ColorRgb.TEX : guide d'utilisation des macros de mise en couleur

Manuel de Référence

Version 3.3

Christophe Cérin

Université de Picardie
Faculté de Mathématique et d'Informatique
Lamifa
33, rue St Leu – 80039 Amiens
tel : 22-82-75-03
&
Université de Paris Sud
LRI bat 490
91405 Orsay - France
mail : cerin@lri.lri.fr
tel : 69-41-70-82

Petite histoire de ces macros

Dans un article des Cahiers Gutenberg [1], nous présentions les avantages d'utilisation de la couleur dans le traitement de texte TEX au travers de deux jeux de macros : colorrgb.tex et colorcmyk.tex fonctionnant pour les imprimantes couleur PostScript des constructeurs QMS, Océ, Hitachi, Seiko.

Dans cet article, nous discutions de la couleur en général, comment la spécifier, la représenter, puis, nous dégagions des classes de problèmes qui appellent des classes de solutions communes, des algorithmes communs. Cela nous conduisait à proposer un ensemble minimal de primitives permettant de résoudre les problèmes identifiés suivants : action sur une entité syntaxique (lettre, mot, phrase, paragraphe) et action sur une entité graphique : vecteur ou flèche de l'environnement `picture` de LATEX.

Au moyen des constructeurs de base `\begincolor` et `\endcolor`, nous réalisions des dégradés de couleur sur des chaînes de caractères : la couleur d'une lettre dépend de celle de ses voisines ou, plus généralement, de paramètres spécifiant l'interaction des couleurs.

La version 2.9 des macros bénéficiait de l'ajout de primitives comme la création de boîtes en couleur contenant du texte dans une couleur différente. Vous trouviez également des macros pour écrire du texte sur un fond dégradé et des primitives pour dessiner des courbes de Bezier en couleur et en particulier des contours fermés quelconques, des cercles hachurés ainsi que la possibilité de placer du texte le long d'une courbe.

Les principaux ajouts de la version 3.3 sont les suivants : dessins de flèches composés d'un spline et d'un symbole de flèche, le symbole de la flèche pouvant être positionné n'importe où le long du spline. Nous proposons également une macros pour dessiner la représentation graphique d'une fonction mathématique et diverses solutions pour tracer des échelles graduées et des grilles ainsi que des macros pour tracer des boites 3D orientables. Vous disposez aussi de macros pour dessiner d'une part des logos de flèches (il y a une macro pour positionner ces logos dans la marge gauche) et d'autre part des logos de stations de travail, de disque, de serveur et de routeur.

L'implantation des constructeurs initiaux proposée est réalisée en partie avec PostScript, par l'intermédiaire de la commande `special` dans la syntaxe reconnue par le translateur `dvips 5.4x` de Tom Rokicki, et en TEX. L'ensemble des macros est écrit pour fonctionner avec LATEX.

Modèles RGB, CMYK et HSB

Ce module décrit l'ensemble des primitives qui spécifient la couleur dans les modèles RGB, CMYK et HSB. Il est constitué d'une partie permettant de déclarer des variables, d'une partie permettant d'effectuer des initialisations sur ces variables et d'une partie qui décrit la manipulation de la couleur sur des chaînes de caractères. Nous proposons ensuite les descriptions des macros graphiques puis nous listons quelques macros de confort qui déclarent des couleurs prédéfinies.

Primitives de base pour le changement de couleur

beginrgbcolor{rouge}{vert}{bleu}
Passage, dans le modèle RGB, dans la couleur spécifiée par les paramètres.

endrgbcolor
Retour à la couleur avant l'appel à `\beginrgbcolor`.

rgbcolor{rouge}{vert}{bleu}
Effectue un `\beginrgbcolor` suivi d'un `\endrgbcolor` sur le groupe (cf définition d'un group dans [5]) courant.

beginhsbcolor{hue}{saturation}{brightness}
Passage, dans le modèle HSB, dans la couleur spécifiée par les paramètres.

endhsbcolor
Retour à la couleur avant l'appel à `\beginhsbcolor`.

begincmykcolor{cyan}{yellow}{magenta}{black}
Passage, dans le modèle CMYK, dans la couleur spécifiée par les paramètres.

endcmykcolor
Retour à la couleur avant l'appel à `\begincmykcolor`.

Déclarations concernant les couleurs

rouge=int
contient la valeur de la composante rouge.

vert=int
contient la valeur de la composante verte.

bleu=int
contient la valeur de la composante bleue.

increrouge=long
contient la valeur de l'incrément du rouge.

increvert=long
contient la valeur de l'incrément du vert.

increbleu=long
contient la valeur de l'incrément du bleu.

modulo=long
valeur du nombre de lettres dans la même couleur.

maxrouge=int
contient la valeur de la composante rouge maximale.

maxvert=int
contient la valeur de la composante verte maximale.

maxbleu=int
contient la valeur de la composante bleue maximale.

Primitives d'initialisation des couleurs

initcolor{rouge}{vert}{bleu}
{maxrouge}{maxvert}{maxbleu}{modulo}
Procédure d'initialisation des couleurs, des couleurs maximales et du paramètre « modulo » permettant de spécifier le nombre de caractères devant être imprimés dans la même couleur.

initinc{rouge}{vert}{bleu}
Procédure d'initialisation des valeurs d'incrément de chaque couleur par des valeurs de type `int`.

Déclarations concernant la forme des traits

Pour certaines primitives présentées plus loin, vous avez la possibilité d'affecter la forme des traits en appelant d'abort les primitives suivantes :

CCap,**Join**
Ces variables permettent de modifier la forme des bouts de traits et la forme des jointures entre traits. Les seules valeurs autorisées sont 0, 1 et 2 (cf opérateurs PostScript `setlinecap` et `setlinejoin`). Par défaut, ces valeurs sont fixées comme suit : `\CCap=1\Join=1`.

InitCapJoinMatrix{cap}{join}
Cette macro réalise l'initialisation des variables `CCap, Join`.

SetCapJoinMatrix{cap}{join}
Cette macro permet de modifier, au niveau du code PostScript, les valeurs des formes des traits en exécutant les ordres `setlinecap` et `setlinejoin`.

Couleur sur les chaînes

fboxcolor{texte}{rouge}{vert}{bleu}
Boite dont le contour est tracé en couleur. La macro est inspirée de la macro LaTeX `\fbox` et elle utilise les variables `\fboxrule` et `\fboxsep`. Elle ne peut donc s'utiliser que sous LaTeX.

boxcolor{texte}{rouge}{vert}{bleu}
Boite dont le contenu est en couleur. Cette macro est inspirée de la macro LaTeX `\fbox` et elle utilise les variables LaTeX `\fboxrule` et `\fboxsep`. De plus, on utilise une nouvelle boîte `\@myboxb` et une nouvelle dimension `\@mydimb` afin d'éviter certains conflits de variables si l'on effectue un `\fbox{}` de la boîte en couleur.

underlinecolor{texte}{rouge}{vert}{bleu}
Le texte `texte` est souligné en couleur.

dentdescie{texte}
Cette procédure prend une chaîne de caractères et imprime les « modulo » premiers caractères dans la couleur spécifiée par les trois composantes rouge, vert, bleu définies au moyen de la primitive `\initcolor` ; puis décrémente chaque composante d'une valeur spécifiée au moyen de la primitive `\initinc` et imprime encore « modulo » caractères dans cette nouvelle couleur jusqu'à ce que « modulo » atteigne une valeur maximale. Dans ce cas, on recommence l'affichage des « modulo » premiers caractères suivants à partir de la couleur initiale spécifiée par les trois premiers paramètres de la primitive `\initcolor`.

echelon{texte}
Cette procédure prend une chaîne de caractères et imprime les « modulo » premiers caractères dans la couleur spécifiée par les trois composantes rouge, vert, bleu définies au moyen de la primitive `\initcolor` ; puis incrémente chaque composante d'une valeur définie au moyen de la primitive `\initinc` et imprime encore « modulo » caractères dans cette nouvelle couleur jusqu'à ce que « modulo » atteigne une borne max. Dans ce cas, on recommence l'affichage des « modulo » caractères suivants en décrémentant les couleurs jusqu'aux valeurs initiales qui sont sauvegardées dans les variables `\sauvxxx`. La variable globale `\sens` initialisée à 1 détermine un éclaircissement progressif des couleurs alors que `\sens` = 0 détermine un noircissement progressif des couleurs.

degrade{texte}
Cette macro prend une chaîne de caractères puis répartit uniformément les gammes de couleurs définies par la primitive `\initcolor` sur l'ensemble de la chaîne.

longueur[texte]
Cette macro calcule la longueur d'une chaîne de caractères. Le résultat est rangé dans la variable globale `\lg`. Le paramètre doit être entre crochets.

Comment créer des effets spéciaux ?

Voici des macros qui effectue des dégradés concentriques sur des cercles ou linéaires sur des boîtes. L'utilisateur peut placer du texte (éventuellement en couleur) au milieu des cercles ou des boîtes et passer des boîtes TEX comme paramètres aux procédures.

CercleCouleur{texte}{max}{min}{couleur}{nb-nuances}
Macro qui trace des cercles degradés sous le texte «texte»en commençant par un grand cercle de couleur «max»jusqu'au petit cercle de couleur «min»en tracant «nb-nuances»nuances de couleur «couleur». Si le paramètre de la couleur vaut 1 on effectue un dégradé en rouge, si le paramètre de la couleur vaut 2 on effectue un dégradé en vert, si le paramètre de la couleur vaut 3 on effectue un dégradé en bleu. Remarquer que «nb-nuances»doit être > 0 pour éviter des divisions par 0 !

