt]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$}
\uput[l](0,0.5){$\vec{\jmath}$}
\end{pspicture}
```

## 4 Le tracé de courbes

Quand tout ceci est mis en place (une bonne fois pour toutes), on va représenter des fonctions.

Par exemple on va tracer la représentation graphique de la fonction  $f : x \mapsto 2x^2 - 5x - 1$ , de la droite  $d$  d'équation  $y = -2x + 1$  et on va placer leurs points d'intersection  $A$  et  $B$ .

L'écriture  $f : x \mapsto 2x^2 - 5x - 1$  s'obtient en entrant `$f: x \longmapsto 2x^2-5x-1$`.

Ceux qui préfèrent  $x \xrightarrow{f} 2x^2 - 5x - 1$  entreront `$x \stackrel{f}{\longmapsto} 2x^2-5x-1$`.

L'instruction qui permet de tracer la représentation graphique d'une fonction est `\psplot` qui nécessite trois paramètres : la plus petite valeur de  $x$  (donc `\xmin`), la plus grande valeur de  $x$  (`\xmax`) et l'expression de la fonction.

Par défaut, il faut entrer l'expression de la fonction en mode postfixé, comme on faisait dans le temps sur les calculatrices HP. On écrit donc `-2 x mul 1 add` pour  $-2x+1$ , et si on écrit  $2x^2-5x-1$  sous la forme  $2x(x-2,5)-1$ , on entrera `2 x x 2.5 sub mul mul 1 sub`.

Il suffit de connaître `mul`, `div`, `add` et `sub` respectivement pour la multiplication, la division, l'addition et la soustraction ; on rajoute `exp` pour la puissance et `sqrt` pour la racine carrée et on peut ainsi définir la plupart des fonctions dont on a besoin en lycée.

Ceux que cette méthode effraie peuvent entrer les fonctions sous forme algébrique en entrant `algebraic=true` comme paramètre dans `\psset` (voir page 18).

Si on veut lisser un peu plus la courbe, on peut augmenter le nombre de points à tracer par l'option `[plotpoints=1000]`.

Pour tracer la parabole, on entrera donc :

```
\psplot[plotpoints=1000]{\xmin}{\xmax}{2 x x 2.5 sub mul mul 1 sub}
```

et pour la droite :

```
\psplot[plotpoints=1000]{\xmin}{\xmax}{-2 x mul 1 add}
```

ce qui donne finalement comme code :

```
\psset{xunit=1cm, yunit=1cm, runit=1cm}
\def\xmin {-3} \def\xmax {4}
\def\ymin {-5} \def\ymax {6}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=2, gridlabels=0, gridcolor=gray, subgridcolor=orange]
\psaxes[ticks=-2pt 2pt, labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[d1](0,0){$0$}
\psaxes[linewidth=1.8pt]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$} \uput[l](0,0.5){$\vec{\jmath}$}
\psplot[plotpoints=1000]{\xmin}{\xmax}{2 x x 2.5 sub mul mul 1 sub}
\psplot[plotpoints=1000]{\xmin}{\xmax}{-2 x mul 1 add}
\end{pspicture}
```

Mais si on entre exactement ce code, c'est... la catastrophe!

En effet, comme  $f(x) = 2x^2 - 5x - 1$ ,  $f(\text{xmin}) = f(-3) = 32$  dépasse largement le `\ymax` c'est-à-dire 6. Donc la parabole sort du rectangle défini par `pspicture`, et la droite aussi!

Une solution ? Bien sûr ! Il suffit d'utiliser l'environnement `pspicture*` qui va limiter tous les tracés au rectangle `(\xmin,\ymin)(\xmax,\ymax)`.

Enfin, comme on a défini par des variables les valeurs extrêmes de  $x$  et de  $y$ , on peut donner des noms aux fonctions que l'on utilise, surtout si on les utilise plusieurs fois, par exemple dans le cas d'un tracé d'hyperbole (page ??) ou pour tracer une aire sous une courbe.

On définit donc la fonction  $f$  par `\def\f{2 x x 2.5 sub mul mul 1 sub}`,  
la fonction affine par `\def\g{-2 x mul 1 add}`.



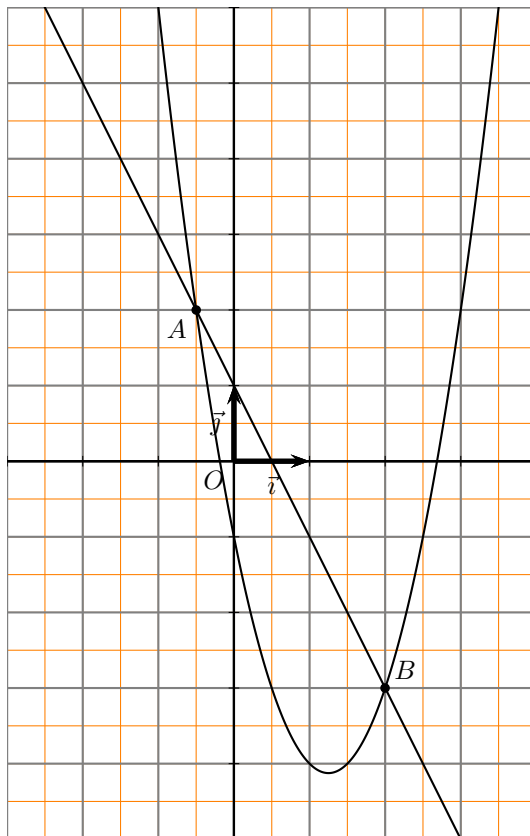
Il ne reste que les points d'intersection  $A(-0,5;2)$  et  $B(2;-3)$  à placer en utilisant l'instruction `\psdots` suivie des coordonnées des points : `\psdots(-0.5,2)(2,-3)`.

On écrit leurs noms avec `\uput[d1](-0.5,2){$A$}` pour  $A$  et `\uput[ur](2,-3){$B$}` pour  $B$ .

On entre donc le code suivant :

```
\psset{xunit=1cm, yunit=1cm, runit=1cm}
\def\xmin {-3}
\def\xmax {4}
\def\ymin {-5}
\def\ymax {6}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=2, gridlabels=0, gridcolor=gray, subgridcolor=orange]
\psaxes[ticks=-2pt 2pt, labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[d1](0,0){$0$}
\psaxes[linewidth=1.8pt]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$} \uput[l](0,0.5){$\vec{\jmath}$}
\def\ff{2 x x 2.5 sub mul mul 1 sub}
\def\gg{-2 x mul 1 add}
\psplot[plotpoints=1000]{\xmin}{\xmax}{\ff}
\psplot[plotpoints=1000]{\xmin}{\xmax}{\gg}
\psdots(-0.5,2)(2,-3)
\uput[d1](-0.5,2){$A$}
\uput[ur](2,-3){$B$}
\end{pspicture*}
```

Ce qui donne :



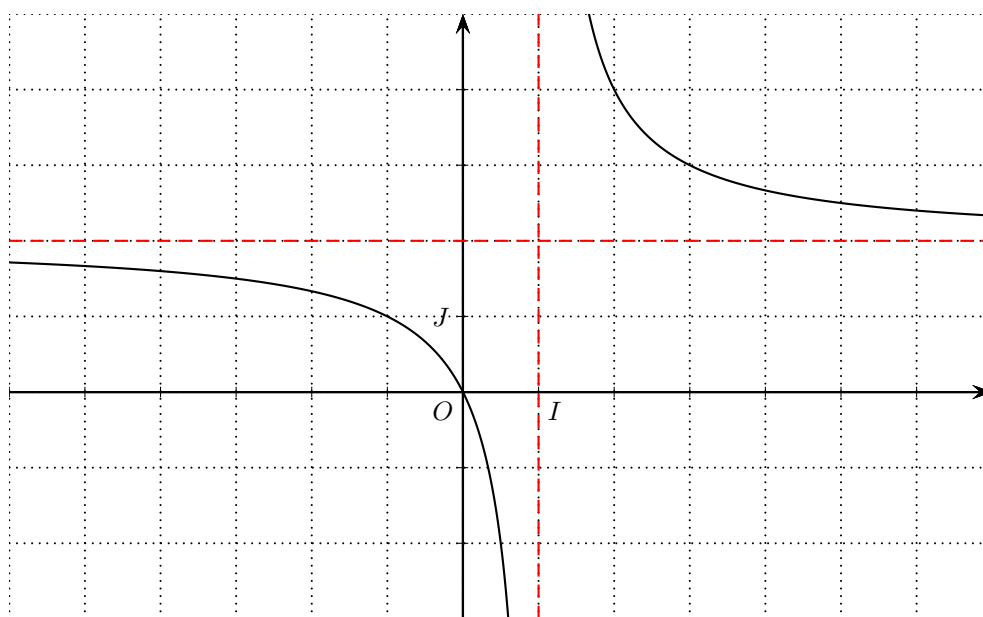
## 5 Autre exemple

Si on représente une fonction rationnelle, on peut avoir un problème d'ensemble de définition. Voici un exemple de tracé d'une hyperbole représentant la fonction  $f$  définie sur  $\mathbf{R} \setminus \{1\}$  par  $f(x) = \frac{2x}{x-1}$ , et de ses deux asymptotes.

```
\psset{xunit=1cm, yunit=1cm, runit=1cm, algebraic=true}
\def\xmin {-6} \def\xmax {7}
\def\ymin {-3} \def\ymax {5}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=1, griddots=10, gridlabels=0, gridcolor=black]
\psaxes[arrowsize=3pt 3, ticksize=-2pt 2pt, labels=none]{->}(0,0)
(\xmin,\ymin)(\xmax,\ymax) % à écrire à la suite de la ligne précédente
\uput[dl](0,0){$0$}
%\psaxes[linewidth=1.8pt]{->}(0,0)(1,1)
%\uput[d](0.5,0){$\vec{\imath}$} \uput[l](0,0.5){$\vec{\jmath}$}
\uput[dr](1,0){$I$} \uput[l](0,1){$J$}
\def\ff{2*x/(x-1)} % définition de la fonction
\psplot[plotpoints=1000]{\xmin}{0.99}{\ff} % une branche de l'hyperbole
\psplot[plotpoints=1000]{1.01}{\xmax}{\ff} % l'autre branche
\psline[linestyle=dashed, linecolor=red](\xmin,2)(\xmax,2) % y=2
\psline[linestyle=dashed, linecolor=red](1,\ymin)(1,\ymax) % x=1
\end{pspicture*}
```

Au passage, on peut voir les tracés des deux branches de l'hyperbole, le `griddots=10` qui trace la grille en pointillés avec 10 points par division (donc par centimètre), et le `linestyle=dashed` qui trace les asymptotes en mode « tirets » ; pour des pointillés, il faudrait écrire `linestyle=dotted`. L'expression `algebraic=true` dans `\psset` permet de ne pas utiliser la notation postfixée pour entrer la fonction.

Enfin tout ce qui se trouve après un signe `%` est un commentaire qui n'est pas pris en compte.



À vous de jouer !

## 5 – Multicolonnage et minipage

Même quand on écrit des textes mathématiques, on peut être conduit à écrire sur plusieurs colonnes, par exemple si on veut mettre côte à côte un texte et une figure.

### 1 Multicolonnage

#### 1.1 Le principe

Pour écrire sur plusieurs colonnes, la méthode la plus simple consiste à utiliser l’environnement `multicols` (avec un `s`) qui a besoin d’un paramètre : le nombre de colonnes que l’on veut. Cet environnement nécessite l’extension `multicol` (sans `s`) que l’on charge par un `\usepackage{multicol}`.

Pour écrire du texte sur deux colonnes, on entrera :

```
\begin{multicols}{2}
...
\end{multicols}
```

Exemple de texte sur deux colonnes Exemple de texte sur deux colonnes Exemple de texte sur deux colonnes Exemple de texte sur deux colonnes Exemple de texte sur deux colonnes Exemple de texte sur deux colonnes

Le texte est réparti automatiquement sur les deux colonnes.

#### 1.2 Quelques améliorations

On peut agrandir l’espace entre les colonnes géré par la variable `\columnsep`, ou tracer une ligne séparatrice entre les colonnes au moyen de la variable `\columnseprule`.

