Mathématiques

et

LaTeX

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 TEX et a créé ce « traitement de textes mathématiques ». Le LATEX a été développé à partir du TEX par Leslie Lamport.

Un document LATEX (on prononce TEX 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 `USB`TEX ; lors de son implantation, ce système installait tous les logiciels utiles et les chemins (`path`) nécessaires au bon fonctionnement de l'ensemble.
`USB`TEX 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 `USB`TEX.
Comme `USB`TEX 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 LATEX est la notion d'environnement ; un environnement commence par un `\begin{...}` et se termine par `\end{...}` :

```
\begin{environnement}
...
\end{environnement}
```

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

```
\begin{environnement 1}
...
  \begin{environnement 2}
  ...
  \end{environnement 2}
...
\end{environnement 1}
```

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 LaTeX doit débuter par la déclaration de la classe de document que l'on va écrire :

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

Beaucoup d'instructions en 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 LaTeX, tout ce qui est optionnel s'écrit entre crochets `[` et `]` ; 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 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 LaTeX ne connaît pas les caractères accentués ; il faut donc charger des extensions (`packages` en anglais) qui vont dire à LaTeX de reconnaître les caractères accentués. On écrira donc :

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

Un texte en LaTeX a donc une structure de la forme :

Définition de classe	`\documentclass[...]{...}`
Chargement des extensions	`\usepackage[...]{...}` `\usepackage[...]{...}` ...

Définitions personnelles	`\newcommand{...}{...}`
	`\renewcommand{...}{...}`
	...

Corps du texte	`\begin{document}`
	...
	...
	`\end{document}`

4 Conseil pour démarrer

Après avoir installé un système 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 LaTeX.

Bonnes rédactions en 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 préfèreront un trait vertical : $A\left|\begin{array}{c}3\\-2\end{array}\right.$.
Il existe en LaTeX 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 $A\left|\begin{array}{c}3\\-2\end{array}\right.$ 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 : `\,` `\;` `\␣` (attention : le ␣ signifie « appui sur la barre d'espacement » !)

Donc en tapant
```
$A\;
\begin{array}{|c}
3 \\
-2
\end{array}$
```
on obtient $A\left|\begin{array}{c}3\\-2\end{array}\right.$

3 Autres environnements

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

En tapant
```
$\begin{matrix}
3 \\
-2
\end{matrix}$
```
on obtient $\begin{matrix}3\\-2\end{matrix}$

En tapant
```
$\begin{bmatrix}
3 \\
-2
\end{bmatrix}$
```
on obtient $\begin{bmatrix}3\\-2\end{bmatrix}$

En tapant
```
$\begin{vmatrix}
3 \\
-2
\end{vmatrix}$
```
on obtient $\begin{vmatrix}3\\-2\end{vmatrix}$

En tapant
```
$\begin{Vmatrix}
3 \\
-2
\end{Vmatrix}$
```
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
```
$\begin{vmatrix}
3 & -5 & 9\\
-2 & 8 & 7\\
1 & -6 & 11
\end{vmatrix}$
```

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 LaTeX !!!).
Mais non !
Justement 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 $\mathbf{R}$ (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 :

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

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

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 !

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

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

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 apparaitraient dans la suite du document, on remet ce facteur à 1 après avoir quitté l'environnement `pmatrix` :

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

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

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

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

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

3 Deuxième méthode – À connaître

Comme, dans un tableau on passe d'une ligne à la ligne suivante en entrant la commande de passage à la ligne \\, il suffit de dire à LaTeX de passer à la ligne en augmentant l'interligne ; par exemple si on veut l'augmenter de 7 points, on entre `\\[7pt]` à la place de `\\`.

```
$\begin{pmatrix}
\dfrac{1}{2} & \dfrac{1}{3} & \dfrac{1}{4} \\[7pt]
\dfrac{1}{5} & \dfrac{1}{6} & \dfrac{1}{7} \\[7pt]
\dfrac{1}{8} & \dfrac{1}{9} & \dfrac{1}{2}
\end{pmatrix}$
```

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

Il faut connaître cette méthode, car elle reste valable dans tous les environnements.