CercleGeneral{texte}{rouge}{vert}{bleu}
{maxrouge}{maxvert}{maxbleu}{nb-nuances}
Cette macro trace des cercles degradés sous le texte «texte»en commencant par un grand cercle de couleur «rouge, vert, bleu»jusqu'au petit cercle de couleur «maxrouge, maxvert, maxbleu»en tracant «nb-nuances»nuances de couleur. Cette macro est une généralisation de la précédente que l'on conserve pour raison historique. De plus, l'utilisateur peut spécifier un décalage pour le centre du cercle au moyen des variables `\xtrans` et `ytrans`. On peut alors obtenir l'effet d'une boule dégradée (cf exemples). Remarquer que «nb-nuances»doit être > 0 pour éviter des divisions par 0 !

RectangleDegrade{texte}{rouge}{vert}{bleu}
{maxrouge}{maxvert}{maxbleu}{nb-nuances}
Cette macro trace des rectangles dégradés sous le texte «texte»en commencant par un rectangle de couleur «rouge, vert, bleu»jusqu'au rectangle de couleur «maxrouge, maxvert, maxbleu»en tracant «nb-nuances»rectangles de couleur differentes. Pour choisir le sens du dégradé (de gauche à droite ou de haut en bas), vous utiliserez un booléen représenté en TEX par `\ifhorizontal`. Si vous souhaitez un dégradé de couleurs disposées de haut en bas, vous écrirez `\horizontaltrue` ; si vous souhaitez une disposition de couleurs de la gauche vers la droite, vous écrirez `\horizontalfalse` avant l'appel à `\RectangleDegrade`. Remarquer que «nb-nuances»doit être > 0 pour éviter des divisions par 0 !

Tracer des courbes

bezierps liste **endbezierps**
{wd}{dash1}{dash2}{red}{green}{blue}

Cette macro trace une courbe de bezier en couleur avec les points du paramètre `liste` qui est un tableau de tableaux au sens PostScript. Le paramètre `wd` contient la largeur des traits. Lorsque l'on veut tracer des courbes pointillées, il faut positionner le booléen `\ifdashps` à vrai (`\dashpstrue`) et à ce moment `dash1` spécifie la forme des pointillés (voir la commande Postscript `setdash`) et `dash2` définit le déplacement (à partir du début du trait) après lequel débutera le tracé. Si on a positionné `\dashpsfalse` alors la courbe est continue. Attention, les unités sont des millimètres et il faut un blanc avant `\endbezierps`. Enfin, l'ordre des paramètres est :

```
\bezierps [ [ x0 y0 x1 y1 x2 y2 x3 y3 ]...]
\endbezierps {}{}{}{}{}
```

avec les points x_0, y_0, x_1, y_1 définissant la première tangente à la courbe et les points x_2, y_2, x_3, y_3 définissant la deuxième tangente à la courbe.

ellipseps{l}{h}{line}{dash}{deb}{rouge}{vert}{bleu}
Trace une ellipse avec les dimensions (l, h). Le paramètre l représente la *largeur* de l'ellipse et h la hauteur (exprimées en mm). Le paramètre *line* contient la largeur des traits. Lorsque l'on veut tracer une ellipse en pointillées, il faut positionner le boolean `\ifdashps` à vrai (faire la commande `\dashpstrue`) et à ce moment *dash* contient le nombre total d'espaces noirs et d'espaces blancs selon les unités de mesures courantes ; *deb* définit le nombre d'unités à placer en début de trait. Ces deux paramètres sont de la forme [9] 4 ($dash = [9], deb = 4$) car ils doivent correspondre aux paramètres de la commande PostScript `setdash`. Si on a positionné `\dashpsfalse` alors la courbe est continue. Attention, les unités sont des MILLIMETRES. Le paramètre "rouge" spécifie l'intensité du rouge qui remplit l'ellipse (cette valeur doit être inférieure à 256), "vert" l'intensité du vert (cette valeur doit être inférieure à 256) et "bleu" l'intensité du bleu (valeur inférieure à 256). Enfin, le trait du contour de l'ellipse est tracé dans la couleur actuelle définie par les composantes `\rouge, \vert, \bleu`.

Line liste **endLine**{larg}{dash}{deb}{rouge}{vert}{bleu}
Trace une ligne avec les points du paramètre "liste" qui est un tableau de tableaux. Le paramètre "larg" contient la largeur des traits. Lorsque l'on veut tracer des lignes pointillées, il faut positionner le booléen `\ifdashps` a vrai au moyen d'un ordre `\dashpstrue` et à ce moment "dash" contient le nombre total d'espaces noirs et d'espaces blancs selon les unités de mesures courantes ; "deb" définit le nombre d'unités à placer en debut de trait (cf commande précédente). Si on a positionner `\dashpsfalse` alors la courbe est continue. Attention les unités sont des MILLIMETRES et il faut un blanc avant `\endLine`. Le paramètre "rouge" spécifie l'intensité du rouge du trait (cette valeur doit être inférieure à 256), "vert" l'intensité du vert (cette valeur doit

être inférieure à 256) et "bleu" l'intensité du bleu (valeur inférieure à 256). Enfin les valeur courantes de `CCap` et `Join` sont utilisées pour le rendu des traits (cf commande `\SetCapJoinMatrix`)

Pour tracer une ligne brisée il faut taper :

```
\Line [ [ x0 y0 x1 y1 ] [ x1 y1 x2 y2 ] ...] \endLine{}{}...
```

En fait, cet appel provoque, en sous main, la répétition de la fonction qui trace un trait.

CourbeRempliee liste **endcourbe**
{wd}{dash1}{dash2}{red}{green}{blue}
Trace une courbe fermée au moyen de courbes de Bezier avec les points du paramètre `liste` qui est un tableau de tableaux. Le paramètre wd contient la largeur des traits du contour. Lorsque l'on veut tracer des courbes pointillées, il faut positionner le booléen `\ifdashps` à vrai (`\dashpstrue`) et à ce moment `dash1` spécifie la forme des pointillés (voir la commande Postscript `setdash`) et `dash2` définit le déplacement (à partir du début du trait) après lequel débutera le tracé. Si on a positionné `\dashpsfalse` alors la courbe est continue. (cf fonction `\bezierps`). Attention, les unités sont des millimètres et il faut un blanc avant `\endcourbe`. L'ordre des paramètres est le même que celui de la fonction `\bezierps`. Le remplissage est effectué avec les couleurs spécifiées par les trois derniers paramètres. Le contour est tracé dans la couleur actuelle définie par les composantes `\rouge, \vert, \bleu`. Le contour est fermé si le booléen `\ifclosepath` est positionné à vrai (`\closepathtrue`). Enfin, tous les points sont en positions relatives par rapport à x_0, y_0 contrairement aux points définissant les tangentes dans la macro `\bezierps` et il faut respecter les blancs convenablement :

```
\CourbeRempliee [ [ x0 y0 x1 y1 x2 y2 x3 y3 ]...]
\endcourbe{}{}{}{}{}
```

Les objets graphiques "hachurés"

CercleHachure{rayon}{wdrayon}{wd} {*angle*}{*r*}{*g*}{*b*}{*space*}
Cette macro trace un cercle de rayon `rayon` mm et d'épaisseur `wdrayon` mm, puis remplit ce cercle avec des hachures de largeur égale à `wd` mm inclinées de `angle` degrés. La couleur est specifiée par les paramètres r, g, b. Le paramètre `space` donne l'espacement entre les hachures. Les angles doivent être compris entre 0 et 180 degrés.

CarreHachure{ht}{wd1}{wd2} {*angle*}{*r*}{*g*}{*b*}{*space*}
Cette macro trace un carré de hauteur `2*ht` mm et d'épaisseur `wd1` mm, puis remplit ce cercle avec des hachures de largeur égale à `wd2` mm inclinées de `angle` degrés. La couleur est specifiée par les paramètres r,

g, b. Le paramètre `space` donne l'espacement entre les hachures. Les angles doivent être compris entre 0 et 180 degrés.

Faire une rotation

RotObj{angle}{texte}
Pour mettre du texte en rotation ou tout autre chose, programmez avec la macro `\RotObj`. Le point à partir duquel la rotation se fait peut varier suivant l'appel à certaines macros : par défaut ce point est fixé au premier caractère du texte. Par exemple, l'appel `\mc` possitionne ce point au milieu de la chaîne, l'appel `\fc` à la fin, et l'appel `\dc` au début. Il y a seulement création d'une boite de largeur nulle, donc il n'y a pas de place de réservée pour l'objet. Voici un exemple de programmation : `\mc\RotObj{45}{\Huge\bf Exemple}`.