```
\setlength{\columnsep}{1.5cm}
\setlength{\columnseprule}{0.5pt}
\begin{multicols}{2}
...
\end{multicols}
\setlength{\columnsep}{1em}
\setlength{\columnseprule}{0pt}
```

Texte sur deux colonnes avec espacement de 1,5 cm et ligne séparatrice de largeur 0,5 point Texte sur deux colonnes avec espacement de 1,5 cm et ligne séparatrice de lar-

geur 0,5 point Texte sur deux colonnes avec espacement de 1,5 cm et ligne séparatrice de largeur 0,5 point

Les deux dernières lignes servent à remettre les variables dans leur état initial, le `em` étant l’unité de largeur de base, le cadratin.

Enfin on peut forcer le passage d’une colonne à l’autre avec `\columnbreak` qu’il faut faire précéder et faire suivre par une ligne vide :

```
\begin{multicols}{2}
texte de la colonne de gauche

\columnbreak

texte de la colonne de droite
\end{multicols}
```

Texte avec changement de colonne forcé  
avec changement de colonne forcé

Texte avec changement de colonne forcé  
avec changement de colonne forcé  
Texte avec changement de colonne forcé  
avec changement de colonne forcé

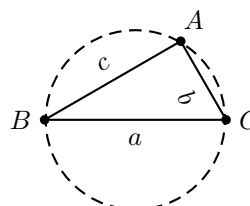
### 1.3 Un exemple

Voici un exemple d'un texte mathématique placé à côté d'une figure :

Le triangle  $ABC$  est inscrit dans le cercle de diamètre  $[BC]$ .

On peut en déduire que le triangle  $ABC$  est rectangle  $A$ .

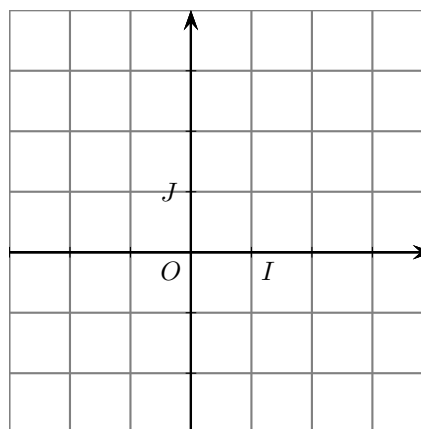
D'après le théorème de Pythagore, on peut écrire que :  $BC^2 = AC^2 + AB^2$ .  
Si on pose  $AB = c$ ,  $AC = b$  et  $BC = a$ , alors  $BC^2 = AC^2 + AB^2 \Leftrightarrow a^2 = b^2 + c^2$ .



### 1.4 Autre exemple

Dans l'exemple ci-dessous, on a rentré sans précaution particulière un court texte que l'on veut placer à côté d'un repère sur deux colonnes :

Dans le repère ci-contre, tracer les représentations graphiques des fonctions affines  $f$  et  $g$  définies par  $f(x) = 2x - 1$  et  $g(x) = -x + 3$ .



Calculer les coordonnées du point  $A$  d'intersection des deux droites tracées.

L'espacement entre les deux phrases de la colonne de gauche n'est pas voulu : c'est  $\LaTeX$  qui a essayé de répartir « au mieux » le texte sur toute la hauteur de la colonne.  
Si on veut éviter cela, il faut forcer la remontée de la deuxième phrase en remplissant la colonne par des espaces verticaux au moyen de la commande `\vfill`; on entrera :

```
\begin{multicols}{2}
```

Dans le repère ci-contre, tracer les représentations graphiques des fonctions affines  $f$  et  $g$  définies par  $f(x) = 2x - 1$  et  $g(x) = -x + 3$ .

Calculer les coordonnées du point  $A$  d'intersection des deux droites tracées.

```
\vfill
```

```

\begin{center}
\psset{unit=0.8cm, arrowsize=3pt 3}
\def\xmin {-3} \def\xmax {4}
\def\ymin {-3} \def\ymax {4}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=0, gridlabels=0, gridcolor=gray]
\psaxes[ticks=-2pt 2pt, labels=none]{->}(0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[d1](0,0){$0$}
\uput[dr](1,0){$I$} \uput[l](0,1){$J$}
\end{pspicture*}
\end{center}

\end{multicols}

```

À essayer avec le `\vfill` et sans.

## 1.5 Limite du multicolonnage

Malheureusement avec `multicols` les colonnes ont toutes la même largeur, et ce n'est pas forcément ce que l'on veut : on pourrait par exemple vouloir réduire la largeur de la colonne contenant la figure, et donc élargir l'autre.

Il y a, bien sûr, une solution !

## 2 Minipage

Pour obtenir des colonnes de largeurs différentes, on va utiliser la `minipage` en en créant deux côte à côte ; l'intérêt de la `minipage`, c'est qu'on peut en définir la largeur.

Il faut pour cela avoir chargé l'extension `graphicx` en tapant dans le préambule `usepackage{graphicx}`.

On utilise souvent la variable `\linewidth` qui donne la largeur d'une ligne et on définit la largeur de la minipage en fonction de `\linewidth`.

Voyons le mode de fonctionnement :

```

\begin{minipage}{0.7\linewidth}
...
\end{minipage}
\begin{minipage}{0.3\linewidth}
...
\end{minipage}

```

On aura ainsi deux colonnes, l'une de largeur égale à  $7/10^e$  de la largeur de la page, l'autre de largeur égale à  $3/10^e$  de cette largeur.

Attention : il ne faut pas laisser de ligne vide entre le `\end{minipage}` qui termine la première `minipage`, et le `\begin{minipage}{0.3\linewidth}` qui démarre la deuxième, sinon les minipages seront placées l'une au dessus de l'autre, ce qui n'est pas l'effet recherché.

Naturellement, on peut créer plus de deux minipages l'une à côté de l'autre ; il faut quand même que la somme des largeurs des minipages ne dépasse pas la largeur de la page (sauf si on veut dépasser dans la marge droite).

Voici donc ce donne l'exemple du paragraphe 1.3 en remplaçant le `multicols` par des minipages :

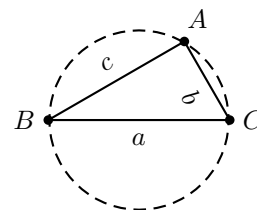
Le triangle  $ABC$  est inscrit dans le cercle de diamètre  $[BC]$ .

On peut en déduire que le triangle  $ABC$  est rectangle  $A$ .

D'après le théorème de Pythagore, on peut écrire que :

$$BC^2 = AC^2 + AB^2.$$

Si on pose  $AB = c$ ,  $AC = b$  et  $BC = a$ , alors  $BC^2 = AC^2 + AB^2 \Leftrightarrow a^2 = b^2 + c^2$ .



Le code à rentrer pour obtenir l'exemple précédent est :

```
\begin{minipage}{0.7\linewidth}
Le triangle  $ABC$  est inscrit dans le cercle de diamètre  $[BC]$ .\\

On peut en déduire que le triangle  $ABC$  est rectangle  $A$ .\\

D'après le théorème de Pythagore, on peut écrire que:\\
 $BC^2=AC^2+AB^2$ .\\
Si on pose  $AB=c$ ,  $AC=b$  et  $BC=a$ , alors  $BC^2=AC^2+AB^2$  \ssi  $a^2=b^2+c^2$ .

\end{minipage}
\begin{minipage}{0.3\linewidth}
\begin{center}
\psset{unit=0.6cm}
\def\xmin {-3} \def\xmax {3}
\def\ymin {-2} \def\ymax {3}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pscircle[linestyle=dashed](0,0){2}
\pspolygon[showpoints=true](-2,0)(2,0)(1,1.732)
\uput[60](1,1.732){ $A$ } \uput[-90](0,0){ $a$ }
\uput[180](-2,0){ $B$ } \uput[-130]{-60}(1.5,0.866){ $b$ }
\uput[0](2,0){ $C$ } \uput[120]{30}(-0.5,0.866){ $c$ }
\end{pspicture}
\end{center}
\end{minipage}
```

On peut faire d'autres choses avec les minipages, par exemple les encadrer ou n'en encadrer qu'une :

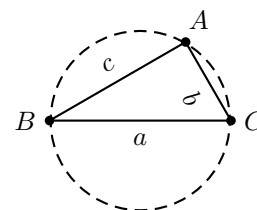
Le triangle  $ABC$  est inscrit dans le cercle de diamètre  $[BC]$ .

On peut en déduire que le triangle  $ABC$  est rectangle  $A$ .

D'après le théorème de Pythagore, on peut écrire que :

$$BC^2 = AC^2 + AB^2.$$

Si on pose  $AB = c$ ,  $AC = b$  et  $BC = a$ , alors  $BC^2 = AC^2 + AB^2 \Leftrightarrow a^2 = b^2 + c^2$ .



Mais on en parlera dans une autre rubrique...

## 6 – Solides de l'espace

On a déjà vu comment tracer des courbes avec `PsTricks`, ou construire des petites figures de géométrie. On va encore utiliser `PsTricks` pour dessiner des solides dans l'espace.

Et comme on utilise `PsTricks`, on ajoute `\usepackage{pstricks-add}` dans le préambule.

### 1 Cube

Voici les instructions qui permettent de réaliser un cube :

```
\psset{xunit=0.8cm, yunit=0.8cm}
\def\xmin{0} \def\xmax{10} \def\ymin{0} \def\ymax{10}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pspolygon[showpoints=true](1,7)(7,7)(9,9)(3,9)
\psline[showpoints=true](1,7)(1,1)(7,1)(9,3)(9,9)
\psline(7,1)(7,7)
\psline[linestyle=dashed](1,1)(3,3)(3,9)
\psline[linestyle=dashed, showpoints=true](3,3)(9,3)
\uput[d1](1,1){$A$} \uput[dr](7,1){$B$}
\uput[r](9,3){$C$} \uput[ul](3,3){$D$}
\uput[ul](1,7){$E$} \uput[100](7,7){$F$}
\uput[ur](9,9){$G$} \uput[ul](3,9){$H$}
\end{pspicture}
```

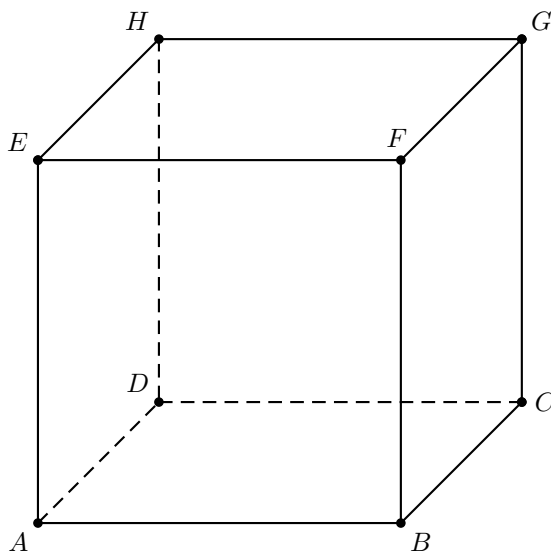
L'instruction `\pspolygon`, comme son nom l'indique, trace un polygone fermé.

L'option `[showpoints=true]` dessine les points aux sommets du polygone.

Par défaut, les points sont représentés par des petits disques dont on peut modifier la taille; si on veut changer l'aspect des points, on peut entrer dans `\psset` l'instruction `dotstyle=+` pour avoir une petite croix, ou `dotstyle=x` pour avoir une autre croix comme symbole du point.

L'option `[showpoints=true]` peut être rajoutée si on trace un segment avec `\psline`, ou une courbe avec `\pscurve`. Pour ne pas tracer les points, il suffit de ne pas entrer cette option.

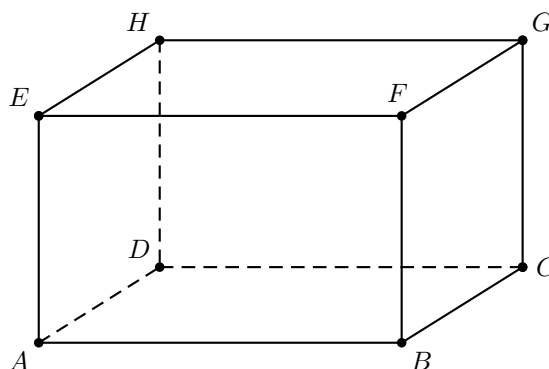
Cela donne un cube assez convenable :



## 2 Parallélépipède

Il faut changer peu de choses au cube pour en faire un parallélépipède rectangle : on réduit l'unité en  $x$  ou en  $y$  et le tour est joué.

En entrant `\psset{xunit=0.8cm, yunit=0.5cm}` comme première ligne d'instruction à la place de celle qui s'y trouve, on obtient un magnifique parallélépipède rectangle :



Et si on veut un parallélépipède non rectangle, c'est possible? Bien sûr!

On va utiliser l'instruction `\pstilt` qui a besoin de deux paramètres : le premier représente l'angle (en degrés) par rapport à l'horizontale (donc si on met 90, il ne se passe rien), et en second paramètre ce que l'on veut « tilter ».

En tapant `\pstilt{60}{LaTeX}` on obtient *LaTeX* (angle de 60°).

En tapant `\pstilt{90}{LaTeX}` on obtient LaTeX (aucun changement!).

En tapant `\pstilt{120}{LaTeX}` on obtient *LaTeX* (angle de 120°).

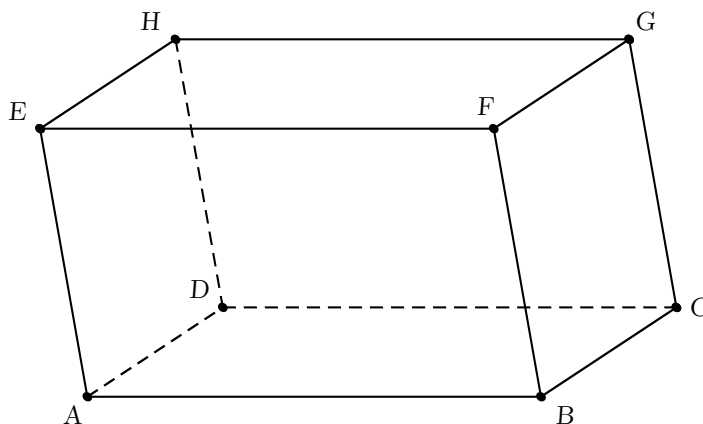
L'instruction `\pstilt` fait partie de l'extension `pst-3d`, qu'il est inutile de charger car elle est incluse dans l'extension `pstricks-add` (alors qu'elle ne l'était pas dans l'extension `pstricks`).

En entrant :