4 Troisième méthode – Le contrôle complet

On peut également insérer sur chaque ligne une « béquille » de largeur nulle dont on peut contrôler le décalage par rapport à l'axe de la ligne ainsi que la hauteur :

`\rule[-12pt]{0pt}{30pt}`

Cette « béquille » partira de 12 points en dessous de l'axe normal de la ligne, aura une épaisseur de 0 point et une hauteur totale de 30 points.
Le décalage négatif par rapport à l'axe normal de la ligne – `[-12pt]` – est écrit entre crochets parce que c'est un paramètre facultatif (qui vaut 0 si on n'entre rien).

```
$\begin{pmatrix}
\dfrac{1}{2} & \dfrac{1}{3} & \dfrac{1}{4} \rule[-12pt]{0pt}{30pt}\\
\dfrac{1}{5} & \dfrac{1}{6} & \dfrac{1}{7} \rule[-12pt]{0pt}{30pt}\\
\dfrac{1}{8} & \dfrac{1}{9} & \dfrac{1}{2} \rule[-12pt]{0pt}{30pt}
\end{pmatrix}$
```

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

Cette méthode est en fait un détournement de fonction de l'instruction `\rule` qui est destinée à tracer des traits ; mais c'est celle qui, à mon avis, donne le plus joli résultat.

Et on peut utiliser cette méthode, comme les autres, pour écrire un déterminant en utilisant l'environnement `vmatrix` :

```
$\begin{vmatrix}
\dfrac{1}{2} & \dfrac{1}{3}  \rule[-12pt]{0pt}{30pt}\\
\dfrac{1}{5} & \dfrac{1}{6}  \rule[-12pt]{0pt}{30pt}
\end{vmatrix}$
```

$$\begin{vmatrix} \dfrac{1}{2} & \dfrac{1}{3} \\ \dfrac{1}{5} & \dfrac{1}{6} \end{vmatrix}$$

4 – Courbes en PsTricks

On peut, en 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 ⟶

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 $(O\,;\vec{\imath},\vec{\jmath})$ ou $(O\,;\vec{u},\vec{v})$;
- des points I et J, ou des vecteurs $\vec{\imath}$ et $\vec{\jmath}$, 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[dl](0,0){$O$}`.
Le `dl` 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°` et le `u` correspond à `90°` ; 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 $\vec{\imath}$ et $\vec{\jmath}$, ou $\vec{u}$ et $\vec{v}$ selon le cas :

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

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 $\vec{\imath}$ et $\vec{\jmath}$.

Naturellement, si on a des repères $(O\,;\vec{\imath},\vec{\jmath})$ 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 l'est peut-être moins). C'est un choix ! On peut les écrire en romain...

Voici donc ce qu'il faut entrer pour un repère $(O\,;\vec{\imath},\vec{\jmath})$, 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[ticksize=-2pt 2pt, labels=none](0,0)(-3,-5)(4,6)
\uput[dl](0,0){$O$}
\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[ticksize=-2pt 2pt, labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[dl](0,0){$O$}
\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 $f : x \longmapsto 2x^2 - 5x - 1$, de la droite d d'équation $y = -2x + 1$ et on va placer leurs points d'intersection A et B.
L'écriture $f : x \longmapsto 2x^2 - 5x - 1$ s'obtient en entrant `$f: x \longmapsto 2x^2-5x-1$`.
Ceux qui préfèrent $x \overset{f}{\longmapsto} 2x^2 - 5x - 1$ 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[ticksize=-2pt 2pt, labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[dl](0,0){$O$}
\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 $f\,(x) = 2x^2 - 5x - 1$, $f\,(\texttt{xmin}) = f\,(-3) = 32$ 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 18) ou pour tracer une aire sous une courbe.