Mettre automatiquement du texte le long d'une courbe

TraceTexte{texte}{x_0}{y_0}{x_1}{y_1}{x_2}{y_2}{x_3}{y_3}
Cette macro répartit pas à pas chaque caractère de la chaîne `texte` le long de la courbe spécifiée par les 4 points $x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3$ (cf macro `bezierps`). La formule mathématique de la courbe de Bezier est dérivée de la paire d'équation cubique paramétrique de paramètre $t\,(0 \leq t \leq 1)$ suivante :

$$\begin{array}{rcll} x(t) & = & a_xt^3+b_xt^2+c_xt+x_0 & \textit{coordonnées en } x \\ y(t) & = & a_yt^3+b_yt^2+c_yt+y_0 & \textit{coordonnées en } y \end{array}$$

Les points de contrôle sont définis par :

$$\begin{array}{ll} x_1 = x_0+c_x/3 & y_1 = y_0+c_y/3 \\ x_2 = x_1+(c_x+b_x)/3 & y_2 = y_1+(c_y+b_y)/3 \\ x_3 = x_0+c_x+b_x+a_x & y_3 = y_0+c_y+b_y+a_y \end{array}$$

Pour chaque lettre à positionner, les différentes valeurs du paramètre t sont calculées en fonction de la largeur des caractères et de la tantente à la courbe. Par exemple, supposons que l'on cherche à positionner le caractère `c` de la figure .4 page 24. On connait la valeur t' et les coordonnées (x', y') du point sur la courbe, obtenu au moyen des équations paramétriques, qui nous ont servi à positionner le caracère `a`. Pour calculer la tangente, nous cherchons la valeur de t'' correspondant à un déplacement le long de la courbe égal à la longueur du caractère `c`. Soit (x'', y'') les coordonées du point sur la courbe associé à t''. On pose :

$$tan(\alpha) = \frac{y' - y''}{x' - x''} \qquad (.1)$$

Le caractère est positionné à partir de (x', y') en le faisant pivoté de l'angle α. On continue ce même traitement avec le prochain caractère de la chaîne à partir du point (x'', y''). Les conditions initiales sont : le premier caractère à positionner est le premier caractère de la chaîne, $t = 0$, l'angle est déterminé au moyen du procédé décrit plus haut et ne correspond pas à la tangente définie par les points x_0, y_0, x_1, y_1 qui est la tangente au point x_0, y_0 (un caractère a une largeur non nulle).

La variable `\BlancInterLettre` vous permet de rajouter de l'espace entre les caractères. Par défaut, sa valeur est fixée à 1mm (on rappelle que l'unité de tous les paramètres numériques doit être le mm).

L'implantation est réalisée en TEX pour ce qui est de l'application récursive des calculs et de l'obtention de la largeur des caractères. Les calculs numériques sont implantés, quant à eux, en PostScript. L'utilisation successive, dans le corps de la macro, de plusieurs ordres `\special` pour insérer du code PostScript nécessite des traitements particuliers car l'implantation du `\special` introduit des ordres de positionnement, en plus du code PostScript que l'on souhaite inséré, qui perturbent le bon fonctionnement. La macro a donc été testée et ne fonctionne pour l'instant qu'avec le translateur `dvips 5.4`. De petits ajustements de ces macros sont nécessaires pour des versions antérieures du translateur `dvips`.

ColleTexte{texte}{x_0}{y_0}{x_1}{y_1}{x_2}{y_2}{x_3}{y_3}
Cette macro répartit la chaîne `texte` le long de la **totalité** de la courbe de Bezier spécifiée par les points $x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3$. Le calcul des différentes valeurs du paramètre t est plus facile que dans le cas de la macro `TraceTexte` : la nouvelle valeur du paramètre est obtenue en augmentant la dernière valeur du paramètre de la quantité $\frac{1}{\mid texte - 1 \mid}$ (attention : une croissance linéaire de t n'entraine pas nécessairement un espacement identique entre chaque caractère du fait de l'équation cubique paramétrique utilisée). Un exemple est présenté à la figure .4 page 24.

À partir de maintenant, les courbes tracées avec les macros qui suivent sont dans la couleur courante spécifiée par `\rouge`, `\vert`, `\bleu`. Il convient donc de positionner convenablement chacune de ces valeurs avant les appels.

Tracer des grilles et des échelles graduées

On vous offre la possibilité de tracer une grille sans graduations ou bien de tracer une grille et des échelles graduées. Pour tracer des grilles, les variables suivantes sont utilisées :

1. `\pasx` contient le "pas" d'espacement en (mm) pour les graduations de l'axe des x ; `\pasy` contient le "pas" d'espacement en (mm) pour les graduations de l'axe des y ;
2. `\vinitx` et `\vincx` contiennent respectivement les valeurs initiales et d'incrémentation des puissances sur l'axe des x ;
3. `\vinity` et `\vincy` contiennent respectivement les valeurs initiales et d'incrémentation des puissances sur l'axe des y ;

La macro de base pour tracer une grille est la suivante :

Grille (larg,haut)(esp-x,esp-y){dash1}{dash2}{wd}
Trace une grille au point courant sur larg mm en largeur et de haut mm en hauteur. Les lignes de la grille sont espacées de esp-x mm selon l'axe des x et de esp-y mm selon l'axe des y. Les paramètres dash1 et dash2 sont les paramètres du setdash dans le cas ou l'on souhaite tracer des lignes pointillées : dans ce cas, le booléen `dashps` doit être positionné à true. wd est la largeur des traits.

GridNumNum (x1,y1)(l,h)(esp-x,esp-y){dash1}{dash2}{wd}
Trace une grille avec des échelles graduées au point $(x1, y1)$. Les graduations sur l'axes des x et des y sont des entiers. Les autres paramètres de cette macro ont les significations de ceux de Grille. Les valeurs de `\pasx` et `\pasy` sont les valeurs en mm d'espacement entre 2 graduations consécutives.

GridNumPuiss (x1,y1)(l,h)(esp-x,esp-y){dash1}{dash2}{wd}
Trace une grille avec des échelles graduées au point $(x1, y1)$. Les graduations sur l'axes des x sont des entiers de la forme de `\vinitx`, les graduations sur l'axes des y des entiers de la forme `\vinity^\vincy`. Les autres paramètres de cette macro ont les mêmes significations que ceux de Grille. Les valeurs de `\pasx` et `\pasy` sont les valeurs en mm d'espacement entre 2 graduations consécutives.

multiputnum (x1,y1)(x-t,y-t){num}{obj}{inc}
Place num objets a partir de $(x1, y1)$ en se translatant de x-t et y-t à chaque nouvel objet. L'objet de départ est *obj*, puis *obj*+*inc* (pas d'itération), puis *obj*+(2∗*inc*)... Attention, obj doit être de type entier.

multiputpuissance (x1,y1)(x-trans,y-trans){num}{x}{y}
Place num objets de la forme x^y à partir de $(x1, y1)$ en se translatant de x-trans mm selon l'axe des x et y-trans mm selon l'axe des y à chaque nouvel objet. L'objet de départ est x^y, puis x^{y+1}, puis x^{y+2}... Attention, x doit être un entier ainsi que le paramètre num.

Tracer des fonctions mathématiques

plotfunction{deb}{fin}{x-fact}{y-fact}{width}{dash}{fct}
Trace une fonction décrite au format PostScript par le paramètre fct. Le paramètre en x à faire figurer doit s'appeler XX. La valeur de XX varie de deb à fin. On applique le facteur d'echelle x-fact selon l'axe des x et le facteur y-fact selon l'axe des y. Attention, l'unité par défaut est ici le centimètre. width est l'épaisseur des traits en cm. dash spécifie le paramètre des pointillés si le booléen dashps est vrai (`dashpstrue`). La fonction est tracée dans la couleur spécifiée par `\rouge, \vert, \bleu`.

Tracer des flèches

Vous avez la possibilité de tracer des flèches avec le symbole de flèche déposé n'importe où sur le trait ou encore une flèche comportant deux symboles de flèche, un à chaque extrémité du trait. Les traits des flèches sont définis comme des splines.

Les booléens suivant permettent de choisir un endroit pour déposer les flèches :

- `\ifarrowheads` positionné à la valeur true vous permet d'avoir deux symboles de flèches ;
- `\ifarrowheadleft` positionné à la valeur true vous permet d'obtenir un symbole de flèche au début du trait de flèche ;
- `\ifarrowheadright` positionné à la valeur true vous permet d'obtenir un symbole de flèche en fin du trait de flèche ;

ArrowForm{sens}
Cette macro permet d'initialiser la forme des flèches. Un appel avec le paramètre sens valant h va sélectionner le mode"deux symboles de flèches", un appel avec le paramètre sens valant l va sélectionner le mode "symbole de flèches au début" et un appel avec le paramètre sens valant r va sélectionner le mode "symboles de flèches à droite". Au moment de la compilation, un message vous averti si vous avez effectué un appel à cette macro avec un paramètre autre que ceux explicités ici.

Pour tracer les flèches proprement dites vous avez à votre disposition les macros suivantes :

Fleche spline **endfleche**{*pourc*}{*prof*}{*haut*}{*larg*}{*pt-size*}
{wd}{dash1}{dash2}
Les paramètres de cette macro sont les suivants : spline est un tableau de tableaux à deux cases. La première case contient du texte, la seconde

4 points définissant le spline support de la flèche, pourc est le pourcentage de la longueur du spline où va figurer le symbole de flèche, prof est la profondeur de la flèche, haut sa hauteur, larg sa largeur, pt-size est la hauteur de la fonte pour le texte wd (cf exemples d'utilisation) est l'epaisseur des traits dash1 et dash2 permettent de tracer des pointillés (cf commande PostScript `setdash`).

FlecheSpeciale spline **endflechespeciale**{*pos*}{*prof*}
{haut}{larg}{pt-size}{wd}{dash1}{dash2}
Cette macro est un raccourci pour tracer une flèche avec le(s) symbole(s) de flèche à l'une des extrémités du trait. Le paramètre pos est soit un h soit un r soit un l (cf au dessus pour les définitions de ces symboles). Les autres paramètres sont les mêmes que ceux de `Fleche`.