```
\psset{xunit=1cm, yunit=0.6cm}
\def\xmin{0} \def\xmax{10}
\def\ymin{0} \def\ymax{10}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
  \pstilt{100}
  {
    % début de ce que l'on veut incliner
    \pspolygon[showpoints=true](1,7)(7,7)(9,9)(3,9)
    \psline[showpoints=true](1,7)(1,1)(7,1)(9,3)(9,9)
    \psline(7,1)(7,7)
    \psline[linestyle=dashed](1,1)(3,3)(3,9)
    \psline[linestyle=dashed, showpoints=true](3,3)(9,3)
    \uput[d1](1,1){$A$} \uput[dr](7,1){$B$}
    \uput[r](9,3){$C$} \uput[ul](3,3){$D$}
    \uput[ul](1,7){$E$} \uput[100](7,7){$F$}
    \uput[ur](9,9){$G$} \uput[ul](3,9){$H$}
  }
  % fin du \pstilt
\end{pspicture}
```



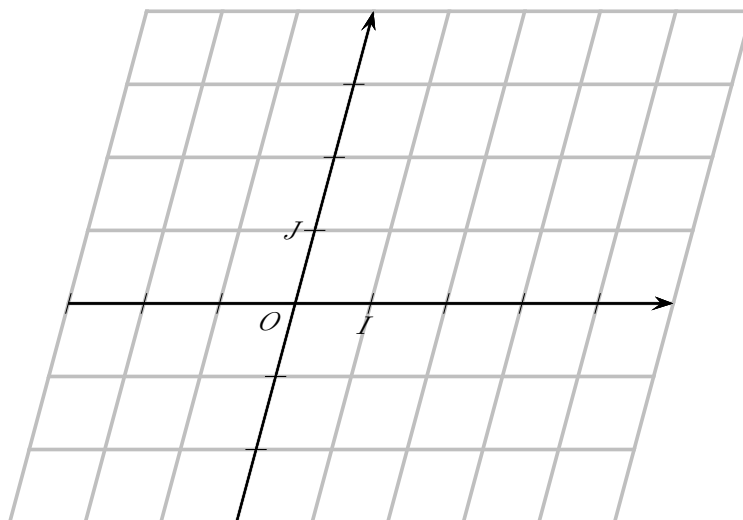
on obtient :



Attention : tout ce qui est entre les deuxièmes accolades du `\pstilt` sera incliné, y compris les noms des points ; il ne faut donc pas donner un angle trop important, sinon l'ensemble devient assez disgracieux !

### 3 Repère oblique

Autre application du `\pstilt` : construire un repère oblique.

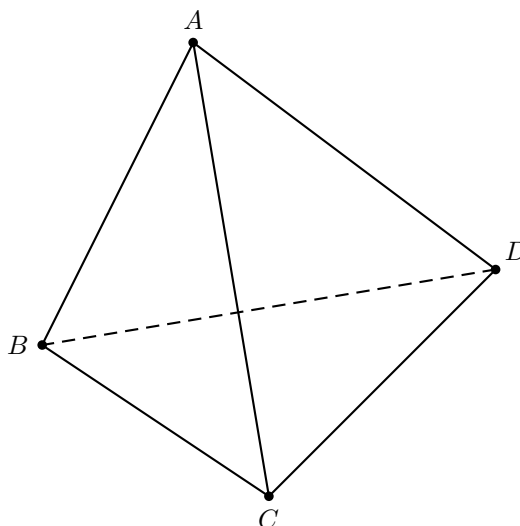


qui s'obtient en entrant :

```
\psset{xunit=1cm,yunit=1cm}
\def\xmin{-3} \def\xmax{5} \def\ymin{-3} \def\ymax{4}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pstilt{75}{
\psgrid[gridlabels=0,gridwidth=0.5mm,subgriddiv=0,gridcolor=lightgray]
\psaxes[labels=none,arrowsize=3pt 3]{->}(0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[d](1,0){$I$} \uput[1](0,1){$J$}
\uput[-135](0,0){$O$}
}
\end{pspicture}
```

## 4 Tétraèdre

Enfin, comme il n'y a pas que le cube dans l'espace, voici un tétraèdre :



et son code :

```
\psset{xunit=1cm, yunit=1cm}
\def\xmin{0} \def\xmax{8} \def\ymin{0} \def\ymax{8}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\psline(3,7)(1,3)(4,1)(3,7)(7,4)(4,1)
\psline[linestyle=dashed](7,4)(1,3)
\point{3}{7}{u}{A$}
\point{7}{4}{ur}{D$}
\point{1}{3}{l}{B$}
\point{4}{1}{d}{C$}
\end{pspicture}
```

À cette occasion, on peut voir une commande personnelle appelée `\point`, qui place un point et écrit son nom en même temps ; elle nécessite quatre paramètres : l'abscisse du point, son ordonnée, la position du nom et le nom lui-même. On entre les paramètres entre des accolades.

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

On peut comprendre aisément le fonctionnement de cette commande qui ne fait qu'appliquer les deux commandes `\psdots` et `\uput`.

Elle est à copier dans le préambule du document.

J'ai l'habitude de commenter mes commandes, c'est-à-dire d'expliquer ce qu'elles font ; je vous conseille de faire la même chose car on oublie vite, surtout si on ne les utilise pas tous les jours !

## 7 – Compléments sur les minipages

On a vu dans la chronique 5 comment créer des minipages ; on va voir un peu plus d'options sur les minipages dans celle-ci.

### 1 Largeur

En L<sup>A</sup>T<sub>E</sub>X,  $0,7 + 0,3$  n'est pas toujours égal à 1.

En effet, si on met côte à côte deux minipages de largeurs  $0.7\text{\textwidth}$  et  $0.3\text{\textwidth}$ , on constate que le texte dépasse un peu dans la marge droite.

Pour bien voir ce qui se passe, j'ai rajouté deux lignes horizontales ayant la largeur de la page, au moyen de `\hrulefill` :

```
\hrulefill

\begin{minipage}{0.7\textwidth}
...
\end{minipage}
\begin{minipage}{0.3\textwidth}
...
\end{minipage}

\hrulefill
```

---

essai essai essai essai essai es-  
 essai  
 essai  
 essai essai essai essai essai essai  
 essai essai essai essai essai essai

---

Ce sera encore plus flagrant quand les minipages seront encadrées.

La solution ? Je n'en ai pas trouvé, donc je triche : je mets  $0.7\text{\textwidth}$  pour la première minipage, et  $0.29\text{\textwidth}$  au lieu de  $0.3\text{\textwidth}$  pour la seconde ; le résultat est alors correct. Il faudra tricher un peu plus si on met côte à côte trois ou quatre minipages.

---

essai essai essai essai essai es-  
 essai  
 essai  
 essai essai essai essai essai essai  
 essai essai essai essai essai essai

---

On peut remarquer au passage que le texte est justifié à l'intérieur de chaque minipage, et que les mots sont coupés correctement en fin de ligne si nécessaire.

### 2 Positionnement sur la ligne

On peut régler le positionnement d'une minipage par rapport à la ligne sur laquelle on écrit.

Il suffit de rentrer un paramètre optionnel (donc entre crochets) : `[t]` pour `top`, `[c]` pour `center`, ou `[b]` pour `bottom`.

Avec `\begin{minipage}[t]{3cm}` on obtient :

```
essai essai essai es-
sai essai essai essai
essai essai
```

Avec `\begin{minipage}[c]{3cm}` on obtient :

```
essai essai essai es-
sai essai essai essai
essai essai
essai essai essai es-
sai essai essai essai
```

Et avec `\begin{minipage}[b]{3cm}` on obtient : essai essai

Le paramètre par défaut est `[c]`.

### 3 Encadrement

En  $\text{\LaTeX}$  il y a plusieurs façons d'encadrer du texte ou des images.

Une façon simple d'encadrer du texte que j'utilise souvent est `\fbox` :

- `\fbox{Mathématiques}` donne : Mathématiques
- `\fbox{L'ensemble solution est  $S=\left\{2\right\}; 5\right\}}$`  donne :

L'ensemble solution est  $S = \{2; 5\}$

On peut encadrer du texte mathématique délimité par des `$`, mais on ne peut pas utiliser `\fbox` entre deux `$` (il y a d'autres instructions d'encadrement pour le mode mathématique).

Pour encadrer une minipage, il suffit de faire précéder le début de la minipage par `\fbox{` et de ne pas oublier de rentrer une accolade fermante `}` après la minipage.

En entrant :

```
\fbox{
\begin{minipage}[t]{3cm}
essai essai essai essai essai
essai essai essai essai
\end{minipage}
}
```

on obtient :

```
essai essai essai es-
sai essai essai essai
essai essai
```

On voit qu'on peut combiner la minipage et le multicolonnage.

### 4 Hauteur

On peut aussi régler la hauteur d'une minipage : il suffit de rajouter un paramètre supplémentaire. Pour qu'on voie bien ce qui se passe, j'ai encadré les minipages.

Avec `\begin{minipage}[t][3cm]{4cm}` on obtient :

```
Hauteur de 3 cm, largeur
de 4 cm Hauteur de 3 cm,
largeur de 4 cm Hauteur
de 3 cm, largeur de 4 cm
```

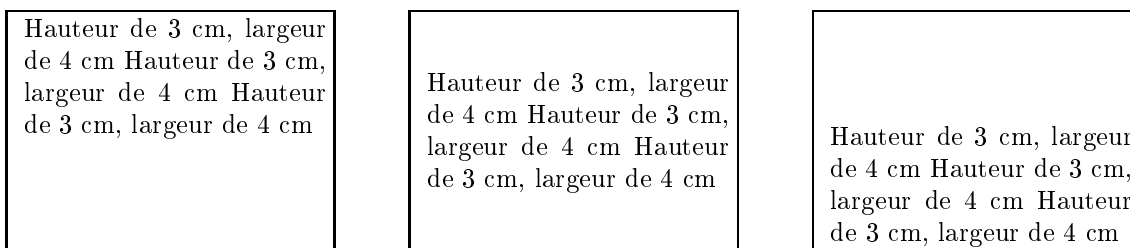
Attention !

Une option de positionnement par rapport à la ligne (ici `[t]`) est indispensable pour que la minipage ait la hauteur choisie ; sinon le paramètre de hauteur n'est pas pris en compte (mais il n'y a pas de message d'erreur).

Les dimensions de la minipage sont les dimensions du texte qui se trouve à l'intérieur, ce ne sont pas les dimensions du cadre (qui sont forcément un peu plus grandes).

## 5 Positionnement intérieur

Et comment faire pour obtenir ces trois minipages ?



Il suffit d'entrer un nouveau paramètre qui détermine la position du texte à l'intérieur de la minipage : avec `[t]` il sera en haut de la minipage, avec `[c]` il sera centré en hauteur, et avec `[b]` il sera en bas de la minipage.

Pour que ce paramètre soit pris en compte, il faut que les deux dont on a parlé précédemment (positionnement sur la ligne et hauteur) soient entrés ; sinon on a droit à un message d'erreur !

Voici le texte exact qui donne cet alignement des trois minipages :

