

Sommaire

- 5.1 Apéritifs
- 5.2 Du format des fichiers graphiques
- 5.3 Le package `graphicx`
- 5.4 Quelques extensions utiles
- 5.5 Utiliser `make`
- 5.6 À part ça

Graphisme

*Tu ne te feras aucune image sculptée de rien
qui ressemble à ce qui est dans les cieux là-haut [...]
Tu ne te prosternerás pas devant ces images ni ne les serviras.*

Le Deutéronome Dt 5 8.

AUJOURD'HUI il est tout à fait naturel d'insérer des dessins, figures et autres images dans un document. Ceci est dû aux imprimantes de plus en plus performantes et bon marché. Il faut cependant se replacer dans le contexte des années 80 à l'essor de `TEX`. C'est l'époque de l'apparition des imprimantes et le matériel de qualité professionnelle n'était pas accessible au particulier. Cependant beaucoup de solutions d'impression émergeront s'appuyant la plupart sur le langage PostScript devenu *ipso facto* un standard.

Il existe plusieurs solutions autour de \LaTeX pour insérer des graphiques dans un document. Parmi elles on notera l'utilisation de `metafont` (l'utilitaire qui gère les fontes de \LaTeX), la programmation d'un environnement `picture` ou la mise en œuvre d'un code `PICTEX`. Ces solutions ne seront pas décrites ici car nous considérons qu'elles sont d'une utilisation un peu déroutante au premier abord ; il est tout de même bon de connaître leur existence. L'approche adoptée dans ce manuel pour manipuler des graphiques est d'insérer dans le source \LaTeX un fichier au format PostScript encapsulé contenant le graphique en question, ce dernier ayant été créé par un logiciel de dessin tel que `xfig`, `gnuplot`, `gimp`, etc.

5.1 Apéritifs

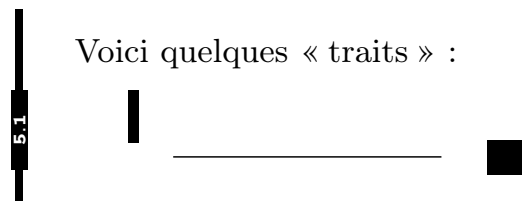
Il n'est pas inutile de connaître la commande `\rule` qui permet de faire des traits :

```
\rule[hpos]{largeur}{hauteur}
```

où `hpos` impose une éventuelle translation verticale du trait, les deux autres arguments ont un nom suffisamment explicite :

Voici quelques `\og traits \fg{}` :

```
\begin{center}
  \rule[1ex]{1mm}{5mm}\quad\rule{1in}{0.4pt}
  \quad\rule[-0.5em]{1em}{1em}
\end{center}
```



5.2 Du format des fichiers graphiques

Pour inclure des dessins ou des images dans vos documents, il faut insérer un *fichier*. La configuration de \LaTeX permet d'incorporer des fichiers de type PS pour PostScript et EPS pour Encapsuled PostScript. Ce fichier peut être généré par n'importe quel programme. Si le format PostScript vous semble contraignant, sachez que :

- tout *bon* logiciel de dessin « vectoriel » vous permet d'exporter vos schémas au format EPS. Ce format est devenu la référence en matière d'impression.

- toute image peut être convertie au format EPS. Sur un système UNIX, le programme `convert` permet d'effectuer cette opération.¹ On pourra aussi recourir au logiciel `gimp` (logiciel libre de retouche et de création d'image numérique), présent également sur d'autres systèmes d'exploitation.

5.3 Le package graphicx

L^AT_EX, ou plutôt T_EX, n'a pas été initialement conçu pour manipuler des graphiques (images, dessins,...). De ce fait, une multitude d'extensions ont été proposées, aucune n'ayant vraiment réussi à s'imposer ou à être vraiment indépendante des systèmes d'exploitation.

5.3.1 Un standard

Aujourd'hui, les concepteurs de L^AT_EX semblent s'être mis d'accord pour standardiser une extension *graphique*. Deux extensions ont donc vu le jour à fin de l'année 1994 :

- `graphics` l'extension « standard » ;
- `graphicx` l'extension « plus plus ».