TraceArrow (x1,y1)(x2,y2)|shape|
Trace deux symboles de flèches aux extrémités d'une droite fictive définie par les points (x_1, y_1) et (x_2, y_2). Le paramètre shape est de la forme $p\ h\ w$ ou p, h, w sont respectivement la profondeur la hauteur et la largeur de la flèche. Enfin, si on a le booléen `\externe` positionné à true alors les flèches sont tournées vers l'extérieur, sinon vers l'intérieur. Cette macro est d'une utilité pour le tracé des cotations.

Pour tracer des cotations vous disposez des macros suivantes :

Cotes (x1,y1)(x2,y2)[ep,ep-per][lg-g,lg-d]|dash|
Trace une droite du point $A = (x_1, y_1)$ au point $B = (x_2, y_2)$ d'épaisseur ep mm puis deux traits perpendiculaires à la droite précédente d'épaisseur ep-per et de longueurs respectives lg-g mm du point A et lg-d mm du point B. Enfin, si le booléen `\dashps` est positionné à true alors dash spécifie les pointillés. Ainsi, dash doit être de la forme : `[ xx ] yy` (cd commande PostScript `setdash`).

CotesFleche (x1,y1)(x2,y2)[ep,ep-per][lg-g,lg-d]|shape|
Cette macro est très semblable à la précédente. On ajoute des symboles de flèches au bout des traits. Elle commence par tracer une droite du point $A = (x_1, y_1)$ au point $B = (x_2, y_2)$ d'épaisseur ep mm puis deux traits perpendiculaires à la droite précédente d'épaisseur ep-per et de longueurs respectives lg-g mm du point A et lg-d mm du point B. Le paramètre shape est de la forme w h w pour un appel a Arrow. Enfin, si on a `externetrue` alors les flèches sont tournées vers l'extérieur, sinon vers l'intérieur.

Pour tracer des logos de flèches vous disposez des fonctions suivantes :

LogoFlecheOne{epaisseur}{scale-x}{scale-y}
Trace un logo de flèche dont les traits sont d'épaisseur *epaisseur*. Les facteurs *scale-x* et *scale-y* permettent de grossir ou de diminuer la taille

du dessin selon l'axe des x et l'axe des y respectivement. Le symbole est dessiné dans la couleur des paramètres `\rouge`, `\vert`, `bleu`.

LeftLogoFlecheOne
Positionne dans la marge gauche le logo précédent. Le symbole est dessiné dans la couleur des paramètres `\rouge`, `\vert`, `bleu`.

LogoFlecheTwo{epaisseur}{scale-x}{scale-y}
Pour la définition des paramètres, voir la macro `LogoFlecheOne`.

LeftLogoFlecheTwo
Positionne dans la marge gauche le logo précédent. Le symbole est dessiné dans la couleur des paramètres `\rouge`, `\vert`, `bleu`.

Tracer des boites 3D pour réaliser des histogrammes

TroisDbox{h}{w}{ang}{c-box}{c-inner}
{x-retrait}{y-retrait}{rayon}
Trace une boite 3D de hauteur h.mm et w.mm orienté de ang. La couleur de la boite est donnée par le paramètre c-box qui doit être un triplet de la forme 'rouge vert bleu'. Le parametre c-inner est la couleur des faces. Ces faces ont une hauteur égale a h $-$ 2$*$x-retrait (mm) et une largeur de w $-$ 2$*$y-retrait (mm). Le paramètre rayon est le rayon des arrondis aux quatre coins des faces. Noter que si ce paramètre vaut 0 alors on obtient le dessin d'un rectangle. De plus, lorsque le booléen innerbox est True (`\innerboxtrue`) la face avant de la boite ne comporte pas de boite intérieure en couleur.

Pour obtenir des logos informatiques

SunStation{epaisseur}{scale-x}{scale-y}
Trace un logo de station de travail dont les traits sont d'épaisseur *epaisseur*. Les facteurs *scale-x* et *scale-y* permettent de grossir ou de diminuer la taille du dessin selon l'axe des x et l'axe des y respectivement. Le symbole est dessiné dans la couleur des paramètres `\rouge`, `\vert`, `bleu`.

SunDisk{epaisseur}{scale-x}{scale-y}
Trace un logo d'unité de disque dont les traits sont d'épaisseur *epaisseur*. Les facteurs *scale-x* et *scale-y* permettent de grossir ou de diminuer la taille du dessin selon l'axe des x et l'axe des y respectivement. Le symbole est dessiné dans la couleur des paramètres `\rouge`, `\vert`, `bleu`.

SunServer{epaisseur}{scale-x}{scale-y}
Trace un logo de serveur dont les traits sont d'épaisseur *epaisseur*. Les facteurs *scale-x* et *scale-y* permettent de grossir ou de diminuer la taille du dessin selon l'axe des x et l'axe des y respectivement. Le symbole est dessiné dans la couleur des paramètres `\rouge`, `\vert`, `bleu`.

SunRouter{epaisseur}{scale-x}{scale-y}
Trace un logo de routeur de communication dont les traits sont d'épaisseur *epaisseur*. Les facteurs *scale-x* et *scale-y* permettent de grossir ou de diminuer la taille du dessin selon l'axe des x et l'axe des y respectivement. Le symbole est dessiné dans la couleur des paramètres `\rouge`, `\vert`, `bleu`.

Quelques macros de confort

Afin de faciliter la manipulation de textes, de graphiques en couleur nous proposons quelques raccourcis.

icone
Equivalent à un `\item` et place sur la feuille un point en bleu.

ictwo
Equivalent à un `\item` et place sur la feuille un diamand en bleu.

bcrouge
Remplace la couleur courante par la couleur rouge.

bcbleu
Remplace la couleur courante par la couleur bleue.

bcjaune
Remplace la couleur courante par la couleur jaune.

bcvert
Remplace la couleur courante par la couleur vert.

bcmauve
Remplace la couleur courante par la couleur mauve.

bcmagenta
Remplace la couleur courante par la couleur magenta.

bcorange
Remplace la couleur courante par la couleur orange.

bcvj
Remplace la couleur courante par la couleur vert-jaune.

bcblanc
Remplace la couleur courante par la couleur blanche.

ec
Substitue la couleur sauvegardée à la couleur courante.

Pour changer de couleur, vous disposez également de macros qui portent les noms des primitives qui permettent de spécifier une couleur sous l'environement X11 : `\MediumAquamarine`, `\coral`, `\DarkOliveGreen`, `\MediumOrchid`... Reportez-vous au fichier `colorrgb.tex` pour obtenir toutes les redéfinitions.

Exemples d'utilisation dans le modèle RGB

Pour mettre en couleur le mot « bonjour » dans cette phrase, il suffit de baliser le mot par `\beginrgbcolor{255}{0}{0}` (ou `\bcrouge`) et `\endrgbcolor` (ou `\ec`) pour obtenir bonjour . De la même façon, on peut baliser des traits, des flèches de l'environnement `\picture`.

Par exemple, le trait qui suit est codé de la manière suivante :

```
\bcbleu %un trait bleu de longueur 8cm
\hrule height 2pt width 8cm %et d'epaisseur 2pt
\ec
```

Voici un exemple d'utilisation de la macros `\echelon` avec initialisation des couleurs et des incréments de chaque couleur. Les couleurs sont initialisées avec la valeur 20 et on désire qu'elles croissent jusqu'à la valeur 210 par saut de 40 entre chaque caractères (modulo = 1).

```
{% debut de groupe
\Huge\bf
% rouge=bleu=vert=20
% maxrouge=maxvert=maxbleu=210
% chaque caractere change de couleur
\initcolor{20}{20}{20}{210}{210}{210}{1}
% increrouge=increvert=increbleu=40
\initinc{40}{40}{40}
\echelon{la\ fonction\ echelon}
}% fin du groupe
```

la fonction echelon

Pour construire des boites dont le fond est en couleur, et le texte sur deux colonnes dans une autre couleur, nous programmons :

```
\setbox2=\vbox{\hsize=0.485\textwidth
Dans un article r\'ecent \cite{cc}, nous pr\'esentions
les avantages d'utilisation de la couleur dans le
traitement de texte \TeX\  au travers de deux jeux
de macros~: Color\TeX rgb et color\TeX cmyk fonctionnant
pour les imprimantes couleur PostScript des constructeurs
QMS, Oc\'e, Hitachi, Sei\-ko.

identifi\'es suivants~: action sur une entit\'e
syntaxique (lettre, mot, phrase, paragraphe) ; et action
sur une entit\'e graphique~: vecteur ou fl\`eche de
l'environnement {\bf picture} de \LaTeX .
\par
}% fin de la bo\^\i te numero 2
% on annule le blanc interbo\^\i te
\splittopskip=0pt
% on coupe la bo\^\i te 2 en deux selon la hauteur
% et on range dans la bo\^\i te 1
\setbox1=\vsplit2 to 0.5\ht2%
\setbox1=\vbox{\unvbox1}%
% on pose sur la feuille les deux bo\^\i tes
\hbox{\boxcolor{\box1}{200}{255}{255}\quad\vrule\quad
      \boxcolor{\box2}{50}{255}{255}}
```

On remarque que le texte est collé sur le bord de la boîte en couleur ; pour ajouter du blanc, il faut utiliser la variable LaTeX `\fboxsep`.