```
\fbox{
\begin{minipage}[t][3cm][t]{4cm}
\vspace{0pt}          %% indispensable avec [t] %%
Hauteur de 3~cm, largeur de 4~cm Hauteur de 3~cm,
largeur de 4~cm Hauteur de 3~cm, largeur de 4~cm
\end{minipage}}
\hfill
\fbox{
\begin{minipage}[t][3cm][c]{4cm}
Hauteur de 3~cm, largeur de 4~cm Hauteur de 3~cm,
largeur de 4~cm Hauteur de 3~cm, largeur de 4~cm
\end{minipage}}
\hfill
\fbox{
\begin{minipage}[t][3cm][b]{4cm}
Hauteur de 3~cm, largeur de 4~cm Hauteur de 3~cm,
largeur de 4~cm Hauteur de 3~cm, largeur de 4~cm
\par\vspace{0cm}      %% indispensable avec [b] %%
\end{minipage}}

```

On peut voir au début de la première minipage (option `[t]`) un `\vspace{0pt}`, et à la fin de la troisième un `\par\vspace{0pt}` ; ce n'est pas nécessaire si on n'entre qu'une minipage, mais si on en a plusieurs, il faut entrer ces lignes sous peine de voir notre bel alignement un peu chamboulé. Essayez sans pour voir !

Merci à ANTHONY YOUND, professeur à NEWCASTLE pour ces précisions.

Enfin vous aurez remarqué que la troisième minipage ne déborde pas dans la marge droite : les `\hfill` font parfaitement leur travail et remplissent d'espaces horizontales la place entre les minipages, sans les pousser en dehors des limites du texte, même si elles sont encadrées !



## 8 – Trucs mathématiques

Voici une chronique consacrée à l'écriture de textes mathématiques et à l'étude de quelques fonctionnalités que j'utilise.

### 1 Fonctions et parenthèses

Combien de fois dans l'écriture de textes mathématiques est-on amené à écrire  $f(x)$  ?

D'abord est-ce qu'on peut voir une différence entre  $f(x)$  et  $f(x)$  ?

Naturellement ! Dans la deuxième version, il y a un petit espacement entre la lettre  $f$  et la parenthèse, et le résultat est nettement plus joli ! Cet espacement est mis automatiquement parce que j'ai utilisé `\left(` et `\right)` à la place de `(` et `)` : `$f \left( x \right)$`

Le `\left(` est la parenthèse gauche qui s'adapte automatiquement à la taille de ce qui suit ; elle doit être obligatoirement suivie d'un `\right` quelque chose, le quelque chose pouvant être une parenthèse fermante, ou une accolade fermante, ou un point (qui sert juste à terminer la parenthèse ouvrante), ou...

#### 1.1 Première amélioration

Comme on entre souvent des parenthèses, au lieu de taper `\left(` puis `\right)` (un peu fastidieux), j'utilise deux petites commandes qui raccourcissent la frappe :

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

Il suffit donc de taper `\left(` à la place de `\left(`, et `\right)` à la place de `\right)`.

Pourquoi `\renewcommand` et pas `\newcommand` ?

Tout simplement parce que les commandes `\left(` et `\right)` existent déjà (à essayer!).

Une fois ces commandes définies dans le préambule du document,

on tape	<code>\$f\left(x\right)\$</code>	pour obtenir	$f(x)$
	<code>\$g\left(\dfrac{3}{4}\right)\$</code>		$g\left(\frac{3}{4}\right)$
	<code>\$A\left(\dfrac{4}{3}, 2\right)\$</code>		$A\left(\frac{4}{3}, 2\right)$

#### 1.2 Deuxième amélioration

Il n'en reste pas moins que taper `$f\left(x\right)$` pour obtenir  $f(x)$  peut encore paraître trop long.

On crée d'autres commandes qui feront le travail :

```
\newcommand{\fx}{f\left(x\right)}
\newcommand{\gx}{g\left(x\right)}
...
```

Il suffira de taper `$\fx$` pour avoir  $f(x)$ .

On pourrait être tenté, pour gagner encore de la frappe, d'intégrer les `$` à l'intérieur de la commande, et de la définir ainsi : `\newcommand{\fx}{$f\left(x\right)$}`

Il suffirait donc de taper `\fx` pour obtenir  $f(x)$ .

À l'usage, ce n'est pas une bonne idée car on écrit rarement  $f(x)$  tout seul, mais plutôt  $f(x) = \dots$  et il faut se mettre de toute façon en mode mathématique pour écrire ce qui suit le  $f(x)$ .

## 2 Personnalisation du mode mathématique

On aimerait bien parfois écrire une formule mathématique en couleur ou en gras pour la mettre en évidence dans un texte.

### 2.1 Couleur

Comme tout texte, on peut mettre en couleur un texte mathématique : par exemple pour écrire « l'ensemble solution est  $S = \{1; 2\}$  », il suffit d'entrer :

```
l'ensemble solution est {\red $S=\{1\,; 2\}$}
```

Les accolades servent à délimiter la zone de texte qui sera en rouge.

On peut utiliser le même procédé à l'intérieur d'une formule mathématique comme  $f(x) = x^2$  en entrant `\fx= {\red x^2}`.

Pour mettre en rouge tout ce qui est écrit en mode mathématique, il faut entrer au début du texte, la commande `\everymath{\color{red}}` :

Dans ce paragraphe, tous les textes mathématiques seront automatiquement écrits en rouge : soit  $f$  la fonction définie sur  $[-2, 3]$  par  $f(x) = x^2$ .

Pour remettre en mode normal, on entrera `\everymath{\color{black}}`

### 2.2 Gras

Pour mettre en **gras** un mot dans un texte, on écrit `\textbf{gras}`; si on essaie avec une formule mathématique, ça ne fonctionne pas!

Pour mettre une formule en gras, on utilisera `\boldmath` ainsi :

```
{\boldmath $\fx=x^2$} donne  $f(x) = x^2$ 
```

La commande `\boldmath` ne doit pas être utilisée en mode mathématique (c'est-à-dire entre des \$).

Si, dans un texte mathématique, on veut mettre une partie d'une formule en gras, on utilisera `\boldsymbol` :

```
$\fx = \boldsymbol{x^2}$ donne  $f(x) = x^2$ 
```

Enfin si on veut que dans un texte, tout ce que l'on écrit en mode mathématique soit en gras, on utilisera `\mathversion{bold}` dont on annulera l'effet par `\mathversion{normal}`.

Si on entre :

```
\mathversion{bold}
\begin{quote}
Dans ce paragraphe, toutes les formules mathématiques seront en gras:
soit  $f$  la fonction définie sur  $[0[, +\infty[$  par  $\fx=\dfrac{x-1}{x^2+2}$ .
\end{quote}
\mathversion{normal}
```

on obtient :

Dans ce paragraphe, toutes les formules mathématiques seront en gras : soit  $f$  la fonction définie sur  $[0, +\infty[$  par  $f(x) = \frac{x-1}{x^2+2}$ .

Au passage, on peut voir l'environnement `quote` qui ajoute un retrait gauche et un retrait droit au paragraphe pour le mettre en exergue.

On peut naturellement combiner la couleur et le gras dans le mode mathématique.



### 3 Accolade et superposition de texte

On peut facilement couvrir une expression avec une accolade sur laquelle on écrira du texte, comme dans :

$$a^n = \overbrace{a \times a \times \dots \times a}^{n \text{ facteurs}}$$

La commande utilisée est `\overbrace{}^{\}`. Ce que l'on entre comme premier paramètre est le texte qui sera situé sur la ligne courante, et on entrera comme deuxième paramètre ce que l'on voudra écrire au dessus de l'accolade. On entrera donc :

```
$a^n = \overbrace{a \times a \times \dots \times a}^{n \text{ facteurs}}$
```

Ceux qui préfèrent écrire en dessous plutôt qu'au dessus choisiront :

```
$a^n = \underbrace{a \times a \times \dots \times a}_{n \text{ facteurs}}$
```

pour obtenir :

$$a^n = \underbrace{a \times a \times \dots \times a}_{n \text{ facteurs}}$$

`over` a été remplacé par `under`, et le signe d'exposant `^` a été remplacé par le signe d'indice `_`.

On peut naturellement utiliser à la fois `\overbrace` et `\underbrace` dans une même formule :

$$\binom{n}{p} = \frac{n!}{p!(n-p)!} = \frac{\overbrace{n \times (n-1) \times \dots \times (n-p+1)}^{p \text{ facteurs}}}{\underbrace{p \times (p-1) \times \dots \times 2 \times 1}_{p \text{ facteurs}}}$$

Dans le même esprit, on peut écrire un nombre en binaire  $\overline{10110011}^2$  en utilisant `\overline{\}` :

```
$\overline{1011\,0011}^2$
```

On remarque que le statut de `^{\}` n'est pas le même après `\overbrace` qu'après `\overline{\}` ; dans le premier cas, ce qui est entre accolades après le signe exposant est écrit au dessus de la grande accolade, dans le deuxième cas c'est écrit normalement en exposant ; il faudrait donc procéder différemment pour écrire  $\overline{1234}^2$  (que l'on rencontre peu en mathématiques, il est vrai!).

### 4 Crochets

Les crochets que l'on écrit à partir du clavier sont un peu petits :  $[0,1]$ .

Ceux-ci sont plus lisibles :  $[0,1]$ .

Mais comme ils n'existent pas, il a fallu les dessiner au moyen de `\rceil` et `\rfloor`, `\lceil` et `\lfloor`.

Voici les deux commandes qui définissent les crochets, `\cd` pour `crochet droit` et `\cg` pour `crochet gauche` :

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

On obtient donc  $[0,1]$  en tapant `$\cd 0\,, 1\cg$`.

L'instruction `\hspace{-4.5pt}` permet de revenir en arrière de 4,5 points, de façon à aligner les deux signes `\rceil` et `\rfloor`.

On peut également contruire les crochets permettant d'écrire des intervalles d'entiers comme  $\llbracket 0, 12 \rrbracket$  en utilisant le même procédé.

Il suffit pour cela de créer deux nouvelles commandes `\cgZ` et `\cdZ`.  
Attention, il faut écrire la définition de la commande sur une seule ligne.

```
\newcommand{\cgZ}{\rceil \hspace{-4.5pt} \rfloor \hspace{-3pt}
\rceil \hspace{-4.5pt} \rfloor}
\newcommand{\cdZ}{\lceil \hspace{-4.5pt} \lfloor \hspace{-3pt}
\lceil \hspace{-4.5pt} \lfloor}
```

On entrera donc `\cdZ 0\,, 12 \cgZ` pour écrire  $\llbracket 0, 12 \rrbracket$ .

Enfin, si on veut des crochets qui s'adaptent à la hauteur du texte, on utilisera `\left[` et `\right]` comme dans  $\left[ \frac{1}{2}, \frac{3}{4} \right]$  que l'on obtient par : `\left[ \dfrac{1}{2}\,, \dfrac{3}{4}\right]`

## 5 Alignement

On est parfois amené à écrire des calculs mathématiques qui tiennent sur plusieurs lignes :

$$f'(x) = 2x(e^{x-1} - x - 1) + x^2(e^{x-1} - 1) = 2xe^{x-1} - 2x^2 - 2x + x^2e^{x-1} - x^2 = e^{x-1}(2x + x^2) - x(3x + 2)$$

Ce serait plus joli (et surtout plus facile à lire) de présenter ce calcul ainsi :

$$\begin{aligned} f'(x) &= 2x(e^{x-1} - x - 1) + x^2(e^{x-1} - 1) \\ &= 2xe^{x-1} - 2x^2 - 2x + x^2e^{x-1} - x^2 \\ &= e^{x-1}(2x + x^2) - x(3x + 2) \end{aligned}$$

On peut voir dans cette deuxième version que les signes `=` sont parfaitement alignés verticalement. On pourrait utiliser un tableau ou poser des taquets de tabulation, mais il y a une autre méthode pour laisser en début de deuxième et de troisième ligne, un espacement de la largeur exacte de l'expression  $f'(x)$ .

On utilise pour ça la commande `\phantom{}` :