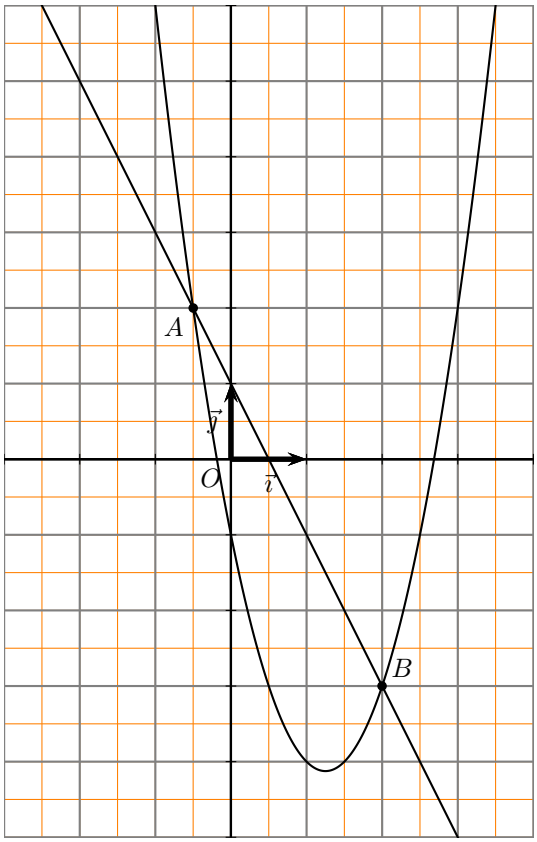
On définit donc la fonction f par `\def\f{2 x x 2.5 sub mul mul 1 sub}`,
la fonction affine par `\def\g{-2 x mul 1 add}`.