Dans un article récent [1], nous présentions les avantages d'utilisation de la couleur dans le traitement de texte TeX au travers de deux jeux de macros : ColorTeXrgb et colorTeXcmyk fonctionnant pour les imprimantes couleur PostScript des constructeurs QMS, Océ, Hitachi, Seiko.

Dans cet article, nous discutions de la couleur en général, comment la spécifier, la représenter, puis, nous dégagions des classes de problèmes qui appellent des classes de solutions communes, des algorithmes communs. Cela nous conduisait à proposer un ensemble minimal de primitives permettant de résoudre les problèmes identifiés suivants : action sur une entité syntaxique (lettre, mot, phrase, paragraphe) ; et action sur une entité graphique : vecteur ou flèche de l'environnement **picture** de LaTeX.

Nous présentons maintenant le codage d'un tableau avec les pourtours en magenta. On notera l'utilisation de la macro `\halign` avec la séquence `\bcmagenta\tvc#\endrgbcolor&` ce qui provoque un passage à la couleur magenta pour tous les traits verticaux séparant les colonnes. Pour une explication détaillée de la séquence `#&` dans l'environnement d'un `\halign`, le lecteur est invité à se reporter à [7], pages 126 et suivantes.

```
$$\vbox{\offinterlineskip
\def\cc#1{\kern .2em\hfill\raise 1.5mm%
                 \hbox{#1}\hfill\kern .2em}
\def\tvi{\vrule height 24pt depth 2pt width 0pt}
\def\tvc{\tvi\vrule width 1mm}
\halign{\bcmagenta\tvc#\endrgbcolor&%
&\cc{$\displaystyle#$}&\bcmagenta\tvc#\endrgbcolor\cr
\noalign{\bcmagenta\hrule height 1mm\endrgbcolor}
&\omit\cc{\bf Fonction}&&\omit\cc{\bf Primitive}&\cr
\noalign{\bcmagenta\hrule height 1mm\endrgbcolor}
     height 34pt depth 30pt&{ 1 \over 1 + x^2 }&&{\rm
         Arctg}\,(x)+C&\cr
\noalign{\bcmagenta\hrule height 1mm\endrgbcolor}
}}$$
```

Fonction	**Primitive**
$\frac{1}{1}+x^2$	$\mathrm{Arctg}\,(x)+C$

Pour coder une boîte en couleur avec un cadre dans une autre couleur :

```
{\Large
\fboxsep=0pt% on initialise le blanc
\fboxrule=3pt% la dimension du cadre est de 3pt
\fboxcolor{\fboxsep=8pt\fboxrule=0pt
    \boxcolor{du texte en noir}%
            {255}{255}{0}}{255}{0}{0}
}
```

du texte en noir

Pour tracer un cercle avec des dégradés concentriques qui partent de la couleur rouge jusqu'au noir en décalant le centre (effet boule dégradée), on programme :


```
\setbox100=\vbox{\Huge\bf
\halign{\quad#&\quad#&\quad#\cr
\beginrgbcolor{255}{255}{255}
& LRI& \cr
LRI& LRI& LRI\cr
& LRI& \cr}
\endrgbcolor}
\xtrans=50pt% on decale le centre de 50pt vers la droite
\ytrans=50pt% et 50pt vers le haut
\CercleGeneral{\copy100}{255}{0}{0}{0}{0}{0}{10}
```

Pour tracer une boîte dont le contenu est dégradé, on programme :

Dans un article récent, nous présentions les avantages d'utilisation de la couleur dans le traitement de texte TEX au travers de deux jeux de macros : ColorTEXrgb et colorTEXcmyk fonctionnant pour les imprimantes couleur PostScript des constructeurs QMS, Océ, Hitachi, Seiko.

```
\setbox10=\vbox{\hsize=0.6\textwidth\bf
\beginrgbcolor{255}{255}{255}% texte blanc
Dans un article r\'ecent, nous pr\'esentions
les avantages d'utilisation de la couleur dans le
traitement de texte \TeX\  au travers de deux jeux
de macros~: Color\TeX rgb et color\TeX cmyk fonctionnant
pour les impri\-man\-tes couleur PostScript des constructeurs
QMS, Oc\'e, Hitachi, Seiko.

\par
}
\horizontaltrue % des degrades de haut en bas
\centerline{%
\RectangleDegrade{\copy10}{255}{255}{0}{0}{0}{0}{50}}
```

Pour tracer les courbes de bezier des figures .1 et .2, on pourra marquer avec `Pictex` ou `TexCad` les points définissant les tangentes, puis récupérer la valeur de ces points sous éditeur et programmer :

```
\begin{figure}[th]
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(74.00,61.00)
\put(0.00,0.00){\framebox(74.00,61.00)[t]{}}
\put(3.97,10.00){\vector(1,0){28.10}}
....
% debut de la premiere courbe de bezier (celle de gauche)
\put(0,0){\bezierps [ [ 2.06 36.98 6.03 39.00 10.95 39.00
 14.03 29.02][14.03 29.02 19.05 16.06 23.02 16.98 30.00 13.97]]
 \endbezierps{.4}{3}{2}{255}{0}{0}}
```

```
% fin de la premiere courbe de bezier (celle de droite)
% debut de la 2 courbe de bezier (celle de gauche)
\put(0,0){\bezierps [[33.97 39.00 37.94 39.00 40.00 37.94
42.06 34.92] [42.06 34.92 44.92 26.98 50.95 16.98 66.98 12.06]]
\endbezierps{.4}{3}{2}{255}{0}{0}}
% fin de la 2 courbe de bezier
% je place un cercle hachure
\put(6.03,38){\CercleHachure{4.5}{.5}{.1}{60}{255}{60}{255}{2}}
\put(42,35){\CarreHachure{4}{.6}{.1}{135}{255}{0}{255}{1}}
....
\put(6.03,54){\makebox(0,0)[cc]{$f(x)$}}
\put(42,54){\makebox(0,0)[cc]{$f(x)$}}
\end{picture}
\caption{{\it extremum particulier}}
\label{extre}
\end{center}
\end{figure}
```

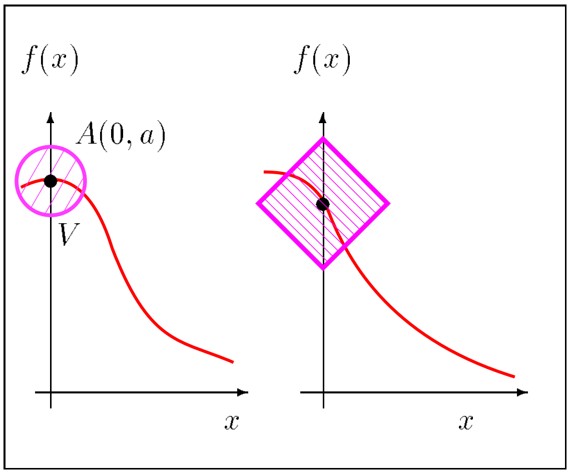


Figure .1 : *extremum particulier*

```
\begin{figure}[th]
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(74.00,61.00)
\put(0.00,0.00){\framebox(74.00,61.00)[t]{}}
....
% debut de la courbe de bezier
\put(0,0){\bezierps [
[ 16.03 14.92 20.00 16.03 26.98 39.05 33.97 43.02 ]
```

```
[ 33.97 43.02 42.06 40.95 43.97 20.95 47.94 19.05 ]
[ 47.94 19.05 52.06 20.95 54.92 49.05 56.98 52.06 ]
[ 56.98 52.60 60.00 49.05 62.06 32.06 64.92 17.94 ]
] \endbezierps{.4}{2}{.5}{255}{0}{0}}
....
\end{picture}
\caption{{\it extremums locaux et globaux}}
\label{extretwo}
\end{center}
\end{figure}
```

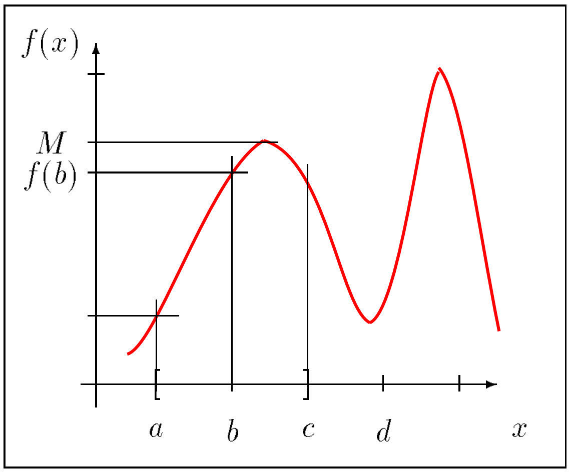


Figure .2 : *extremums locaux et globaux*

Pour tracer des objets quelconques au moyen de courbes de Bezier dont le fond est en couleur, nous programmons :

```
\begin{figure}
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(124,100)
\put(0,0){\framebox(124,100)[cc]{}}
\put(20,80){\makebox(0,0)[cc]{\Large\bf\LaTeX}}
\put(97,5){\makebox(0,0)[cc]{\small
          $\copyright - lri - 1991$}}
% Definition de la couleur du contour.
% Le trait est en pointille.
\rouge=255\vert=0\bleu=255\dashpstrue
```