```
$f'(x)=2x\(\e^{x-1}-x-1)+x^2\(\e^{x-1}-1)\ \ \ [3pt]
\phantom{f'(x)}= 2x\e^{x-1}-2x^2-2x+x^2\e^{x-1}-x^2 \ \ \ [3pt]
\phantom{f'(x)}= \e^{x-1}\(2x+x^2)-x\ (3x+2)\$
```

L'instruction `\phantom{f'(x)}` réserve la place exacte que prend l'écriture de  $f'(x)$ , ce qui permet l'alignement vertical des signes `=`.

Petits rappels : `\` permet de passer à la ligne suivante (tout en restant dans le même paragraphe), et `[3pt]` (obligatoirement placé après `\`) crée un espacement vertical de 3 points.

On peut voir aussi l'instruction `\e` créée par `\newcommand{\e}{\,\text{e}\,}` qui permet d'écrire le `e` de l'exponentielle en mode texte avec des petites espaces `\`, avant et après.

## 9 – Tableaux

On a de nombreuses occasions d'écrire des tableaux en mathématiques : tableau de valeurs, tableau de signes, tableau de variations... Mais on verra aussi que les tableaux sont utilisés en d'autres occasions, par exemple pour écrire des systèmes ou pour représenter des divisions.

Le principe des tableaux est assez simple, mais pour faire exactement ce que l'on veut, ça peut être fastidieux au début.

On a déjà vu des tableaux pour écrire des matrices dans le chronique 3 (page 11) ; que les débutants en  $\text{\LaTeX}$  la relisent avant de se lancer dans cette chronique.

Comme pour les autres chroniques, il n'est pas question d'être exhaustif sur le sujet ; je vais juste expliquer comment j'utilise les tableaux et en présenter des exemples.

### 1 Tableau de valeurs

Le tableau de valeurs d'une fonction  $f$ , est un simple tableau à deux lignes ou deux colonnes, voire trois si on fait écrire les coordonnées de points de la courbe représentant  $f$  :

$x$	-2	-1	0	1	2	3
$f(x) = x^2$	4	1	0	1	4	9
point	(-2, 4)	(-1, 1)	(0, 0)	(1, 1)	(2, 4)	(3, 9)

Le code :

```

 $\begin{array}{|c|c|c|c|c|c|c|}
\hline
x & -2 & -1 & 0 & 1 & 2 & 3 \\
\hline
f(x)=x^2 & 4 & 1 & 0 & 1 & 4 & 9 \\
\hline
\text{point} & (-2,, 4) & (-1,, 1) & (0,, 0) & (1,, 1) \\
& & & & & (2,, 4) & (3,, 9) \\
\hline
\end{array}$ 

```

et les explications :

$\begin{array}{|c|c|c|c|c|c|c|}$  ☞ définit un tableau de type `array` (donc en mode mathématique) de sept colonnes centrées, séparées et encadrées par des traits verticaux (AltGr 6)

`\hline` ☞ trace un trait horizontal

`&` ☞ passe d'une colonne à la suivante

`\` ☞ change de ligne

`\text{point}` ☞ écrit du texte en mode mathématique

`\end{array}` ☞ ferme l'environnement `array` et sort du mode mathématique

Si toutes les colonnes du tableau sont de même type, comme c'est le cas ici, on peut remplacer, dans la définition du tableau, `|c|c|c|c|c|c|c|` par `*{7}{c|}` ce qui donne donc comme ligne de définition du tableau  $\begin{array}{|*{7}{c|}}$  ; on crée 7 colonnes centrées et suivies d'un trait vertical, le tout précédé d'un trait vertical.

Si on trouve les lignes horizontales un peu serrées (le carré de  $x^2$  touche la ligne du dessus), on peut les espacer en redéfinissant la commande `\arraystretch` qui contrôle l'espacement des lignes (déjà vu page 11). Cette redéfinition doit se faire avant de déclarer le tableau.

On obtient :

$x$	-2	-1	0	1	2	3
$f(x) = x^2$	4	1	0	1	4	9
point	(-2, 4)	(-1, 1)	(0, 0)	(1, 1)	(2, 4)	(3, 9)

Et le code à entrer est :

```
\renewcommand{\arraystretch}{1.2}
$\begin{array}{|*{7}{c|}}
...
\end{array}$
\renewcommand{\arraystretch}{1}
```

On remarque que le trait vertical s'adapte à la hauteur de la ligne (heureusement!).

Je ne sais pas trop quel est le statut de `\arraystretch`.

Certains auteurs disent que c'est la hauteur des lignes (je ne le crois pas), certains autres parlent de « ratio » qui laisse penser à un coefficient multiplicateur ; j'aurais tendance à croire ces derniers. Mais c'est sûr que `\arraystretch` tient compte de ce qui est écrit entre les lignes ; essayez de mettre `\renewcommand{\arraystretch}{0}` pour voir!

## 2 Fusion de cellules horizontales

Voici un petit tableau donnant les trois identités remarquables « de base » :

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

Analysons ce tableau : il a manifestement trois colonnes, celle de gauche justifiée à droite, celle de droite justifiée à gauche et celle du milieu centrée. Il a une double ligne horizontale en haut et en bas du tableau, et pas de traits verticaux séparant les colonnes. Mais surtout sa première ligne contient du texte qui est à cheval sur les trois colonnes : les cellules ont été fusionnées.

C'est ce que fait la commande `\multicolumn{nb}{type}{texte}` où :

nb désigne le nombre de cellules à fusionner
type désigne c pour center, l pour left et r pour right
texte désigne le texte à écrire

Pour tracer un trait vertical à gauche et à droite du texte situé dans les cellules fusionnées, on entrerait comme type : |c|

Pour obtenir le tableau ci-dessus, on tapera :

```
\renewcommand{\arraystretch}{1.4}
$\begin{array}{|r c l}
\hline \hline % trace deux lignes horizontales
\multicolumn{3}{c}{\textbf{Identités remarquables}}\\
\hline \text{Forme développée} & & \text{Forme factorisée}\\
a^2+2ab+b^2 & = & \ (a+b)^2\\
a^2-2ab+b^2 & = & \ (a-b)^2\\
a^2-b^2 & = & \ (a+b)\ (a-b)\\
\hline \hline
\end{array}$
\renewcommand{\arraystretch}{1}
```

Une petite remarque pédagogique : j'ai toujours pensé qu'un professeur devrait présenter exclusivement les identités remarquables sous la forme  $a^2 + 2ab + b^2 = (a + b)^2$  et pas sous la forme  $(a + b)^2 = a^2 + 2ab + b^2$  comme on le voit dans 99 % des cas (et même dans les programmes officiels). Pour développer  $(2x + 3)^2$ , on peut toujours développer  $(2x + 3)(2x + 3)$  (ce qui engendre beaucoup moins d'erreurs chez les élèves), mais pour factoriser  $4x^2 + 12x + 9$  il est indispensable de savoir que  $a^2 + 2ab + b^2$  se factorise en  $(a + b)^2$ . Fin de remarque!

Dans l'exemple que l'on vient de voir, on a fusionné trois cellules horizontales dans un tableau qui comportait trois colonnes; on peut ne fusionner que deux cellules sur trois, ou seulement trois cellules dans un tableau qui comporte quatre colonnes :

Série	Année		
	2010	2011	2012
S	100 %	100 %	100 %

Deux nouveautés dans ce tableau : un double trait vertical après la première colonne, et un trait horizontal après la première ligne qui ne démarre qu'à la deuxième colonne.

Pour le double trait, il suffit de rentrer `||` à la place de `|`.

Quant au trait horizontal partiel, c'est l'instruction `\cline{2-4}` qui permet de tracer cette ligne horizontale de la 2<sup>e</sup> colonne à la 4<sup>e</sup>.

Voici le code pour dessiner ce tableau :

```
\renewcommand{\arraystretch}{1.2}
\begin{tabular}{|c||c|c|c|}
\hline
Série & \multicolumn{3}{c|}{Année} \\
\cline{2-4}
& 2010 & 2011 & 2012 \\
\hline
S & 100\,\% & 100\,\% & 100\,\% \\
\hline
\end{tabular}
\renewcommand{\arraystretch}{1}
```

Comme le caractère `%` est un caractère réservé (qui sert à écrire des commentaires), il faut, pour pouvoir l'afficher, le faire précéder d'un antislash `\`.

Pour ce tableau qui ne contient pas de texte mathématique, j'ai utilisé l'environnement `tabular`. On peut aussi utiliser cet environnement pour dessiner des tableaux contenant à la fois du texte et des formules mathématiques; il faudra juste entourer chaque formule mathématique des fameux `$`. Et si on veut écrire du texte dans un tableau créé dans l'environnement `array`, il faudra utiliser la fonction `\text{}`, ou `\textbf{}` si on veut que le texte soit en gras (voir page 36).

On doit choisir son environnement à la création du tableau mais on ne peut pas avoir le beurre et l'argent du beurre!

Quoique...

On peut, dans un environnement `tabular`, définir un type de colonne dans laquelle on entrera des formules mathématiques sans les encadrer par des `$`; c'est le type de la colonne qui rajoutera en début et en fin de texte les `$` nécessaires.

Mais ce sera pour une autre chronique...

Deux remarques pour terminer ce paragraphe :

- l'instruction `\multicolumn` ne nécessite pas d'extension particulière;
- il existe une instruction similaire à `\multicolumn` pour fusionner des cellules situées l'une en dessous de l'autre : c'est `\multirow` qui nécessite le package `multirow`.

### 3 Tableau de signes

Passons maintenant au tableau de signes :

$x$	$-\infty$	$-2$	$3$	$+\infty$
$3 - x$	<b>+</b>	<b>+</b>	<b>0</b>	<b>-</b>
$2x + 4$	<b>-</b>	<b>0</b>	<b>+</b>	<b>+</b>
$\frac{2x + 4}{3 - x}$	<b>-</b>	<b>0</b>	<b>+</b>	<b>-</b>

Il s'agit d'un tableau de type `array`, de huit colonnes et quatre lignes.

La hauteur des lignes a été augmentée par une modification de `\arraystretch`. La dernière ligne a été traitée de façon spéciale pour en augmenter la hauteur (voir la commande `\rule` page 12).

Les traits verticaux ont été créés avec `\vline`, que l'on double si on veut une double barre (en intercalant une petite espace `\,`).

Pour mettre un 0 à cheval sur le trait, il faut tracer le trait puis revenir un peu en arrière de 2,7 points puis écrire le 0.

Si on veut que les traits verticaux soient bien alignés, qu'il y ait le 0 ou non, il faut mettre un « faux » 0 au moyen de la commande `\phantom{}` (voir page 34) ; c'est parce que la colonne est de type `center`.

Enfin, pour être plus visibles, les signes + et - ont été mis en gras au moyen de `\boldsymbol{}` (voir page 32).

```

\renewcommand{\arraystretch}{1.5}
$\begin{array}{|c|*{7}{c}|}
\hline
x & -\infty & & -2 & & 3 & & +\infty \\
\hline
3-x & & \boldsymbol{+} & & \vline\hspace{-2.7pt}{\phantom{0}} & & \boldsymbol{-} & \\
\hline
2x+4 & & \boldsymbol{-} & & \vline\hspace{-2.7pt}{0} & & \boldsymbol{+} & \\
\hline
\frac{2x+4}{3-x} & & \boldsymbol{-} & & \vline\hspace{-2.7pt}{0} & & \boldsymbol{+} & \\
\hline
\end{array}$
\renewcommand{\arraystretch}{1}

```

Cela donne un tableau de signes convenable, non ?

De toute façon, vous ferez comme moi : du copier-coller ! C'est beaucoup moins long de modifier un tableau existant que d'en reconstruire un de toute pièce.

Mais pour le modifier efficacement, il faut avoir compris comment il a été créé ; j'espère que c'est maintenant le cas pour tout le monde !

## 10 – Tableaux - Suite

D'autres tableaux sont présentés dans cette chronique, le sujet est inépuisable!

### 1 Systèmes et accolades

On emploie aussi des tableaux pour écrire des systèmes :

$$\left\{ \begin{array}{l} x + 2y = -3 \\ 2x - y = 4 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} x + 2(2x - 4) = -3 \\ 2x - 4 = y \end{array} \right. \Leftrightarrow \dots$$

Voici le code du premier système :

```

 $\left\lbracket$  % accolade gauche
 $\begin{array}{r @{\;=\;} l}$  % définition du tableau
x+2y & -3\\
2x-y & 4
 $\end{array}$  % fin du tableau
 $\right.$  % termine l'accolade

```

Le tableau de type `array` est inséré entre une accolade ouvrante `\left\lbracket` et `\right.`.

Le tableau possède deux colonnes séparées par `\;=\;` ; c'est ce que fait `@{\;=\;}` que l'on a entré dans la définition du tableau : quand on tape le signe `&`, il s'affiche le signe `=` encadré d'espaces.

Sur le même principe, on peut écrire :

$$\left. \begin{array}{l} A \in \mathcal{P} \\ B \in \mathcal{P} \end{array} \right\} \Rightarrow (AB) \subset \mathcal{P} \quad \left. \begin{array}{l} C \in \mathcal{P} \\ D \in \mathcal{P} \end{array} \right\} \Rightarrow (CD) \subset \mathcal{P} \quad \left. \right\} \Rightarrow (AB) \text{ et } (CD) \text{ sont coplanaires.}$$

en faisant un tableau de tableaux :