Nous avons choisi de vous présenter `graphicx`. Il faut bien comprendre que si l'interface de cette extension est indépendante du système d'exploitation, la partie du code gérant les différents types de fichiers graphiques est dépendante du système sous-jacent. Aussi, est-il nécessaire de préciser un *driver* de package. Les drivers existants correspondent aux implantations connues de T_EX sur des plate-formes diverses.² Sous UNIX, le driver utilisé est généralement `dvips`, il est choisi par défaut dans la distribution `teTEX`, si bien que la ligne :

```
\usepackage{graphicx}
```

suffit pour mettre en route l'extension `graphicx`. La commande pour inclure un dessin ou une figure est la suivante :

```
\includegraphics[option]{fichier}
```

où `fichier` est un fichier contenant votre figure, et `option` est une liste d'options séparées par des virgules. La commande `\includegraphics` ne crée pas de


1. Cherchez du côté de la suite « ImageMagick » pour obtenir ce programme s'il n'est pas présent sur votre système.

2. On notera entre autres : `xdvi` et `dvips` pour le monde UNIX, `texture` et `OzTEX` pour le Mac, `emTEX` et `dviwin` pour windows.

mise en page particulière, elle insère juste une boîte contenant le graphique dans le texte. Ainsi :


avant
`\includegraphics{punch}`
 et après.



 Pour assurer la portabilité de vos sources et ainsi pouvoir insérer des fichiers graphiques dans des formats différents, il est impératif de *ne pas préciser l'extension du fichier dans la commande `\includegraphics`*.

En général, on combine `\includegraphics` avec un environnement `figure`. Par exemple, la figure 5.1 a été produite grâce au code suivant :

```
\begin{figure}
  \centering\includegraphics[width=5cm]{punch}
  \caption{Robert (après quelques bières).}
  \label{fig-exemple}
\end{figure}
```

 Notez bien que l'environnement `figure` assure que le graphisme « flotte » dans la page et que ça n'est pas la commande `\includegraphics` qui assure ce rôle. Au cas où ça aurait échappé à certains :

L'environnement `figure` assure que le graphisme « flotte » dans la page ; ça n'est pas la commande `\includegraphics` qui assure ce rôle.

5.3.2 Options

Le package `graphicx` possède plusieurs options permettant de contrôler l'insertion des graphiques. Parmi les options disponibles voici les plus utilisées :

Changement d'échelle

Il existe trois manières d'agir sur la taille d'un graphique.

- `scale=ratio`, où `ratio` est un nombre positif ou négatif, permet de changer la taille globale de la figure ;
- `width=dimen` permet d'imposer la largeur du graphique ;
- `height=dimen` permet d'imposer la hauteur du graphique.



FIG. 5.1 – Robert (après quelques bières).

5

```

\begin{center}
  \includegraphics[scale=0.2]{magma}
  \includegraphics[width=8.5mm]{magma}
  \includegraphics[width=2cm,height=3mm]{magma}
\end{center}

```

5.3



Rotation

Vous pouvez si vous le désirez, faire effectuer une rotation à votre figure en utilisant l'option `angle`, dont la syntaxe est la suivante :

`angle=ndegre`

où `ndegre` est un angle précisé en degrés dans le sens trigonométrique.

```

\includegraphics[angle=45,scale=0.2]{magma}

```

5.4

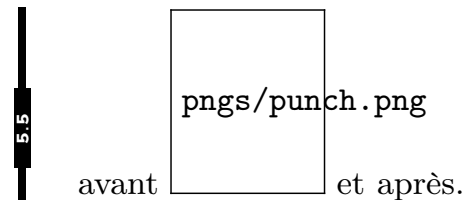


On trouvera dans le fichier `grfguide.pdf`³ une description détaillée de cette extension. On pourra également consulter le fichier `fepslatex.pdf`⁴.

Mode brouillon

L'option `draft` permet de produire les figures en mode « brouillon » : seul un cadre avec le nom du fichier inclus est produit dans le document final.

avant
`\includegraphics[draft,scale=.2]{punch}`
 et après.