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[dl](-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[ticksize=-2pt 2pt, labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[dl](0,0){$O$}
\psaxes[linewidth=1.8pt]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$} \uput[l](0,0.5){$\vec{\jmath}$}
\def\f{2 x x 2.5 sub mul mul 1 sub}
\def\g{ -2 x mul 1 add}
\psplot[plotpoints=1000]{\xmin}{\xmax}{\f}
\psplot[plotpoints=1000]{\xmin}{\xmax}{\g}
\psdots(-0.5,2)(2,-3)
\uput[dl](-0.5,2){$A$}
\uput[ur](2,-3){$B$}
\end{pspicture*}
```

Ce qui donne :

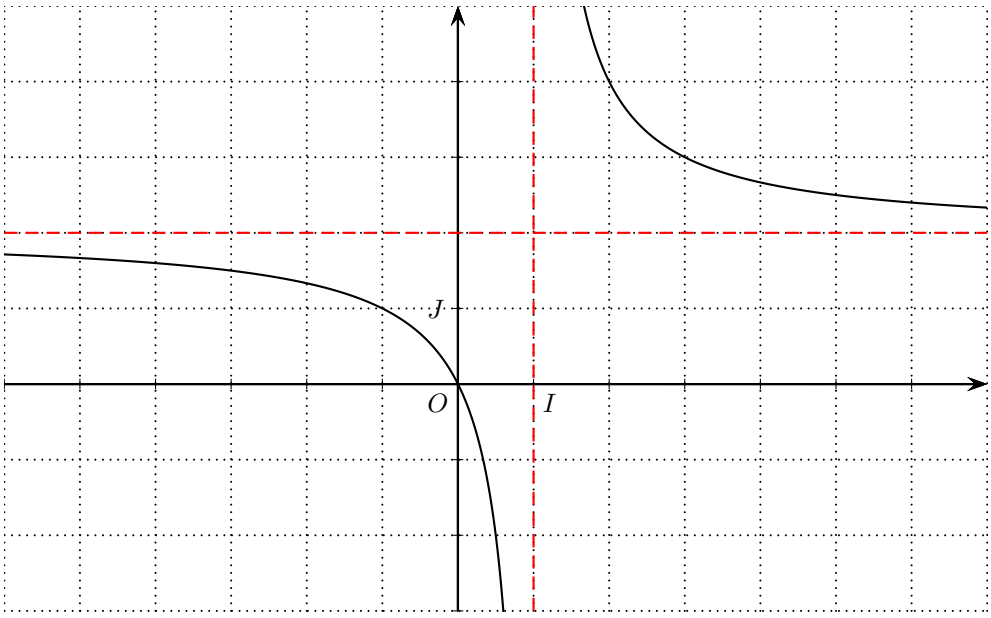


5 Autre exemple

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 $f(x) = \dfrac{2x}{x-1}$, 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){$O$}
%\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\f{2*x/(x-1)}                              % définition de la fonction
\psplot[plotpoints=1000]{\xmin}{0.99}{\f}   % une branche de l'hyperbole
\psplot[plotpoints=1000]{1.01}{\xmax}{\f}   % 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 Exemple de texte sur deux colonnes

Le texte est réparti automatiquement sur les deux colonnes.

1.2 Quelques améliorations

On peut agrandir l'espacement 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 largeur 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 Texte avec changement de colonne forcé

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

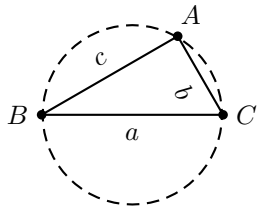
1.3 Un exemple

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.

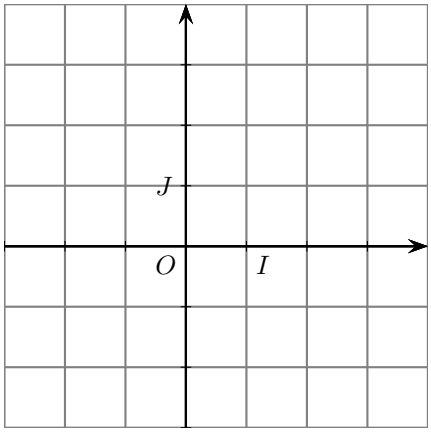
D'après le théorème de Pythagore, on peut écrire que : $BC^2 = AC^2 + AB^2$.
Si on pose $AB = c$, $AC = b$ et $BC = a$, alors $BC^2 = AC^2 + AB^2 \Leftrightarrow a^2 = b^2 + c^2$.



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[ticksize=-2pt 2pt, labels=none]{->}(0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[dl](0,0){$O$}
\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.

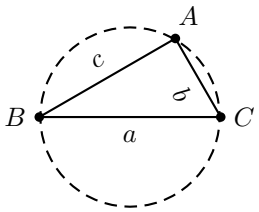
Naturellement, on peut créer plus de deux minipages l'une à côté de l'autre ; il faut quand même que la somme des largeurs des minipages ne dépasse pas la largeur de la page (sauf si on veut dépasser dans la marge droite).

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 :

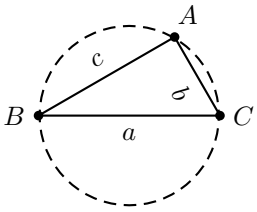
```
\begin{minipage}{0.7\linewidth}
Le triangle $ABC$ est inscrit dans le cercle de diamètre $[BC]$.\\

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.