```

 $\left.$  % prépare la grande accolade droite
 $\begin{array}{r}$  % début du tableau 1
 $\left.$ 
 $\begin{array}{r}$  % début du tableau 2
A \in \mathscr P \\
B \in \mathscr P
 $\end{array}$  % fin du tableau 2
 $\right\rbrace$  % accolade droite
 $\Rightarrow$  % implique
 $(AB) \subset \mathscr P$ 
 $\left.$ 
 $\begin{array}{r}$  % début du tableau 3
C \in \mathscr P \\
D \in \mathscr P
 $\end{array}$  % fin du tableau 3
 $\right\rbrace$ 
 $\Rightarrow$   $(CD) \subset \mathscr P$ 
 $\end{array}$  % fin du tableau 1
 $\right\rbrace$  % grande accolade droite
 $\Rightarrow$   $(AB)$  et  $(CD)$  sont coplanaires.

```

On fait ainsi connaissance avec  $\in$  que l'on écrit `\in`, et avec  $\subset$  que l'on écrit `\subset`.

Le nom du plan  $\mathcal{P}$  en police script s'obtient par `\mathscr P` ; et comme le paramètre ne comporte qu'un seul caractère,  $P$ , on peut écrire `\mathscr P` au lieu de `\mathscr{P}`.

## 2 Tableau de variations

Voici un petit tableau de variations qui n'est pas trop moche :

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

C'est l'environnement `array` qui est utilisé pour tracer ce tableau ; six colonnes et cinq lignes : une pour  $x$ , une pour la dérivée, et trois pour les variations de la fonction  $f$ .

La seule nouveauté est le tracé des flèches pour représenter les variations ; j'ai utilisé `\psline` que l'on peut employer hors de l'environnement `\pspicture` (voir page 13).

Et pour répéter le dessin de ces flèches, j'ai créé deux commandes `\flh` pour « flèche haute », et `\flb` pour « flèche bas ».

J'avais commencé par définir :

```
\newcommand{\flh}{\psline[arrowsize=3pt 3]{->}(-0.5,-0.3)(0.5,0.3)}
\newcommand{\flb}{\psline[arrowsize=3pt 3]{->}(-0.5,0.3)(0.5,-0.3)}
```

mais les flèches étaient un peu trop près des nombres, surtout du  $-5$  ; j'ai donc rajouté des espaces.

Voici le code qui donne le tableau de variations :

```
\begin{center}
\newcommand{\flh}{\hspace*{0.3cm}\psline[arrowsize=3pt 3]{->}
(-0.5,-0.3)(0.5,0.3) \hspace*{0.3cm}} % À écrire sur une seule ligne
\newcommand{\flb}{\hspace*{0.3cm}\psline[arrowsize=3pt 3]{->}
(-0.5,0.3)(0.5,-0.3) \hspace*{0.3cm}} % À écrire sur une seule ligne
$\begin{array}{|c|*5{c}|}
\hline
x & -\infty & & -3 & & +\infty \quad \rule[-8pt]{0pt}{22pt} \\
\hline
f'(x) & & \boldsymbol{-} & & \boldsymbol{+} & & \rule[-6pt]{0pt}{18pt} \\
\hline
& 0 & & & & +\infty \quad \rule{0pt}{12pt} \\
f & & \flb & & \flh & & \\
& & & -5 & & & \\
\hline
\end{array} $
\end{center}
```

On peut compliquer et intégrer une valeur interdite :

$x$	$-\infty$	$-3$	$4$	$+\infty$
$f'(x)$	$-$	$\emptyset$	$+$	$-$
$f$	$0$	$-5$	$+\infty$	$+\infty$
				$-\infty$



en entrant ce code :

```
\begin{center}
\newcommand{\flh}{\hspace*{0.5cm}\psline[arrowsize=3pt 3]{->}
(-0.5,-0.3)(0.5,0.3) \hspace*{0.5cm}} % Sur une seule ligne
\newcommand{\flb}{\hspace*{0.5cm}\psline[arrowsize=3pt 3]{->}
(-0.5,0.3)(0.5,-0.3) \hspace*{0.5cm}}
$\begin{array}{|c|*9{c}|}
\hline
x & -\infty & & -3 & & & 4 & & & +\infty & \rule[-8pt]{0pt}{22pt} \\
\hline
f'(x) & & \boldsymbol{-} & \vline\hspace{-2.7pt}0 & & \boldsymbol{+} & & & & & \rule[-6pt]{0pt}{18pt} \\
& & \vline\, & \vline & & \boldsymbol{-} & & & & & \rule[-6pt]{0pt}{18pt} \\
\hline
& 0 & & & & +\infty & \vline\, & \vline & & +\infty & & \rule{0pt}{12pt} \\
f & & \flb & & \flh & & \vline\, & \vline & & \flb & & \\
& & & & -5 & & & \vline\, & \vline & & & -\infty & \rule[-5pt]{0pt}{17pt} \\
\hline
\end{array} $
\end{center}
```

Il faut signaler l'intéressant package `variations` créé par CHRISTIAN OBRECHT qui permet de construire des tableaux de variations très rapidement ; voir la documentation à l'adresse :

<http://ctan.mirror.ac.za/macros/generic/variations/docvariations.pdf>

le téléchargement de l'extension pouvant se faire à partir de l'adresse :

<http://obrecht.fr/variations/>

### 3 Trait en diagonale

On a parfois besoin de tracer des traits en diagonale dans un tableau :

$\lim f \backslash$	$\ell$	$+\infty$	$-\infty$
$\lim g$	$\ell + \ell'$	$+\infty$	$-\infty$
$+\infty$	$+\infty$	$+\infty$	?
$-\infty$	$-\infty$	?	$-\infty$

Tout d'abord on peut voir comment écrire le  $\ell$  de limite, en entrant `ell`.

Mais surtout la nouveauté du tableau est le trait de séparation oblique.

Il faut d'abord charger le package `diagbox` en tapant `\usepackage{diagbox}`.

Le code pour obtenir le tableau est :

```
\renewcommand{\arraystretch}{1.2}
\begin{tabular}{|c|c|c|c|}
\hline
\diagbox{\lim $g$}{\lim $f$} & $ell$ & $+\infty$ & $-\infty$ \\
\hline
$ell'$ & $ell + ell'$ & $+\infty$ & $-\infty$ \\
\hline
$+\infty$ & $+\infty$ & $+\infty$ & ? \\
\hline
$-\infty$ & $-\infty$ & ? & $-\infty$ \\
\hline
\end{tabular}
```

```

 $-\infty$  &  $-\infty$  & ? &  $-\infty$  \\
\hline
\end{tabular}
\renewcommand{\arraystretch}{1}

```

L'instruction `\diagbox` nécessite deux paramètres dont on comprend bien le fonctionnement. Cette instruction n'est pas valide en mode mathématique, donc il faut l'employer dans l'environnement `tabular` et pas dans l'environnement `array`.

L'extension `diagbox` a été créée par LEO LIU et remplace le package `slashbox`; le fichier `diagbox.pdf`, écrit par l'auteur du package, donne toutes les options possibles :

<http://ctan.mines-albi.fr/macros/latex/contrib/diagbox/diagbox.pdf>

## 4 Largeur fixe

Les explications du paragraphe 1 de la page 35 sont données dans un tableau à deux colonnes séparées par un ☞ obtenu par `\ding{43}`; ce caractère spécial a besoin de l'extension `pifont`. Mais le plus nouveau de ce tableau est la cellule de droite de la 1<sup>re</sup> ligne puisqu'elle contient plusieurs lignes de texte : elle a été définie de largeur fixe et le texte est automatiquement justifié : c'est `p{8cm}` qui permet ça.

On définit donc ce tableau par la ligne : `\begin{tabular}{r @{\ \ \ding{43}\ \ } p{8cm}}`

Le package `array` permet plus d'options pour les tableaux; on en trouve le mode d'emploi à l'adresse : <http://ctan.mackichan.com/macros/latex/required/tools/array.pdf>

## 5 Épaisseur du trait

Pour terminer cette rubrique, on va voir comment augmenter l'épaisseur des traits qui sont tracés dans un tableau; cette épaisseur est contrôlée par la variable `\arrayrulewidth` que l'on peut modifier. Sa valeur par défaut est 0,4 point.

Le code :

```

\begin{center}
{
% début de paragraphe
\setlength{\arrayrulewidth}{1pt}
 $\begin{array}{|*7{c|}}$ 
\hline
x & -2 & -1 & 0 & 1 & 2 & 3 \\
\hline
 $f(x)=x^2$  & 4 & 1 & 0 & 1 & 4 & 9 \rule{0pt}{10pt} \\
\hline
\end{array}
}
% fin de paragraphe
\end{center}

```

donnera :

$x$	-2	-1	0	1	2	3
$f(x) = x^2$	4	1	0	1	4	9

On peut aussi s'arranger, en utilisant d'autres outils, pour ne pas épaissir tous les traits d'un tableau, mais seulement quelques-uns.

Mais c'est une autre histoire...

# 11 – Mise en page

Un document  $\text{\LaTeX}$  se doit d'être joli ; joli au sens de bien présenté : il ne faut pas multiplier les polices de caractères, il ne faut pas abuser du gras, il faut éviter le souligné et évidemment bannir totalement le gras souligné!

En plus, il faut laisser des marges confortables autour du texte de façon à faciliter la lisibilité du document.

On va voir dans cette chronique comment gérer les marges, comment définir des en-têtes et des pieds de page, et également comment modifier les marges prédéfinies pour un paragraphe donné.

## 1 Marges d'un document

Depuis que j'écris en  $\text{\LaTeX}$  j'utilise le package `vmargin` créé par V. KUHLMANN, et qui permet de définir très facilement les marges d'un document (sans doute que le premier document que j'ai recopié utilisait cette extension!).

On peut télécharger ce package à l'adresse :

```
ftp://ftp.inria.fr/pub/TeX/CTAN/macros/latex/contrib/other/misc/
```

Mais si votre environnement  $\text{\LaTeX}$  est bien configuré, il suffit d'écrire `\usepackage{vmargin}` dans le préambule de votre document et le système devrait télécharger et installer tout seul ce package. Ensuite l'instruction `\setmarginsrb` suivie de huit paramètres fait le travail demandé.

Voici les explications des paramètres :

```
\setmarginsrb{1}{2}{3}{4}{5}{6}{7}{8}
% 1 est la marge gauche
% 2 est la marge en haut
% 3 est la marge droite
% 4 est la marge en bas
% 5 fixe la hauteur de l'entête
% 6 fixe la distance entre l'entête et le texte
% 7 fixe la hauteur du pied de page
% 8 fixe la distance entre le texte et le pied de page
```

et comment j'utilise cette instruction dans ces chroniques :

```
\setmarginsrb{3cm}{2.5cm}{3cm}{2.5cm}{0cm}{1cm}{0cm}{1.5cm}
```

### Remarque

On peut augmenter ponctuellement la longueur d'une page ; il suffit d'écrire dans la page en question l'instruction :

```
\enlargethispage{1cm}
```

pour en augmenter sa longueur de 1 cm.

## 2 Augmentation des marges d'un paragraphe

Il existe deux environnements `quote` et `quotation` qui permettent de mettre en exergue des phrases ou des paragraphes, en augmentant les marges gauche et droite d'une valeur donnée.

Inconvénient : on est obligé d'augmenter les deux marges en même temps, et on ne peut pas définir soi-même l'augmentation que l'on désire, sauf en redéfinissant les décalages par défaut de ces deux environnements.

J'ai trouvé à l'adresse <http://www.grappa.univ-lille3.fr/FAQ-LaTeX/6.7.html> un environnement qui fait exactement ce que je souhaitais : réduire ou augmenter la marge gauche, réduire ou augmenter la marge droite.

En voici le code qu'il faut insérer dans le préambule du document :

```
\newenvironment{changemargin}[2]{\begin{list}{}{
\setlength{\topsep}{0pt}%
\setlength{\leftmargin}{0pt}%
\setlength{\rightmargin}{0pt}%
\setlength{\listparindent}{\parindent}%
\setlength{\itemindent}{\parindent}%
\setlength{\parsep}{0pt plus 1pt}%
\addtolength{\leftmargin}{#1}%
\addtolength{\rightmargin}{#2}%
}\item }\end{list}}
```

On l'utilise de la façon suivante :

```
\begin{changemargin}{1cm}{0cm}
...
...
...
\end{changemargin}
```

si l'on veut augmenter la marge gauche de 1 cm sans toucher à la marge droite.

On peut mettre une valeur négative comme paramètre pour diminuer ponctuellement les marges d'un ou plusieurs paragraphes.

## 3 En-tête et pied de page

### 3.1 Mode normal

Un autre package que j'utilise depuis mes débuts en L<sup>A</sup>T<sub>E</sub>X est `fancyhdr` pour gérer les en-têtes et pieds de page. La documentation complète de cette extension se trouve dans le document `fancyhdr.pdf` disponible à l'adresse

<ftp://www.tug.org/tex-archive/macros/latex/contrib/fancyhdr/fancyhdr.pdf>

Voici quelques indications pour ceux qui ne souhaitent pas lire les 26 pages en anglais de ce texte.

Il y a trois positions pour l'en-tête, trois pour le pied : gauche (L), centre (C) et droite (R).

L'en-tête est géré par l'instruction `\fancyhead` que l'on décline trois fois : `\fancyhead[L]{gauche}`, `\fancyhead[C]{milieu}` et `\fancyhead[R]{droite}`.

On utilise de la même manière `\fancyfoot` pour le pied de page.

Voici un exemple qui sera plus parlant qu'un long discours ; les instructions suivantes sont à placer dans le préambule du document, avant la ligne `\begin{document}`.