Le mode `draft` est enclenché par défaut lorsque l'option de document `draft` est spécifiée. Si vous voulez contrer l'effet de l'option de document⁵, il est possible d'utiliser l'option `final` de la commande `\includegraphics` ou au moment d'inclure l'extension avec `\usepackage`.

2

5.4 Quelques extensions utiles

Voici dans les paragraphes qui suivent trois extensions utiles pour la production de documents contenant des graphiques.

5.4.1 subfig

Cette extension permet de gérer des figures comportant plusieurs sous-figures, avec numérotation automatique et possibilité de faire référence aux sous-figures elles-mêmes. Par exemple :

```
\begin{figure}[htbp]
  \begin{center}
    \leavevmode
    \subfloat[Magma]{%
      \label{fig-uniweria-magma}
      \includegraphics[width=2cm]{magma}}
    \hspace{2cm}
  \end{center}
\end{figure}
```

3. Utiliser `locate` ou `find` pour le trouver sur votre système.

4. <http://tug.ctan.org/tex-archive/info/epslatex/french/fepslatex.pdf>

5. Par exemple, si vous voulez voir vos figures mais aussi les « OverfullBoxMark. »



FIG. 5.2 – Uniwersia Zëkt

```

\subfloat[UZMK]{%
  \label{fig-uniwersia-uzmk}
  \includegraphics[height=2cm]{uzmk}}
\caption{Uniwersia Zëkt}
\label{fig-uniwersia}
\end{center}
\end{figure}

```

Pour ce qui concerne les références, on peut soit référencer la figure globale par `\ref{fig-uniwersia}` qui donne : 5.2, soit les sous-figures par leur label respectif : `\ref{fig-uniwersia-magma}` et `\ref{fig-uniwersia-uzmk}` qui donnent : 5.2a et 5.2b.



Une manière élégante de gérer les `\subfigures` est d'encapsuler chacune d'elles dans un environnement `minipage`. Le fichier `subfig.pdf` accompagnant la distribution, montre comment personnaliser l'environnement `subfigure`, notamment les espaces inter-légendes.

5.4.2 Le package `wrapfig`

Le package `wrapfig` propose l'environnement `wrapfigure` permettant de faire flotter une figure dans un paragraphe. Il ne s'agit pas d'un environnement flottant au sens de l'environnement `figure` de \LaTeX puisqu'on spécifie la position de la figure dans le paragraphe. La syntaxe est la suivante :

```

\begin{wrapfigure}{position}{largeur}
  ...
\end{wrapfigure}

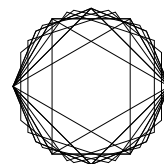
```

où `position` est la position de la figure (l ou r) et `largeur` la largeur de la figure à insérer. Voici un exemple :

```
\begin{wrapfigure}{r}{1.5cm}
  \includegraphics[width=1cm]{polygons}
\end{wrapfigure}
```

Le package `\ltxcom{wrapfig}` n'est ---~à ma connaissance~--- pas documenté sous la forme d'un fichier `\texttt{dvi}` ; par contre il est possible...

Le package `\wrapfig` n'est — à ma connaissance — pas documenté sous la forme d'un fichier `dvi` ; par contre il est possible de trouver des informations très détaillées dans le fichier `.sty` lui-même qui se trouve dans l'arborescence `TEX` dans : `[...]/misc/wrapfig.sty`. On notera au passage — car il faut parler pour faire un paragraphe un peu long — que la règle veut que tout package soit « auto-documenté » grâce à une extension connue sous le nom de `docstrip`. Ainsi toute extension — *package* en anglais — contient aussi bien le code que la documentation. Une procédure d'installation permet d'extraire l'un et l'autre. L'auteur de `wrapfig` n'a vraisemblablement pas suivi cette règle, tant pis...