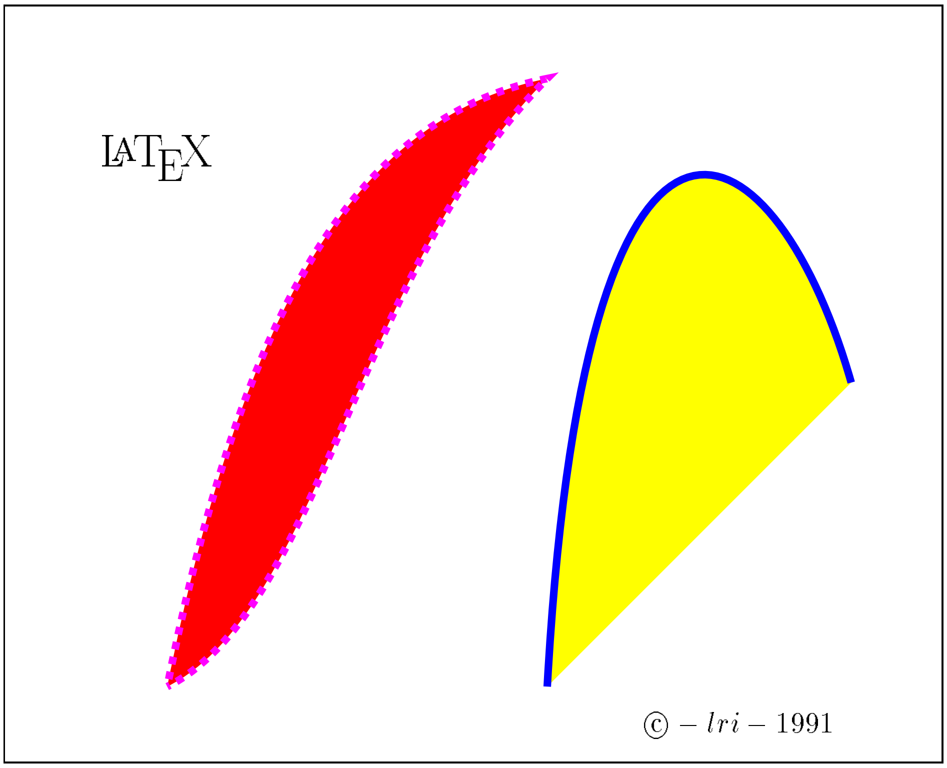


Figure .3 : *Des courbes remplies*

```
\put(0.00,0.00){\CourbeFermee [
[ 20 10 10 50 25 75 50 80 ]
[ 50 80 25 55  20 10 0 0 ]
] \endcourbe{1}{1}{1}{255}{0}{0}}
% Le contour n'est plus en pointille
\dashpsfalse\closepathfalse
\rouge=0\vert=0\bleu=255
\put(0.00,0.00){\CourbeFermee
   [ [ 70 10 5 90 30 75 40 40 ]
] \endcourbe{1}{1}{1}{255}{255}{0}}
\end{picture}
\caption{{\it Des courbes remplies}}
\label{f:remplie}
\end{center}
\end{figure}
```

qui produit le résultat de la figure .3

Pour placer du texte le long de courbes de Bezier il faut programmer, par exemple, qui produit le résultat de la figure .4 :

Il y a un bug lorsque l'on demande une impression à une résolution autre que 300dpi : les lettres ne sont pas placées au bon endroit.

```
\Huge\bf
\BlancInterLettre=1% 1 mm entre les lettres
\begin{figure}[htp]
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(95,70)
\put(0,0){\LightBlue\TraceTexte{TraceTexte}
      {5}{65}{15}{35}{35}{35}{90}{65}\ec}
\put(0,0){\bcvert\ColleTexte{ColleTexte}
      {5}{5}{15}{35}{15}{35}{81}{5}\ec}
\put(0.00,0.00){\framebox(95,70)[cc]{}}
\end{picture}
\caption{{\it courbes sur un texte}}
\label{fd:one}
\end{center}
\end{figure}
```

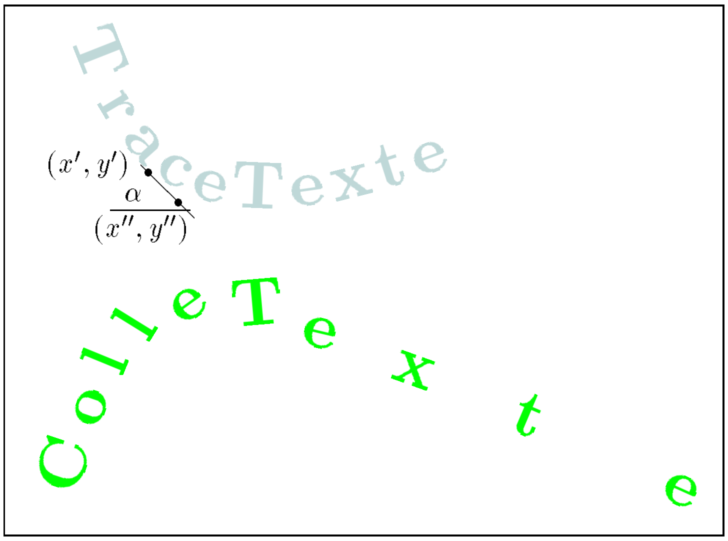
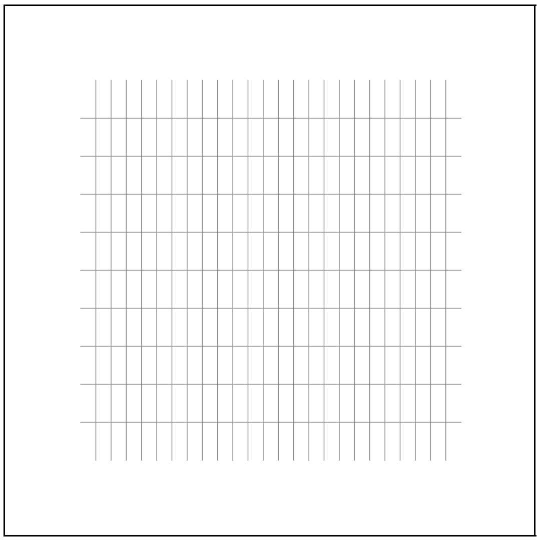


Figure .4 : *courbes sur un texte*

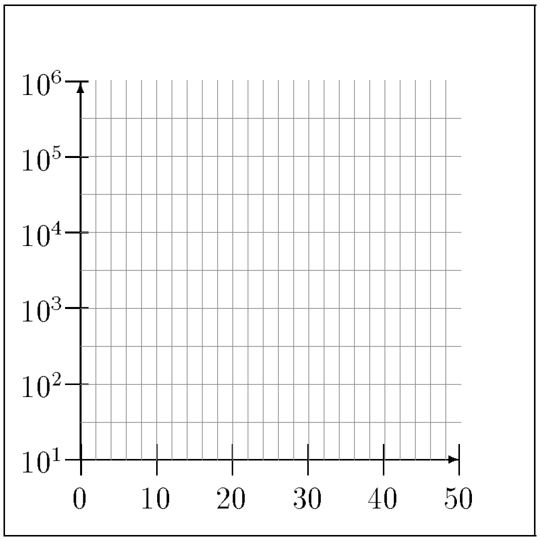
Manipulation des grilles, des échelles graduées

Pour tracer une grille simple (sans axes et en "trait plein") faites :



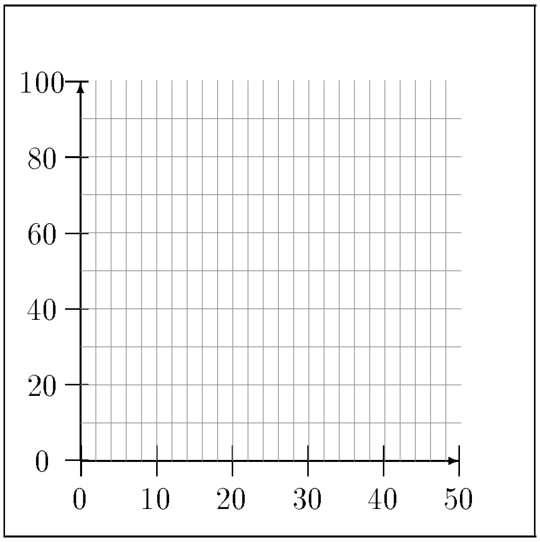
```
\begin{center}
\unitlength=1mm
\linethickness{0.4pt}
\begin{picture}(70,70)
\put(0.00,0.00){\framebox(70,70)[t]{}}
\put(10.00,10.00){\Grille(50,50)(2,5){2}{4}{0.1}}
\end{picture}
\end{center}
```

Pour tracer une grille complété d'axes gradués avec, sur l'axe des x des valeurs augmentant de 10 en 10 et sur l'axe des y des valeurs d'exposant augmentant de 1 en 1, vous pouvez faire :




```
\begin{center}
\unitlength=1mm
\linethickness{0.4pt}
\vinitx=0
\vincx=10
\vinity=10
\vincy=1
\begin{picture}(70,70)
\put(0.00,0.00){\framebox(70,70)[t]{}}
\GridNumPuiss(10,10)(50,50)(2,5){2}{4}{0.1}
\end{picture}
\end{center}
```