```
\usepackage{fancyhdr}
\pagestyle{fancy} % utilisation du style fancy
\fancyhead[L]{FH} % initiales en haut à gauche
\fancyhead[C]{} % rien en haut au milieu
\fancyhead[R]{2013} % 2013 en haut à droite
\fancyfoot[L]{} % rien en bas à gauche
\fancyfoot[C]{\thepage} % le numéro de page en bas au milieu
\fancyfoot[R]{} % rien en bas à droite
```

On voit que par défaut, cet environnement trace un filet en dessous de l'en-tête; ce filet a une épaisseur de 0,4 point. Cette épaisseur est gérée par la variable `\headrulewidth`.

Si on veut supprimer ce filet, il suffit d'en mettre l'épaisseur à zéro :

```
\renewcommand{\headrulewidth}{0pt}
```

et si on veut qu'il soit plus visible, on peut en augmenter l'épaisseur :

```
\renewcommand{\headrulewidth}{1pt}
```

La variable `\headrulewidth` est définie par l'environnement `fancyhdr` donc il faut la redéfinir si on veut en changer les paramètres, ce qui explique l'utilisation de `\renewcommand` et non pas de `\newcommand`.

De la même façon, le filet de bas de page est géré par la variable `\footrulewidth` qui est d'une épaisseur de 0 point par défaut. On tracera un filet d'épaisseur 1 point en bas de page en tapant :

```
\renewcommand{\footrulewidth}{1pt}
```

### Remarque

On peut souhaiter qu'une page n'aie pas d'en-tête ni de pied de page; ça peut être le cas pour une page de garde, ou pour une page vide à la fin d'un chapitre.

Il suffit pour cela d'entrer l'instruction `\thispagestyle{empty}` dans la page en question.

En fait `empty` est un style prédéfini du package `fancyhdr` et qui fait bien ce qu'on veut de lui!

## 3.2 Mode recto verso

Il ne vous aura pas échappé que, dans ces pages que vous avez sous les yeux, les en-têtes et les pieds de page sont différents selon que la page a un numéro impair (donc destinée à être du côté droit du fascicule) ou un numéro pair (pour le côté gauche); ceci pour que les pages, imprimées en recto-verso, aient toujours le numéro de la page vers l'extérieur de la page.

On peut, avec le package `fancy`, différencier les en-têtes et pieds de page selon la parité de la pagination.

Un peu de vocabulaire anglais tout d'abord : pair se dit `even` et impair se dit `odd`.

On entre donc un paramètre optionnel `O` pour les pages impaires, que l'on va combiner avec `R` pour `right`.

Un exemple :

```
\fancyfoot[R0]{\thepage}
```

mettra le numéro de page à droite si la page est impaire (ce que l'on souhaite).

On peut en une seule fois faire écrire le numéro à droite sur une page impaire, et à gauche sur une page paire; il suffit d'entrer :

```
\fancyfoot[R0,LE]{\thepage}
```

Mais si on entre ça comme pied de page, on se rend compte que ça ne donne pas le résultat attendu. En effet, il faut dire au système que l'on veut imprimer en recto-verso ce qui se fait en première ligne du document; au lieu de :

```
\documentclass[10pt, a4paper]{article}
```

il faut entrer :

```
\documentclass[10pt, a4paper, twoside]{article}
```

Du moins si on est en classe `article`, celle que j'utilise en général.

## 3.3 Ajustement à la largeur de la marge

Quelques collègues facétieux ont voulu diminuer les marges de leurs documents, donc augmenter la largeur d'écriture, et ils ont vu que les en-têtes et pieds de page restaient à la même dimension; cela faisait très moche, particulièrement s'il y avait un filet pour l'en-tête ou le pied de page.

Ils m'ont posé la question et je n'ai pas su y répondre tout de suite.

Depuis j'ai gratté un peu dans la documentation de `fancyhdr` et j'ai lu que l'ajustement de la largeur des en-têtes et pieds de page était automatique; ce qui est manifestement faux.

J'en ai donc déduit qu'il y avait un petit conflit entre les extensions `vmargin` et `fancyhdr`. Cette déduction est devenue certitude quand j'ai vu des documents L<sup>A</sup>T<sub>E</sub>X n'utilisant pas l'extension `vmargin`.

Fallait-il renoncer au package `vmargin`?

Certes il existe d'autres façons de régler les marges d'un document ; le package `geometry` fait ça très bien également.

Mais non !

Je me suis donc replongé dans la documentation de `fancyhdr` et j'ai trouvé la solution !

La largeur de l'en-tête et du pied de page (c'est la même !) est gérée par la variable `\headwidth` ; il suffit donc de demander que cette variable prenne la valeur de la largeur du texte en entrant la ligne :

```
\renewcommand{\headwidth}{\textwidth}
```

Ainsi, quelle que soit la largeur du texte, en-tête et pied de page auront la même largeur que celle de ce texte.

### 3.4 En guise de conclusion

Pour terminer cette chronique, je vous livre quels sont les en-têtes et pieds de page de ces pages :

```
\usepackage{fancyhdr}                % En-têtes et pieds de pages
\pagestyle{fancy}
\renewcommand\headrulewidth{1pt}      % 0,4 pt par défaut
%\renewcommand\footrulewidth{1pt}     % 0 pt par défaut
\renewcommand{\headwidth}{\textwidth} % ajustement largeur
\fancyhead[RO,LE]{http://latexetmath.canalblog.com}
\fancyhead[LO,RE]{FH}                 % even : pair et odd : impair
\fancyhead[C]{}
\fancyfoot[RO]{\large\smiley\ \normalsize{--\ \thepage}}
\fancyfoot[LE]{ \normalsize{\thepage\ --\ }\large\smiley}}
\fancyfoot[C]{}

```

Le `\smiley` se trouve dans le package `wasysym`.

Et vous aurez compris qu'il suffit de retirer le % devant la ligne qui en contient un pour tracer un filet de bas de page de largeur 1 point.

Une remarque pour terminer : on n'est pas obligé de gérer dans chaque définition de `fancy` les six paramètres `fancyhead[L]{} , fancyhead[C]{} , fancyhead[R]{} , fancyfoot[L]{} , fancyfoot[C]{}` et `fancyfoot[R]{} ;` certains sont prédéfinis et donnent une valeur par défaut (le numéro de page, par exemple), d'autres sont optionnels.

Comme je ne sais jamais lesquels font quoi, je mets systématiquement les six dans chacune des définitions de mes en-têtes et pieds de page.

Je sais donc exactement où j'en suis et ce qui va se passer !

## 12 – Courbes de Gauss

On a vu dans la chronique « Courbes en PsTricks » comment tracer des courbes (voir page 13); on y fera constamment référence dans cette chronique-ci.

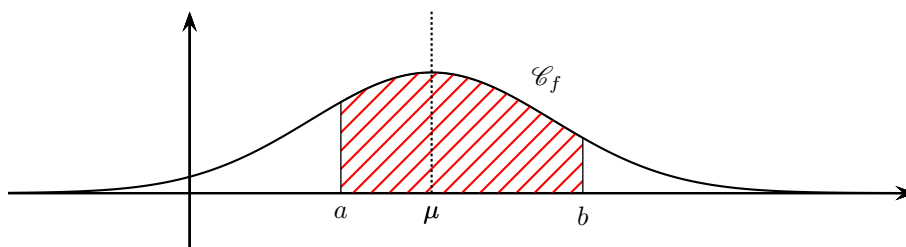
### 1 Notation post-fixée ou notation infixée

Dans la précédente chronique, la plupart des fonctions représentées étaient définies en notation **post-fixée**; c'est à dire qu'au lieu d'écrire  $2x-3$  (notation **infixée**), on écrivait `2 x mul 3 sub`. Cette **Notation Polonaise Inverse** (NPI ou RPN en anglais) a été inventée en 1920 par JAN LUKASIEWICZ, et c'est CHARLES LEONARD HAMBLIN qui l'a popularisée dans les années 50. Mais si on exporte du code L<sup>A</sup>T<sub>E</sub>X depuis GeoGebra, par exemple, on obtient des représentations graphiques de fonctions écrites en notation **infixée**; et on peut constater que la commande `algebraic=true` a été ajoutée comme option dans `\psset` (voir page 18).

On va voir qu'il est parfois plus pratique d'utiliser la notation infixée que la notation post-fixée.

### 2 Objectif

L'objectif de cette chronique est de construire un graphique comme celui-ci où la courbe est la représentation graphique de la fonction  $f : x \mapsto \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$  avec  $\mu$  réel quelconque et  $\sigma$  réel strictement positif.



### 3 La courbe

On va représenter la fonction  $x \mapsto \frac{1}{2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-4}{2}\right)^2}$  qui est la fonction de densité d'une variable aléatoire  $X$  suivant la loi normale de moyenne  $\mu = 4$  et d'écart type  $\sigma = 2$ .

On commence par définir ce qu'il faut pour tracer la courbe en PsTricks avec `\psset`, `\pspicture`, `\psaxes`, etc., puis on définit la fonction de densité en notation infixée :

```
\def\fd{1/(2*sqrt(2*PI))*EXP(-((x-4)/2)^2/2)}
```

Enfin on demande son tracé; 1000 points pour  $x$  variant entre `\xmin` et `\xmax` qu'il faut définir avant le `\pspicture` :

```
\psplot[plotpoints=1000]{\xmin}{\xmax}{\fd}
```

On tracera une courbe avec des axes sur lesquels on ne mettra pas de graduation (`ticks=none`), ni de label (`labels=none`).

Ce qui donne comme code :

```
\psset{xunit=0.8cm, yunit=8cm, runit=1cm, arrowsize=3pt 3, algebraic=true}
\def\xmin {-3} \def\xmax {12} % intervalle des x
\def\ymin {-0.2} \def\ymax {0.3} % intervalle des y
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
\psaxes[ticks=none, labels=none]{->}(0,0)(\xmin,-0.1)(\xmax,\ymax)
\def\f{1/(2*sqrt(2*PI))*EXP(-((x-4)/2)^2/2)}
\psplot[plotpoints=1000]{\xmin}{\xmax}{\f}
\end{pspicture*}
```

## 4 Amélioration

Ce serait pratique de pouvoir changer la moyenne et l'écart type de façon à pouvoir rapidement tracer d'autres fonctions de densité de lois normales.

Facile... à condition d'utiliser la notation infixée car je ne suis pas arrivé à le faire avec la notation post-fixée!

Il suffit de définir la moyenne `\m` et l'écart type `\s` puis d'utiliser ces variables dans la définition de la fonction :

```
...
\def\m{4} \def\s{2} % moyenne et écart type
\def\f{1/(\s*sqrt(2*PI))*EXP(-((x-\m)/\s)^2/2)} % définition de f
\psplot[plotpoints=1000]{\xmin}{\xmax}{\f} % tracé de la fonction
...
```

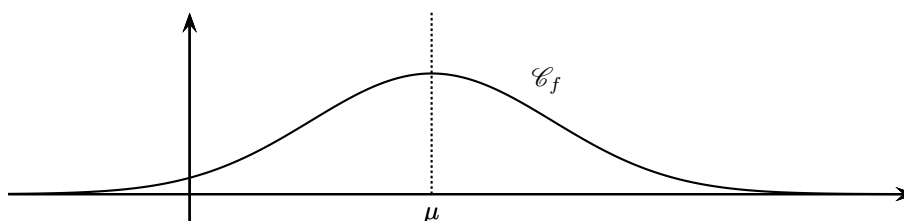
Naturellement, on ajustera en conséquence les valeurs de `\xmin`, `\xmax`, `\ymin` et `\ymax`, ainsi que celles de `xunit` et `yunit` si nécessaire.

Enfin on peut placer la lettre  $\mu$  là où il faut, et tracer en pointillés la droite d'équation  $x = \mu$ , axe de symétrie de la courbe :