2

5.4.3 Le package `psfrag`

Une autre extension intéressante est l'extension `psfrag`. Elle a pour but de pouvoir réunir la puissance d'un fichier PostScript et la beauté des équations de `LATEX`. Un problème se pose en effet lorsque l'on veut intégrer des formules à un dessin, car la génération d'équations n'est pas prévue dans la plupart de ces logiciels. La solution adoptée par les auteurs de `psfrag` est d'utiliser la commande `\psfrag` pour insérer les formules à la place de chaînes de caractères présentes dans le dessin. Ainsi, pour avoir la figure 5.3b au lieu de la figure 5.3a, on a procédé comme suit :

1. ajout avant le `\includegraphics{courbe}` la ligne :

```
\psfrag{exp(-x)*sin(10*x)}[r][r]{ $e^{-x}\sin(10x)$ }
```

qui permet de remplacer la chaîne de caractères faisant office de légende par une belle équation ;

2. le résultat n'est pas visible dans le fichier `.dvi`, par contre `dvips` se charge d'exploiter les instructions précédentes pour modifier le fichier PostScript généré.

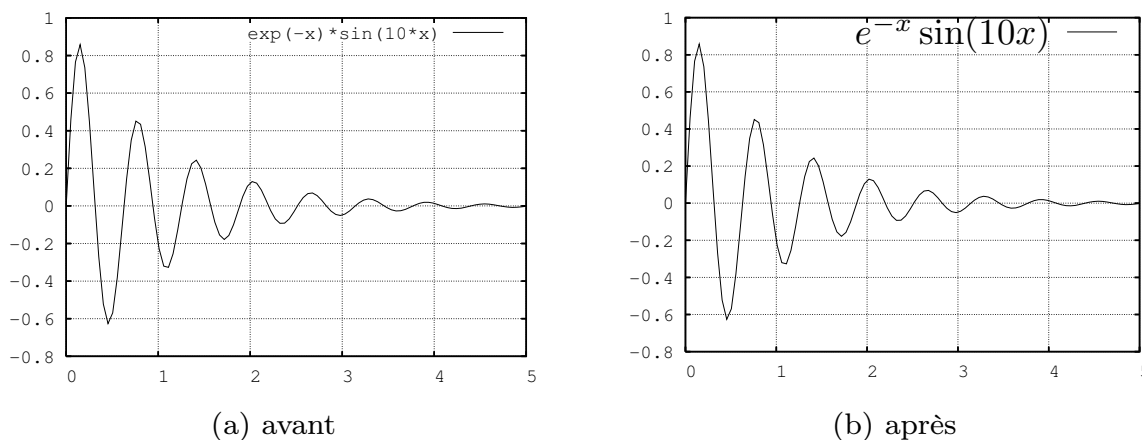
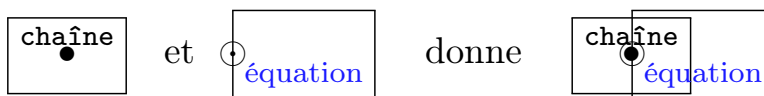


FIG. 5.3 – Utilisation de `psfrag`, à gauche la figure originale, à droite la figure avec un remplacement par une équation \LaTeX .

Le positionnement de la formule se fait en faisant correspondre deux points de référence, l'un appartenant à l'équation, l'autre à la chaîne de caractères à remplacer. C'est à l'utilisateur d'indiquer où se trouve ces points de référence, par l'intermédiaire de deux arguments optionnels à la commande `\psfrag`. Supposons qu'on définisse ces points de référence comme suit :

```
\psfrag{chaîne}[l][c]{équation}
```

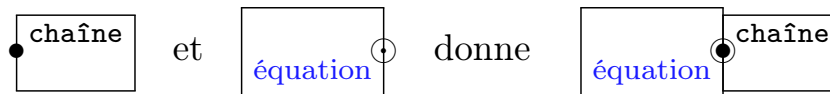
On aura alors l'assemblage suivant :



De même en écrivant :

```
\psfrag{chaîne}[r][l]{équation}
```

on aura :



Dans l'exemple de la figure 5.3b, on a fait correspondre le côté droit de l'équation (1^{er} argument optionnel `r`) avec le côté droit de la chaîne (2^e argument optionnel `r`). La documentation du package est très instructive à ce sujet...



Attention, si on génère un document pdf à partir du source \LaTeX , on ne peut utiliser le package `psfrag` qu'au prix de manipulations un peu tordues.

