Chronique 1

Couleurs en LaTeX

Comment gérer les couleurs en LaTeX ?
Voici un sujet que l'on va traiter ici de manière presque exhaustive.

1.1 Un peu de théorie

1.1.1 Synthèse additive

En synthèse additive, on part du noir (absence de toute couleur) puis on ajoute soit du rouge, soit du vert, soit du bleu ou toute combinaison de ces trois couleurs.
L'addition du vert et du bleu donne le cyan, celle du rouge et du bleu donne le magenta, celle du rouge et du vert donne le jaune. Enfin l'addition du rouge, du vert et du bleu donne le blanc.

1.1.2 Synthèse soustractive

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

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

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

1.1.3 Couleurs primaires

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

1.2 Package `color`

Tout d'abord, on charge l'extension `color` en entrant `\usepackage{color}` dans le préambule du document, si ce n'est pas déjà fait.
Ou pas ! comme on verra au début du paragraphe suivant...

1.2.1 Couleurs prédéfinies

Le package `color` définit six couleurs :

red	rouge	green	vert	blue	bleu
cyan	cyan	magenta	magenta	yellow	jaune

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

1.2.2 Instructions

`\color`

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

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

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

`\textcolor`

Quand on n'a qu'un mot ou un bout de phrase à écrire dans une autre couleur, par exemple en rouge, on utilise plutôt `\textcolor` comme instruction.
Voici ce qu'il faut taper pour obtenir cette phrase :

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

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

`\pagecolor`

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

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

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

`\definecolor`

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

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

`gray`

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

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

Le `coefficient` est un nombre décimal compris entre `0` et `1` qui indique le pourcentage de blanc dans le noir ; ainsi le coefficient `0` donnera du noir, et le coefficient `1` donnera du blanc.
Par exemple `\definecolor{grisfoncé}{gray}{0.35}` définit la couleur `grisfoncé`.
`\textcolor{grisfoncé}{gris foncé}` donne **gris foncé**

`rbg` pour `Red Green Blue`

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

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

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

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

`\textcolor{couleur1}{couleur 1}` donne **couleur 1**
`\textcolor{couleur2}{couleur 2}` donne **couleur 2**

Enfin `\definecolor{couleur3}{rgb}{1,1,1}` donne le blanc et `\definecolor{couleur4}{rgb}{0,0,0}` donne le noir.

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

Je vous conseille le très bon document de Arnaud Gazagnes qui s'intitule :

LaTeX... pour le prof de maths !

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

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

`cmyk` pour `Cyan Magenta Yellow blacK`

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

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

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

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

Précisions sur `\color` et `\textcolor`

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

- Pour `\color`

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

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

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

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

- Pour `\textcolor`
  ça fonctionne exactement de la même façon que `color` :
  l'instruction `\textcolor[rgb]{0.8 0 0}{texte}` écrit le mot texte en rouge foncé.

`\colorbox`

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

`\colorbox{couleur}{texte}`

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

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

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

texte blanc sur fond rouge

`\fcolorbox`

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

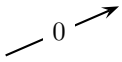
`\fcolorbox{couleur du cadre}{couleur du fond}{texte}`

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

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

On peut également combiner `\fcolorbox` avec `\textcolor` par exemple pour écrire un texte bleu sur fond vert, avec un cadre noir ;
`\fcolorbox{black}{green}{\textcolor{white}{texte bleu sur fond vert}}` donne :

texte bleu sur fond vert

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

pour signifier dans un tableau de variations qu'une fonction continue et strictement croissante passe par 0 ; il suffit de placer au bon endroit une boîte dont le fond et le cadre sont blancs :
`\psline[arrowsize=3pt 2]{->}(-0.5,-0.2)(1,0.4)\fcolorbox{white}{white}{$0$}`

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

1.3 Package `pstricks-add`

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

Pourquoi cette extension dans une chronique consacrée aux couleurs ?

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

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

1.3.1 Couleurs prédéfinies et utilisation

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

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

`{\darkgray gris foncé}` écrira gris foncé

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

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

1.3.2 Définition de nouvelles couleurs

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

`\newgray`

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

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

On l'utilise ainsi :

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

`\newrgbcolor`

C'est l'instruction que j'utilise pour créer de nouvelles couleurs.
Elle nécessite deux paramètres : le nom de la nouvelle couleur, les trois composantes rouge-vert-bleu sous forme de trois nombres décimaux entre 0 et 1 séparés par un espace.
Ainsi `\newrgbcolor{brun}{0.6 0.2 0}` définit le brun que l'on écrit `{\brun brun}`.

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

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

`\newcmykcolor`

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

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

où `c_cyan`, `c_magenta`, `c_jaune` et `c_noir` sont respectivement les coefficients de cyan, de magenta, de jaune et de noir dans la nouvelle couleur.
Le mode d'emploi officiel de `pstricks-add` précise que la définition des couleurs au moyen de cette instruction peut poser des problèmes en PostScript ; laissons tomber !

`\newhsbcolor`

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

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

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

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

1.4 Noir et blanc

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

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

On crée un tableau avec quelques formules :

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

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

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

sans rien effacer, bien sûr !

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

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

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

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

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

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