Chronique 6

Petits trucs mathématiques

6.1 Gras ou très gras

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

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

x	$\boldsymbol{-\infty}$		3		$\boldsymbol{+\infty}$
$3-x$		$+$	0	$-$	

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

`$+\infty$`	`{\boldmath $+\infty$}` ou `$\boldsymbol{+\infty}$`	`$\pmb{+\infty}$`
$+\infty$	$+\infty$	$+\infty$

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

6.2 Environnement `cases`

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

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

$$f(x)=\left\{\begin{array}{r l} 4x+1 & \text{si } x<1\\ 5 & \text{si } x=1\\ -x+6 & \text{si } x>1 \end{array}\right.$$

Il existe un environnement qui permet d'écrire ça de façon un peu plus rapide : c'est `cases` qui est intégré au package `amsmath`.
Inutile de définir l'accolade, et le tableau à deux colonnes est défini directement :

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

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

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

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

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

6.3 Virgule en mode mathématique

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

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

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

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

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

Et comment j'ai fait pour écrire $12,345$ avec l'espace ?
En entrant `{\StandardMathComma $12,345$}` car `\StandardMathComma` va contrecarrer l'action de `\DecimalMathComma`.

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

`$\cd a,b \cg$` donne	`$\cd a\,,\,b \cg$` donne
$[a,b]$	$[a\,,b]$

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

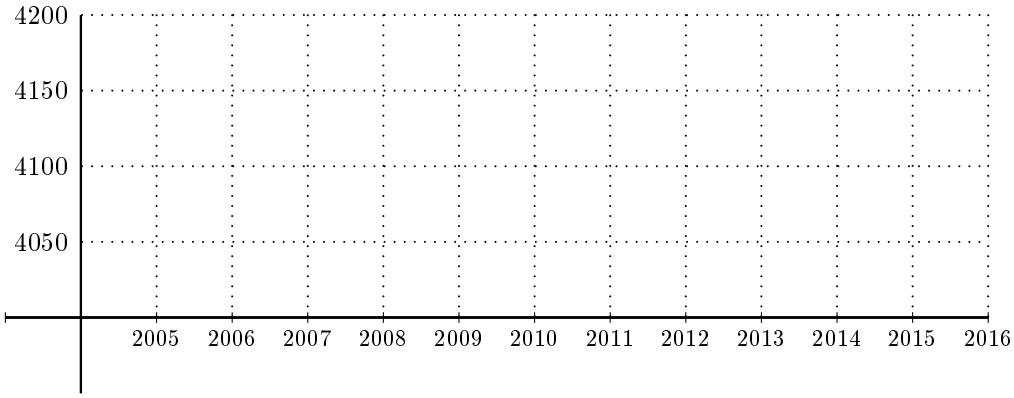
```
\newcommand{\cg}{\rceil \hspace{-4.5pt} \rfloor}
\newcommand{\cd}{\lceil \hspace{-4.5pt} \lfloor}
```

Elles permettent de définir des crochets plus visibles.

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

6.4 Repère

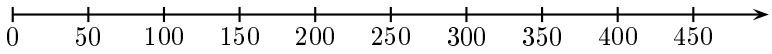
On a parfois besoin de tracer des repères dans lesquels les légendes n'ont pas grand-chose à voir avec les unités, notamment si on veut représenter des ajustements :



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

6.4.1 Premier exemple

Commençons par quelque chose de simple :

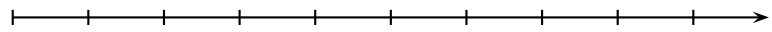


Il s'agit d'un axe horizontal régulièrement gradué tous les centimètres ; la graduation va de 0 à 450 par pas de 50 et il y a 10 graduations.
On va d'abord régler les unités en abscisse : si on veut que la grandeur 50 soit représentée par 1 cm, il faut prendre pour unité 0,02 cm : c'est ce que fait l'affectation `xunit=0,02cm` dans `\psset`.
Ensuite on va définir une variable entière `\i` (`i` comme `integer`) qui va prendre 10 valeurs de 0 à 450 par pas de 50. Puis on va créer une boucle utilisant cette variable comme compteur au moyen de l'instruction `\multido` : `\multido{\i=0+50}{10}{...}`

La variable `\i` va prendre successivement les valeurs 0, 50, 100, 150, 200, 250, 300, 350, 400 et 450 (10 valeurs en tout, c'est le deuxième paramètre).
On mettra ce qu'il faut répéter à la place des pointillés comme troisième paramètre.

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

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

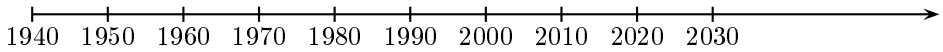


Il ne reste plus qu'à écrire les légendes en dessous des segments tracés, ce que l'on réalise avec des `\uput` bien placés :

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

6.4.2 Deuxième exemple

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



Il s'agit d'un axe gradué tous les centimètres avec des légendes représentant des dates en partant de 1940 par pas de 10. On va donc définir une variable `\i` qui va servir de compteur et qui va permettre de tracer les traits verticaux, et une autre variable `\n` pour écrire les légendes.
La seule contrainte est qu'on doit avoir le même nombre de valeurs pour chaque variable, donc 10.
Voici le code :

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

Deux remarques :

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

6.4.3 Le code du repère

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

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

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