```
...
\uput[d](\m,0){$\mu$}
\psline[linestyle=dashed, dash=1pt 1pt](\m,0)(\m,\ymax)
...
```

Le mode « pointillés » s'appelle `dotted` et pour tracer un trait en pointillés, il faudrait entrer comme option `linestyle=dotted`; mais j'ai trouvé que les pointillés étaient peu visibles, alors j'utilise le mode `dashed` (tirets) en rapprochant suffisamment les tirets pour qu'ils apparaissent comme des pointillés...

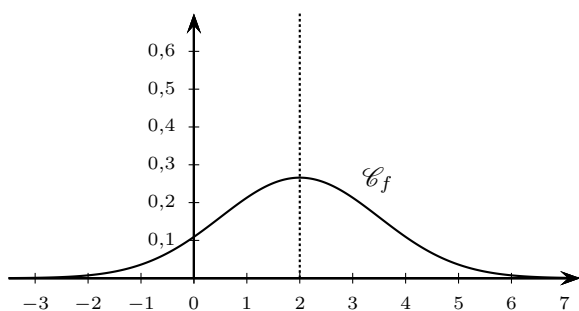
Ce qui donne, en plaçant le nom de la courbe au bon endroit :



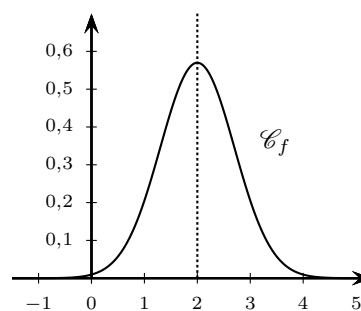


## 5 Influence des paramètres

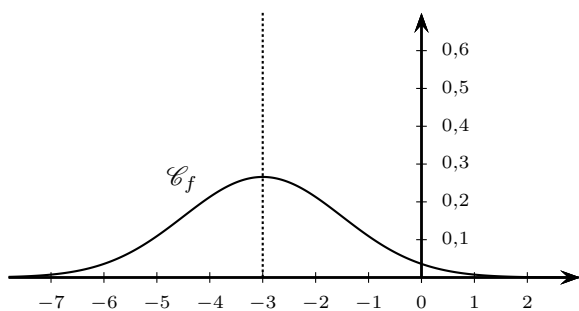
Maintenant que tout ou presque est défini, c'est très facile de montrer l'influence du couple de paramètres (moyenne, écart type) sur des tracés de courbes de Gauss :



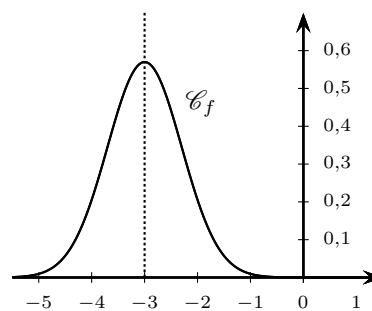
$\mu = 2$  et  $\sigma = 1,5$



$\mu = 2$  et  $\sigma = 0,7$



$\mu = -3$  et  $\sigma = 1,5$



$\mu = -3$  et  $\sigma = 0,7$

Quatre `\pspicture` et quatre minipages et le tour est joué!

Voici quand même le code de la représentation correspondant à  $\mu = 2$  et  $\sigma = 1,5$  :

```
\psset{xunit=0.7cm, yunit=5cm, runit=1cm, arrowsize=3pt 3, algebraic=true}
\def\xmin {-3.5} \def\xmax {7.3}
\def\ymin {-0.1} \def\ymax {0.7}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
\psaxes[labelFontSize=\scriptstyle, ticksize=-2pt 2pt, Dy=0.1, comma]%
{->}(0,0)(\xmin,0)(\xmax,\ymax)
\def\m{2} \def\s{1.5} % moyenne et écart type
\def\ff{1/(\s*sqrt(2*PI))*EXP(-((x-\m)/\s)^2/2)}
\psplot[plotpoints=1000]{\xmin}{\xmax}{\ff}
\psline[linestyle=dashed,dash=1pt 1pt](\m,0)(\m,\ymax) % axe de symétrie
\uput[ur](3,0.2){$\Cr f$}
\end{pspicture*}
```

Dans `\psaxes`, on voit pour options :

- `labelFontSize=\scriptstyle` qui permet d'écrire les labels en plus petit ;
- `Dy=0.1` qui gradue les ordonnées par dixième (par défaut `Dx=1` donc on peut ne pas le mettre) ;
- `comma` qui permet d'afficher la virgule comme séparateur décimal dans les labels à la place du point.

Mais il y a une particularité intéressante dans les deux courbes du bas : les labels en ordonnée sont situés à droite de l'axe.

J'ai mis un moment à trouver comment faire!

Je me suis plongé dans la documentation de `pstricks` et j'ai découvert la variable `labelsep` qui gère l'espacement entre l'axe et le label; sa valeur par défaut est 0,5 pt. J'ai donc essayé, pour écrire les labels de l'autre côté de l'axe, d'entrer une valeur négative pour `labelsep`. Mais hélas, un `labelsep=-20pt`, place de l'autre côté de l'axe à la fois les labels en ordonnée et... ceux en abscisse.

Ce qui n'était pas envisageable!

La solution?

J'ai essayé de voir si, par hasard, la variable `ylabelsep` n'existait pas, et si on ne pouvait pas la modifier, et j'ai obtenu exactement ce que je souhaitais!!!

Il faut donc rajouter comme option dans `\psaxes` : `ylabelsep=-20pt`  
et les labels seront sur le côté droit de l'axe des ordonnées.

## 6 Domaine sous la courbe

On veut maintenant hachurer le domaine compris entre les droites d'équations  $x = a$  et  $x = b$ , la courbe et l'axe des abscisses.

J'ai d'abord commencé par des `\psline` comme il est décrit à cette adresse :

[http://ww2.ac-poitiers.fr/math/spip.php?article134&debut\\_page=2#sommaire\\_1](http://ww2.ac-poitiers.fr/math/spip.php?article134&debut_page=2#sommaire_1)

Cette méthode utilise `\pscustom` mais nécessite de connaître  $f(a)$  et  $f(b)$ , ce qui oblige à les calculer avant.

Il y a mieux en remplaçant les `\psline` par des `\lineto`, sans oublier de terminer la commande par un `\closepath` qui « ferme le chemin »; voilà le code complet, dans lequel on voit qu'on va hachurer en rouge le domaine sous la courbe (options de `\pscustom`) :

```
\psset{xunit=0.8cm, yunit=8cm, runit=1cm, arrowsize=3pt 3, algebraic=true}
\def\xmin {-3} \def\xmax {12}
\def\ymin {-0.2} \def\ymax {0.3}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
\psaxes[labelFontSize=\small, ticksize=-0pt 0pt, Dx=1, Dy=0.1, labels=none]%
{->}(0,0)(\xmin,-0.1)(\xmax,\ymax)
\def\m{4} \def\s{2} % moyenne et écart type
\def\f{1/(\s*sqrt(2*PI))*EXP(-((x-\m)/\s)^2/2)} % définition de la fonction
\psplot[plotpoints=1000]{\xmin}{\xmax}{\f} % tracé de la fonction

\def\inf{2.5} \def\sup{6.5} % définit a et b
\pscustom[fillstyle=hlines, linestyle=solid, linewidth=0.5pt, hatchcolor=red]
{
\psplot{\inf}{\sup}{\f} % courbe de f sur [inf , sup]
\lineto(\sup,0)\lineto(\inf,0) % trace les segments
\closepath % indispensable pour fermer le chemin
}

\uput[d](\m,0){$\mu$} \psline[linestyle=dashed, dash=1pt 1pt](\m,0)(\m,\ymax)
\uput[ur](5.5,0.15){$\Cr f$}
\uput[d](4,0){$\mu$}
\uput[d](\inf,0){$a$} \uput[d](\sup,0){$b$}
\end{pspicture*}
```

L'utilisation de `\pscustom` est décrite dans le mode d'emploi de `pstricks` qui a été traduit et mis en ligne sur le site de l'APMEP :

<http://www.apmep.asso.fr/IMG/pdf/TraductionfinalePSTricks.pdf>

Merci à la personne qui a traduit ce mode d'emploi.

## Sommaire

<b>1 – Quelques principes de base</b>	<b>3</b>
1 Généralités	3
2 Notion d’environnement	3
3 Structure d’un document	4
4 Conseil pour démarrer	5
<b>2 – Coordonnées</b>	<b>7</b>
1 Première méthode avec <code>pmatrix</code>	7
2 Deuxième méthode avec <code>array</code>	7
3 Autres environnements	8
4 Automatisation	8
<b>3 – Matrices</b>	<b>11</b>
1 Sans fioritures	11
2 Première méthode – Simple et efficace	11
3 Deuxième méthode – À connaître	12
4 Troisième méthode – Le contrôle complet	12
<b>4 – Courbes en PsTricks</b>	<b>13</b>
1 Quelques préliminaires	13
2 Création du repère	14
3 Petite amélioration	15
4 Le tracé de courbes	16
5 Autre exemple	18
<b>5 – Multicolonnage et minipage</b>	<b>19</b>
1 Multicolonnage	19
1.1 Le principe . . . . .	19
1.2 Quelques améliorations . . . . .	19
1.3 Un exemple . . . . .	20
1.4 Autre exemple . . . . .	20

1.5	Limite du multicolonnage . . . . .	21
<b>2</b>	<b>Minipage</b>	<b>21</b>
<b>6</b>	<b>Solides de l'espace</b>	<b>23</b>
1	Cube	23
2	Parallélépipède	24
3	Repère oblique	25
4	Tétraèdre	26
<b>7</b>	<b>Compléments sur les minipages</b>	<b>27</b>
1	Largeur	27
2	Positionnement sur la ligne	27
3	Encadrement	28
4	Hauteur	28
5	Positionnement intérieur	29
<b>8</b>	<b>Trucs mathématiques</b>	<b>31</b>
1	Fonctions et parenthèses	31
1.1	Première amélioration . . . . .	31
1.2	Deuxième amélioration . . . . .	31
2	Personnalisation du mode mathématique	32
2.1	Couleur . . . . .	32
2.2	Gras . . . . .	32
3	Accolade et superposition de texte	33
4	Crochets	33
5	Alignement	34
<b>9</b>	<b>Tableaux</b>	<b>35</b>
1	Tableau de valeurs	35
2	Fusion de cellules horizontales	36
3	Tableau de signes	38
<b>10</b>	<b>Tableaux - Suite</b>	<b>39</b>

<b>1</b>	<b>Systèmes et accolades</b>	<b>39</b>
<b>2</b>	<b>Tableau de variations</b>	<b>40</b>
<b>3</b>	<b>Trait en diagonale</b>	<b>41</b>
<b>4</b>	<b>Largeur fixe</b>	<b>42</b>
<b>5</b>	<b>Épaisseur du trait</b>	<b>42</b>
<b>11</b>	<b>– Mise en page</b>	<b>43</b>
<b>1</b>	<b>Marges d'un document</b>	<b>43</b>
<b>2</b>	<b>Augmentation des marges d'un paragraphe</b>	<b>43</b>
<b>3</b>	<b>En-tête et pied de page</b>	<b>44</b>
3.1	Mode normal . . . . .	44
3.2	Mode recto verso . . . . .	45
3.3	Ajustement à la largeur de la marge . . . . .	45
3.4	En guise de conclusion . . . . .	46
<b>12</b>	<b>– Courbes de Gauss</b>	<b>47</b>
<b>1</b>	<b>Notation post-fixée ou notation infixée</b>	<b>47</b>
<b>2</b>	<b>Objectif</b>	<b>47</b>
<b>3</b>	<b>La courbe</b>	<b>47</b>
<b>4</b>	<b>Amélioration</b>	<b>48</b>
<b>5</b>	<b>Influence des paramètres</b>	<b>49</b>
<b>6</b>	<b>Domaine sous la courbe</b>	<b>50</b>
	<b>Sommaire</b>	<b>51</b>