5.4.4 Le package xcolor

L'extension xcolor est mise au point par l'équipe qui a développé le package graphicx. Il peut être intéressant — par exemple pour produire des transparents — de générer du texte en couleur. Le package xcolor permet les constructions suivantes :

Du texte `{en \color{red}rouge}` et
`\textcolor{cyan}{en cyan}`.

Une boîte `\colorbox{green}{Verte}`.

Une `\fcolorbox{blue}{yellow}{autre boîte}`.

5.6

Du texte en rouge et en cyan.

Une boîte Verte.

Une autre boîte.

On aura compris qu'on dispose pour le texte :

- de la déclaration :

```
\color{couleur}
```

- et de la commande :

```
\textcolor{couleur}{texte}
```

et pour les boîtes :

- sans bordure :

```
\colorbox{couleur du fond}{contenu}
```

- avec bordure :

```
\fcolorbox{couleur bordure}{couleur fond}{contenu}
```

Les deux commandes pour les boîtes en couleur sont sensibles à la longueur `\fboxsep`. « *Quid* des couleurs qui n'ont pas de nom ? » vous entends-je marmonner *in petto*... Ce à quoi je réponds sur le champ :

Voici un
`{\color[rgb]{.2,.4,.5}\bfseries bleu gris}`...

5.7

Voici un bleu gris...

Il est aussi possible de donner un « petit nom » à cette dernière couleur :

```
\definecolor{bleugris}{rgb}{.2,.4,.5}
Voici un
{\color{bleugris}\bfseries bleu gris}...
```

5.8

Voici un **bleu gris**...

Vous noterez qu'en lieu et place du modèle de couleur « rgb » il est possible d'utiliser le modèle gray de manière à définir des nuances de gris. De même, en utilisant le modèle html, on pourra utiliser la syntaxe du langage Html pour spécifier les couleurs.

5.5 Utiliser make



Ce paragraphe est destiné aux utilisateurs d'un système d'exploitation disposant de l'utilitaire Gnu make (pour de plus amples informations sur cet utilitaire, n'hésitez pas lire à l'incontournable [7]). Les autres peuvent passer leur chemin...

Voici donc une idée de makefile qui vous permettra d'automatiser la génération :

- des fichiers au format Eps à partir des images « bitmaps » ;
- des fichiers au format Eps à partir de fichiers stockés dans le format d'un logiciel de dessin vectoriel.

On souhaite pour ce faire, élaborer une cible que l'on précisera en ligne de commande :

```
make figs
```

On suppose que les images et les fichiers de dessins sont respectivement stockés dans les sous-répertoire **Imgs** et **Figs** du répertoire contenant le document maître et que les fichiers Eps fabriqués seront également stockés dans un sous-répertoire **Epss** (voir la figure 5.4 page suivante). On commencera donc par définir un ensemble de variables précisant les différents répertoires à utiliser :

```
FIGSDIR=Figs
EPSSDIR=Epss
IMGSDIR=Imgs
```

5.5.1 Convertir les images

Tout d'abord on fait la liste des fichiers au format Jpeg et Png (c'est un exemple) en stockant cette liste dans deux variables comme suit :