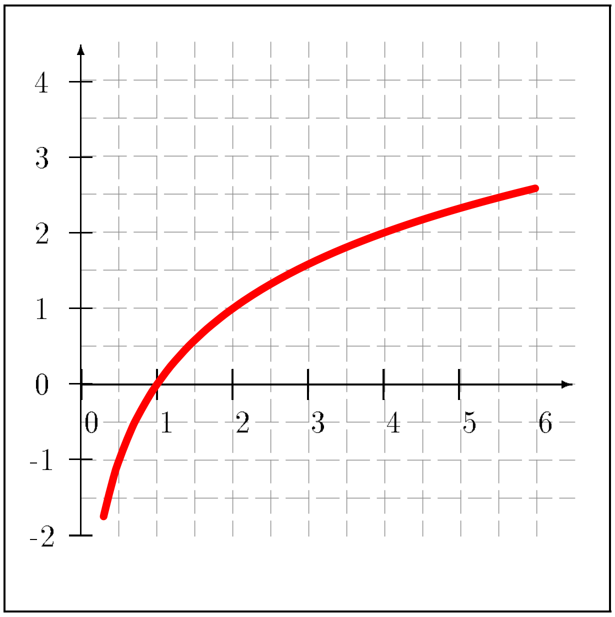
La variante pour obtenir le dessin qui suit se code :



```
\begin{center}
\unitlength=1mm
\linethickness{0.4pt}
\vinitx=0
\vincx=10
\vinity=0
\vincy=20
\begin{picture}(70,70)
\put(0.00,0.00){\framebox(70,70)[t]{}}
\GridNumNum(10,10)(50,50)(2,5){2}{4}{0.1}
\end{picture}
\end{center}
```

Pour tracer des fonctions

Pour tracer en rouge la fonction $\log_2$ pour x variant entre 0.3 et 5.1 accompagnée des axes et de la grille de fond pointillée, vous pouvez faire :




```
\begin{center}
\unitlength=1mm
\linethickness{0.4pt}
\begin{picture}(80,80)
\put(0.00,0.00){\framebox(80,80)[t]{}}
%%% grille en pointillee
\dashpstrue
```

```
\put(10.00,10.00){\Grille(65,65)(2,5){3 1 3 1}{1}{0.1}}
\dashpsfalse% fin des traits pointilles
%%% les axes
\put(10,30){\vector(1,0){65}}
\put(10,10){\vector(0,1){65}}
\multiputnum(11.5,25)(10,0){7}{0}{1}
\multiputnum(5,10)(0,10){7}{-2}{1}
\multiput(8.5,10)(0,10){7}{\traith{3}}
\multiput(10,28)(10,0){6}{\traitv{4}}
%%% la fonction
\put(10,30){\plotfunction{0.3}{5.1}{1}{1}{0.1}{0}%
          {XX  ln 2 ln div}}
\end{picture}
\end{center}
```

Noter que l'on a pas accés en PostScript à la fonction $\log_2$ si bien que nous avons dû la composer : $\log_2(x) = \frac{ln(x)}{ln(2)}$. Reportez vous aux références [4] [3] et [2] pour obtenir la liste des fonctions (arithmétiques...) disponibles en PostScript. De plus, il est rappelé que le nom de la variable des fonctions est `XX` par convention.

Manipulation des flèches

L'exemple de la Figure .5 illustre les différentes formes de flèches que l'on peut obtenir avec `colorrgb.tex` :

```
\begin{figure}[htp]
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(135,165)
\put(0.00,0.00){\framebox(135,165)[t]{}}
%%% debut des axes
\put(10,10){\vector(1,0){110}}
\put(10,10){\vector(0,1){150}}
\multiputnum(10,5)(10,0){10}{0}{1000}
\multiputpuissance(5,10)(0,10){15}{10}{1}
\multiput(8.5,20)(0,10){14}{\traith{3}}
\multiput(20,8)(10,0){9}{\traitv{4}}
%%%% fin des echelles graduees
\dashpstrue\rouge=0\vert=0\bleu=0
\put(0,0){\dashpsfalse\rouge=0\vert=0\bleu=0%
\FlecheSpeciale [
```

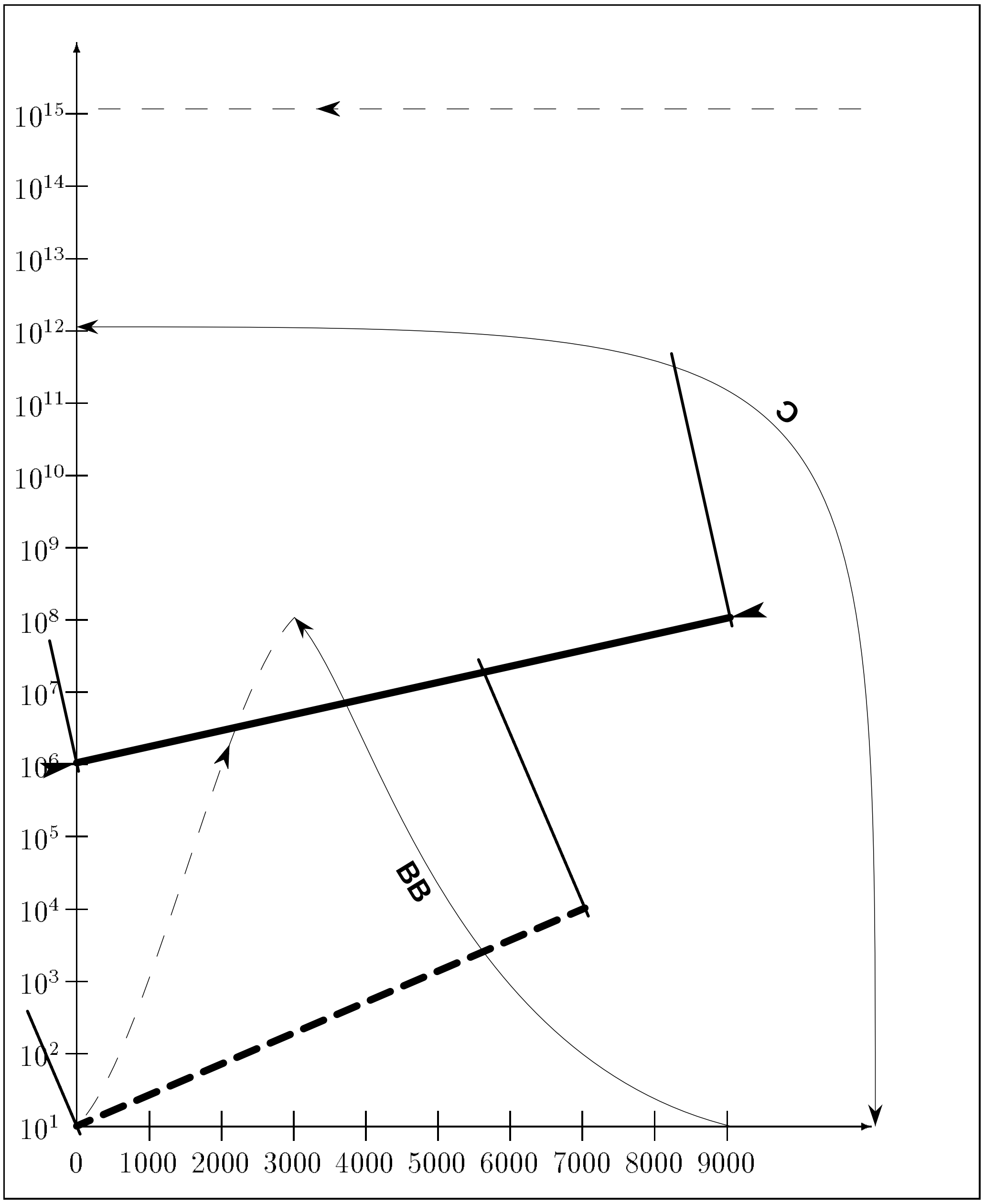


Figure .5 : *Dessins de flèches*

```
  [[(C)] [120 10 120 120 120 120 10 120] ]
] \endflechespeciale{h}{1}{3}{2}{4}{0.1}{3}{4}}%
\put(0,0){\Fleche{[[[()] [10 10 20 20 30 70 40 80]]
  [[()] [120 150 100 150 40 150 10 150] ]
]} \endfleche{70}{1}{3}{2}{4}{0.1}{3}{4}}%
\put(0,0){\dashpsfalse\rouge=0\vert=0\bleu=0%
\FlecheSpeciale [
  [[(BB)] [40 80 50 70 60 20 100 10]]
] \endflechespeciale{l}{1}{3}{2}{4}{0.1}{3}{4}}%
\put(0,0){\dashpstrue\rouge=0\vert=0\bleu=0%
\Cotes(10,10)(80,40)[1,0.4][16,36]|[ 3 2 3 2 ] 1|}%
\put(0,0){\dashpsfalse\rouge=0\vert=0\bleu=0\externefalse%
\CotesFleche(10,60)(100,80)[1,0.4][16,36]|1 5 2.2|}
\end{picture}
\caption{{\it Dessins de fl\`eches}}
\label{fd:f}
\end{center}
\end{figure}
```