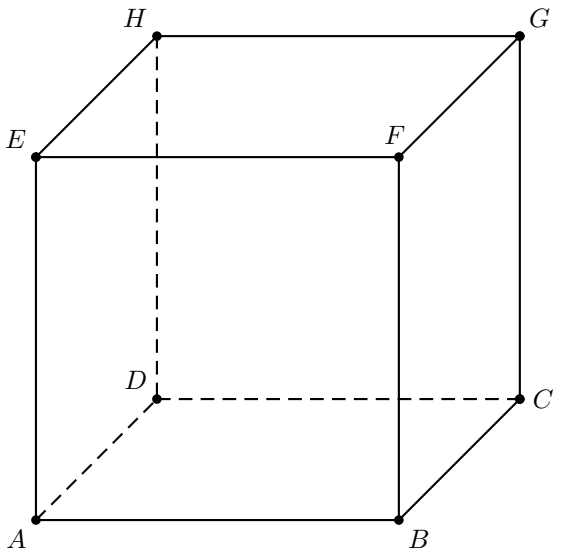
1 Cube

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[dl](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}
```

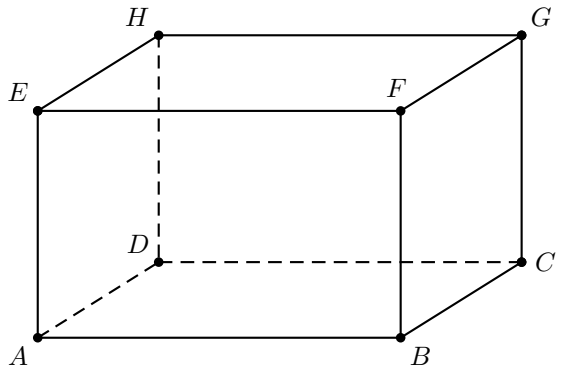
L'instruction `\pspolygon`, comme son nom l'indique, trace un polygone fermé.
L'option `[showpoints=true]` dessine les points aux sommets du polygone.
Par défaut, les points sont représentés par des petits disques dont on peut modifier la taille ; si on veut changer l'aspect des points, on peut entrer 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 ».

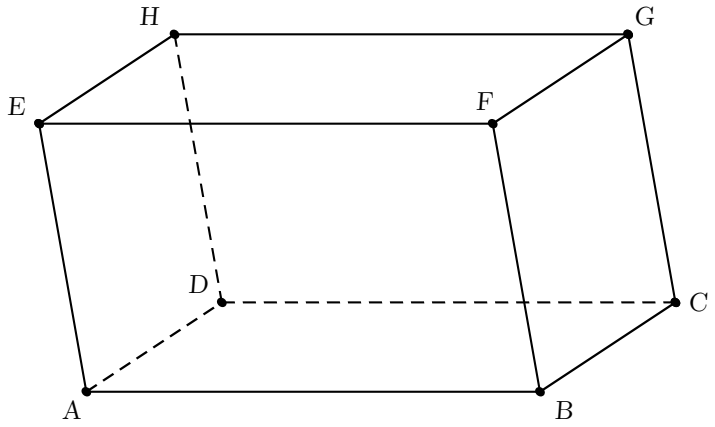
En tapant `\pstilt{60}{LaTeX}` on obtient *LaTeX* (angle de 60°).
En tapant `\pstilt{90}{LaTeX}` on obtient LaTeX (aucun changement !).
En tapant `\pstilt{120}{LaTeX}` on obtient LaTeX (angle de 120°).

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[dl](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}
```

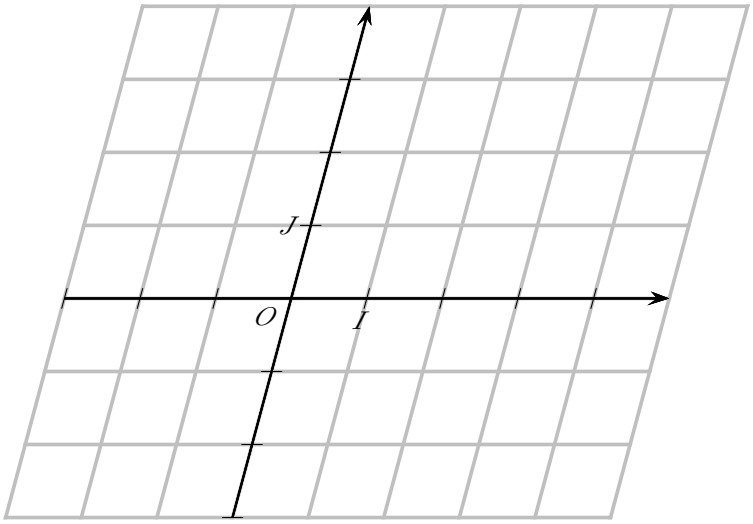
on obtient :



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.