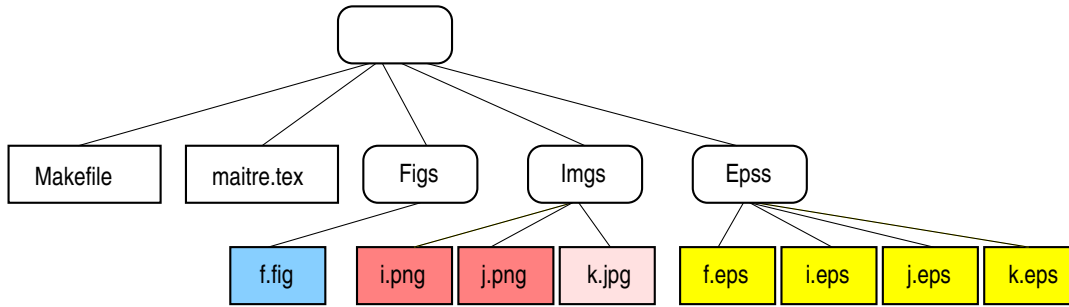


FIG. 5.4 – Proposition d'arborescence pour stocker les fichiers graphiques : un répertoire pour les images et un répertoire pour les graphiques vectoriels. Les fichiers au format Eps sont enregistrés dans un répertoire à part.

```
PNGS=$(notdir $(wildcard $(IMGSDIR)/*.png))
JPGS=$(notdir $(wildcard $(IMGSDIR)/*.jpg))
```

La fonction `wildcard` permet d'obtenir la liste des fichiers contenu dans le dossier `$(IMGSDIR)` tandis que la fonction `notdir` supprime la partie « répertoire » de chacun des fichiers. Finalement la variable `PNGS` contiendra :

```
i.png j.png
```

et `JPGS` :

```
k.jpg
```

On peut ensuite à partir de ces deux variables créer la liste des fichiers Eps à fabriquer (rappelez-vous qu'ils doivent résider dans un répertoire à part) :

```
IMG2EPSS=$(patsubst %, $(EPSSDIR)/%, \
$(PNGS:.png=.eps) $(JPGS:.jpg=.eps))
```

L'expression à droite de l'affectation permet de changer l'extension en `eps` dans les deux listes précédentes et de préfixer chaque nom par le répertoire de stockage des fichiers Eps. `IMG2EPSS` contiendra donc :

```
Epss/i.eps Epss/j.eps Epss/k.eps
```

Cette liste constitue les « prérequis » (au sens de `make`) pour fabriquer les images. On définit donc la cible `figs` comme suit :

```
figs : $(IMGS2EPSS)
```

Il faudra également expliquer à make comment on peut fabriquer un fichier au format postscript encapsulé à partir d'une image. Ceci pourra s'écrire à l'aide de la règle suivante :

```
$(EPSSDIR)/%.eps : $(IMGSDIR)/%.png
↳ convert $< EPS:$@
$(EPSSDIR)/%.eps : $(IMGSDIR)/%.jpg
↳ convert $< EPS:$@
```

qui précise qu'on utilisera l'utilitaire `convert`⁶ pour convertir un fichier Png ou Jpeg en Eps.

5.5.2 Convertir les fichiers de dessin

La conversion des fichiers de dessins suit exactement le même principe. Supposons qu'on dispose de sources au format Fig provenant de xfig et au format Svg provenant de Inkscape. On aura alors dans le makefile :

```
FIGS=$(notdir $(wildcard $(FIGSDIR)/*.fig))
SVGS=$(notdir $(wildcard $(FIGSDIR)/*.svg))
FIGS2EPSS=$(patsubst %, $(EPSSDIR)/%, \
$(FIGS:.fig=.eps) $(SVGS:.svg=.eps))
```

Les règles de conversion sont bien évidemment différentes puisqu'elles font appel l'une à `fig2dev` (utilitaire connexe à xfig) et à Inkscape lui-même pour l'autre :

```
$(EPSSDIR)/%.eps : $(FIGSDIR)/%.fig
↳ fig2dev -L eps $< > $@
$(EPSSDIR)/%.eps : $(FIGSDIR)/%.svg
↳ inkscape -E $@ $<
```

La cible permettant de fabriquer les fichiers de dessins et les images devient finalement :

```
figs : $(IMGS2EPSS) $(FIGS2EPSS)
```

6. Du package ImageMagick disponible à <http://www.imagemagick.org>, qui convertit à peu près tout format d'images

5.6 À part ça

On peut trouver un grand nombre d'extensions permettant de produire des graphiques correspondant à un besoin particulier (arbres, circuits électroniques, histogrammes,...). Vous pouvez en effet avoir besoin, un jour, de générer des graphiques de manière automatique à partir d'une commande. Jetez alors un coup d'oeil sur les différentes extensions disponibles (`pstricks`, `METAPOST`,...) ainsi que sur l'environnement `picture` et ses extensions `epic` et `eepic`. Bon courage !