Tracer des boites 3D

Pour tracer des boites 3D vous pouvez vous inspirer du code suivant pour obtenir le résultat de la Figure .6 :

```
\begin{figure}[htp]
\begin{center}
\unitlength=1.00mm
\linethickness{0.4pt}
\begin{picture}(135,135)
\put(0.00,0.00){\framebox(135,135)[t]{}}
%%%% debut des boites
\put(35,10){\TroisDbox{60}{30}{-10}{ 1 0 0 }%
           { 1 1 0.5 }{2}{1}{5.25}}
\innerboxfalse
\put(95,20){\TroisDbox{50}{20}{10}{ 1 0 0 }%
           { 1 1 0.5 }{1}{1}{1.25}}
\innerboxtrue
\put(95,100){\TroisDbox{20}{20}{0}{ 0 0.5 1 }%
           { 1 1 0.5 }{1}{1}{1.25}}
%%%% fin des boites
\put(35,105){\makebox(0,0)[lc]{\large\bf Boites 3D}}
\end{picture}
\caption{{\it Boites 3D}}
```

```
\label{fd:box}
\end{center}
\end{figure}
```

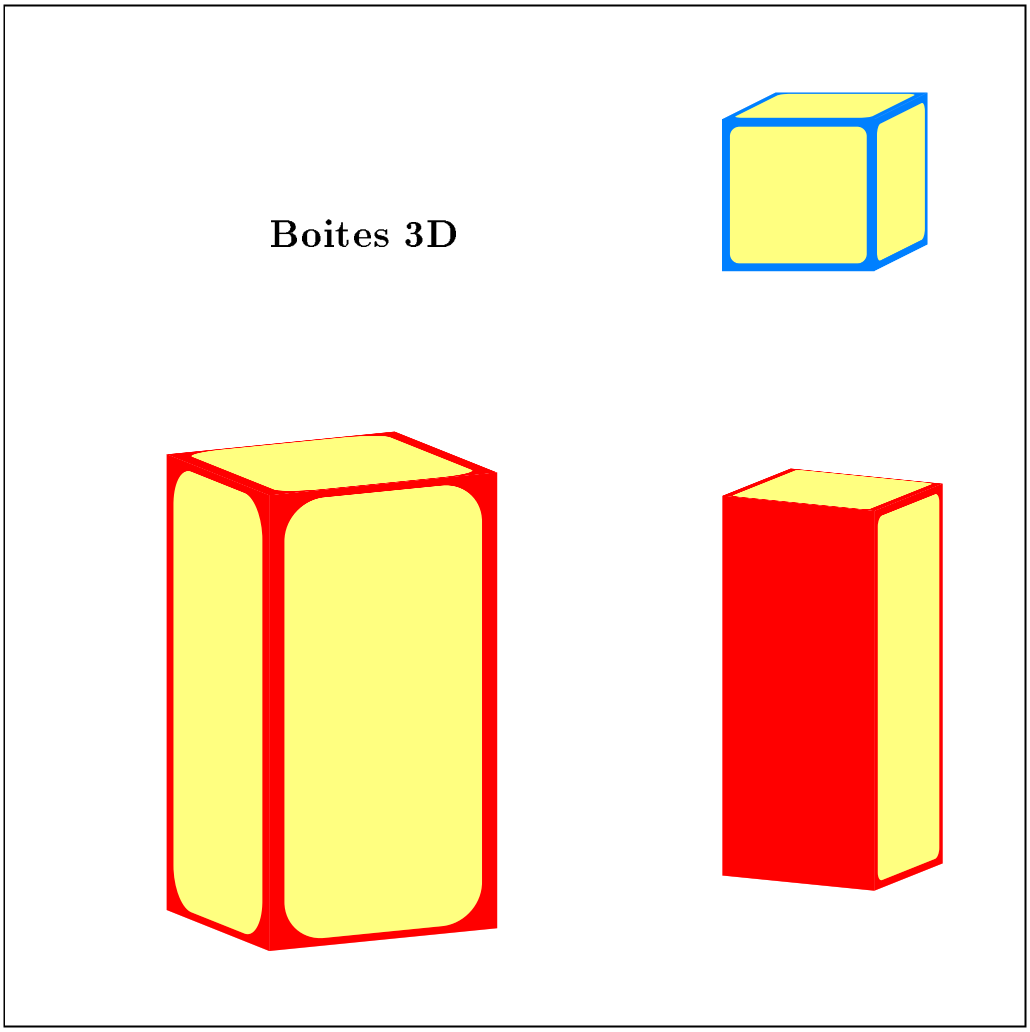


Figure .6 : *Boites 3D*

Quelques logos de flèches

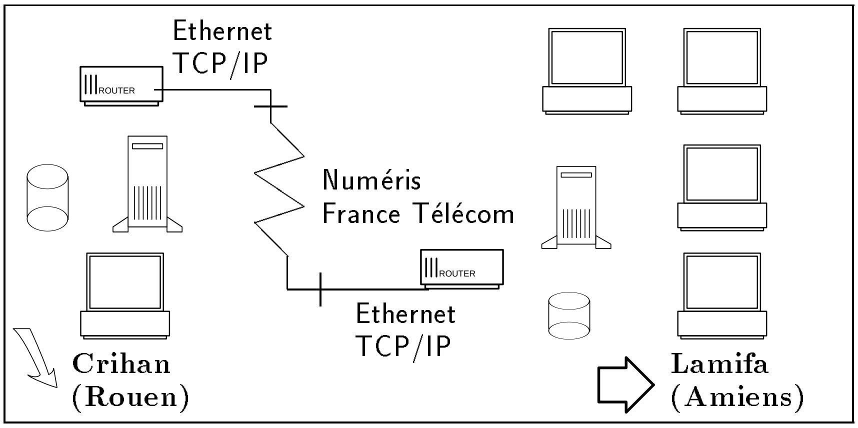
Les macros qui suivent permettent de tracer des logos représentant des symboles de flèches ; il s'agit de `\LogoFlecheOne` et `LogoFlecheTwo`. De plus, un support est prévu pour écrire ces symboles dans la marge gauche au travers des macros `\LeftLogoFlecheOne` et `LeftLogoFlecheTwo` que l'on place au sein d'un paragraphe.

Par exemple, le symbole que vous voyez apparaître dans la marge gauche est codé :

```
Par exemple, le symbole\LeftLogoFlecheTwo\ que vous voyez
appara\^{\i}tre dans la marge gauche est cod\'e :
```

Logos informatiques : dessins de stations de travail, de disques, de routeurs...

Voici maintenant un exemple où est résumé l'utilisation des symboles informatiques :



La programmation est la suivante :

```
\rouge=0\vert=0\bleu=0% c'est du noir !
\setlength{\unitlength}{0.11mm}
\begin{picture}(1020,498)(-580,240)
\put(-580,240){\framebox(1020,500){}}
\put(220,260){\small\bf (Amiens)}
\put(220,300){\small\bf Lamifa}
\put(-500,260){\small\bf (Rouen)}
\put(-500,300){\small\bf Crihan}
\put(-380,660){\small\sf TCP/IP}
\put(-380,700){\small\sf Ethernet}
\put(-160,320){\small\sf TCP/IP}
\put(-160,360){\small\sf Ethernet}
\put(-200,480){\small\sf France T\'el\'ecom}
\put(-200,520){\small\sf Num\'eris}
\put(-280,620){\line(1,0){40}}
\put(-200,420){\line(0,-1){40}}
```

```
\put(-240,400){\line(1,0){170}}
\put(-240,440){\line(0,-1){40}}
\put(-280,480){\line(1,-1){40}}
\put(-220,500){\line(-3,-1){60}}
\put(-280,540){\line(3,-2){60}}
\put(-220,560){\line(-3,-1){60}}
\put(-260,600){\line(1,-1){40}}
\put(-260,640){\line(0,-1){40}}
\put(-400,640){\line(1,0){140}}
%---- Site du crihan ---------
\put(-500,340){\SunStation{0.21}{0.85}{0.85}}
\put(-460,470){\SunServer{0.21}{0.6}{0.7}}
\put(-565,470){\SunDisk{0.21}{0.6}{0.85}}
\put(-500,620){\SunRouter{0.21}{1.2}{1.2}}
\put(-517,282){\closepathfalse%
\LogoFlecheTwo{0.02}{0.55}{0.55}}
%---- Site du Lamifa ---------
\put(218,610){\SunStation{0.21}{0.85}{0.85}}
\put(55,610){\SunStation{0.21}{0.85}{0.85}}
\put(218,470){\SunStation{0.21}{0.85}{0.85}}
\put(218,340){\SunStation{0.21}{0.85}{0.85}}
\put(55,450){\SunServer{0.21}{0.6}{0.6}}
\put(62,340){\SunDisk{0.21}{0.6}{0.6}}
\put(-91,400){\SunRouter{0.21}{1.2}{1.2}}
\put(200,285){\closepathfalse
\LogoFlecheOne{0.04}{0.9}{0.9}}
\end{picture}
```

Bibliographie TEX

[**1**] Christophe Cérin et Benoît Lemaire *Vers l'introduction de la couleur dans TEX*, Cahiers GUTenberg numéro 5, pages .

[**2**] Adobe Systems Incorporated. *PostScript Language Reference Manual*, Addison-Wesley, October 1986.

[**3**] Adobe Systems Incorporated. *PostScript Language Tutorial and Cookbook*, Addison-Wesley, March 1987.

[**4**] Adobe Systems Incorporated. *PostScript Language Program Design*, Addison-Wesley, February 1988.

[**5**] Donald E.Knuth, *The TEXbook*, Addison-Wesley, 1986.

[**6**] Leslie Lamport. LATEX *user's guide & reference Manual*, Addison-Wesley, 1986.

[**7**] Raymond Seroul, *Le petit livre de TEX*, InterEdition, mai 1989.