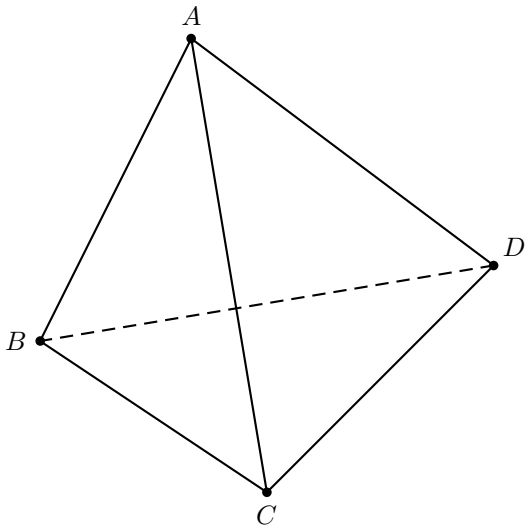


qui s'obtient en entrant :

```
\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[l](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 LaTeX, $0,7 + 0,3$ n'est pas toujours égal à 1.
En effet, si on met côte à côte deux minipages de largeurs `0.7\textwidth` et `0.3\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

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\textwidth` pour la première minipage, et `0.29\textwidth` au lieu de `0.3\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

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 essai essai essai essai essai essai essai essai essai essai essai essai essai

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

Le paramètre par défaut est `[c]`.

3 Encadrement

En 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 $S=\left\lbrace 2\,; 5 \right\rbrace$}` 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 essai essai 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 ?

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

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

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

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

Il suffit donc de taper `\(` à la place de `\left(`, et `\)` à la place de `\right)`.
Pourquoi `\renewcommand` et pas `\newcommand` ?
Tout simplement parce que les commandes `\(` et `\)` existent déjà (à essayer !).
Une fois ces commandes définies dans le préambule du document,

on tape	`$f\(x\)$`	pour obtenir	$f\,(x)$
	`$g\(\dfrac{3}{4}\)$`		$g\left(\dfrac{3}{4}\right)$
	`$A\(\dfrac{4}{3}\,, 2\)$`		$A\left(\dfrac{4}{3}\,, 2\right)$

1.2 Deuxième amélioration

Il n'en reste pas moins que taper `$f\(x\)$` 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 $f\,(x) = ...$ 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 $\boldsymbol{f\,(x) = x^2}$

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 $f\,(x) = \boldsymbol{x^2}$

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 $\boldsymbol{f}$ la fonction définie sur $\boldsymbol{[0\,,+\infty[}$ par $\boldsymbol{f\,(x) = \dfrac{x-1}{x^2+2}}$.

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 :

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

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 :

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

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

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

pour obtenir : $a^n = \underbrace{a \times a \times ... \times a}_{n \text{ facteurs}}$

`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 :

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

Dans le même esprit, on peut écrire un nombre en binaire $\overline{1011\,0011}^2$ 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 : $\left[0\,,1\right]$.

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 $\left[0\,,1\right]$ 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 $[\![0\,,12]\!]$ en utilisant le même procédé.

Il suffit pour cela de créér 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 $[\![0\,,12]\!]$.

Enfin, si on veut des crochets qui s'adaptent à la hauteur du texte, on utilisera `\left[` et `\right]` comme dans $\left[\dfrac{1}{2}\,,\dfrac{3}{4}\right]$ 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 :
$f'(x)=2x\left(\mathrm{e}^{x-1}-x-1\right)+x^2\left(\mathrm{e}^{x-1}-1\right)=2x\,\mathrm{e}^{x-1}-2x^2-2x+x^2\,\mathrm{e}^{x-1}-x^2=\mathrm{e}^{x-1}\left(2x+x^2\right)-x\,(3x+2)$

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

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

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.