Introduction à LaTeX

Thibaut Cousin <physique@nodoka.eu>

11 mai 2023

Table des matières

Chapitre 1

Concepts généraux et installation

1.1 Présentation

Tout d'abord, LaTeX se prononce « latèque » car les trois dernières lettres sont en fait les lettres grecques « tau, epsilon, chi » : $\tau\epsilon\chi$.

Techniquement, LaTeX n'est pas un traitement de texte, c'est un *système typographique* : un logiciel de mise en forme de document. La différence est la même qu'entre Word et un logiciel d'édition comme InDesign : il n'est pas là pour vous permettre de faire n'importe quoi, mais pour être conforme à des règles de présentation. Il s'efforce de séparer fond et forme :

- Le fond est le texte tapé directement par vous.
- Vous ne gérez pas directement la forme, mais informez LaTeX du rôle de chaque partie de votre texte (ceci est un paragraphe, ceci est important, ceci est une équation…) et LaTeX se débrouille pour réaliser la mise en forme.

L'approche KISS (*Keep It Simple, Stupid !*)

En fait, LaTeX offre aussi les outils nécessaires pour intervenir manuellement et en profondeur sur la forme. Ma logique dans ce document est d'éviter à tout prix d'y toucher, et c'est aussi comme ça que je procède dans ma pratique personnelle de ce logiciel. C'est dans ce contexte que LaTeX peut être vu comme simple d'emploi.
Contrairement à ce qu'on entend parfois, il n'est pas nécessaire de bidouiller ou de programmer LaTeX pour en tirer d'excellents résultats.

Une fois les bases de LaTeX assimilées, tout devient très simple puisqu'il n'y a plus à gérer les problèmes de mise en page, etc. LaTeX garantit la cohérence du document puisqu'il la contrôle entièrement à votre place.

1.2 Installation

L'installation est en deux parties : une distribution LaTeX (qui contient LaTeX lui-même) et un éditeur de texte qui servira à saisir vos documents.[1]

1.2.1 La distribution LaTeX

Une distribution LaTeX est un bloc assez conséquent qui occupera plusieurs gigaoctets. La plus répandue est TeX Live, disponible gratuitement sur https://www.tug.org/texlive/ et mise à jour une fois par an. Elle émane du TUG (TeX User Group), la principale organisation internationale travaillant à l'utilisation et au développement de notre système typographique favori.

Je recommande fortement TeX Live dans tous les cas, mais si vous êtes sous Windows et que vous rencontrez des problèmes avec cette distribution, vous pouvez tenter votre chance avec MikTeX.[2] C'est une distribution de bien plus faible envergure, moins sécurisée et moins souple d'utilisation, mais elle peut convenir pour à peu près tous les usages.

[1] Si vous faites de la programmation, vous reconnaîtrez la démarche : on installe d'un côté le langage et de l'autre un éditeur.
[2] https://miktex.org/

1.2.2 Les éditeurs

Le choix d'un éditeur de texte est très personnel mais voici mes recommandations :[3]

- Que ce soit sous Linux, macOS ou Windows, TEXstudio[4] est un excellent choix, que l'on peut adapter à ses habitudes avec beaucoup de souplesse.
- Sous macOS, TEXshop[5] est fourni avec TEX Live et se distingue par son interface multi-fenêtres.
- Sous Linux, il y a pléthore de choix ! Citons Kile,[6] Gnome-LATEX,[7] ou encore TEXmaker.[8]
- Si vous avez de l'expérience en programmation, vous connaissez probablement Visual Studio Code. Son extension LATEX Workshop est aussi performante que les éditeurs ci-dessus.[9]

Retenez que votre travail est indépendant de l'éditeur : vous pouvez librement changer d'éditeur sans avoir à retoucher vos documents.

1.3 Document et fichiers

1.3.1 Séparation entre source et résultat

Une différence importante avec un traitement de texte est que, quand vous rédigez un document avec LATEX, tout ne se passe pas dans un seul et unique fichier.

- Ce que vous tapez est enregistré dans un (ou plusieurs) fichier en « texte brut », appelé fichier source. Son extension est `.tex`, mais c'est en fait la même chose qu'un simple fichier en `.txt`.
  Ce fichier peut donc être ouvert, manipulé et enregistré dans n'importe quel éditeur de texte, mais les éditeurs spécialisés mentionnés ci-dessus vous faciliteront la vie en tapant certaines choses à votre place ou en signalant les erreurs de syntaxe.
- LATEX lui-même entre en jeu quand vous voulez produire le document définitif mis en page, au format PDF : il parcourt le fichier source et génère le PDF au fur et à mesure.
- Au cours de l'opération de conversion (appelée *compilation*), des fichiers temporaires sont créés, de types variés selon ce qu'on fait. Ces fichiers peuvent être supprimés une fois le document PDF créé.[10] Par contre, les fichiers sources et les images seront nécessaires à chaque fois que vous voudrez reconstruire le PDF. Ils doivent donc être précieusement conservés.

Si vous voulez insérer des images dans votre document, ces images doivent être des fichiers séparés que LATEX intègre au PDF au moment de sa création.

Format du fichier `.tex`

Ce document suppose (et encourage !) l'usage systématique du format Unicode, ou UTF-8, dans vos fichiers textes. Ce format universel permet de taper directement au clavier les caractères et symboles de toutes les langues, et qu'ils soient correctement reconnus lors de la fabrication du PDF.
Tout éditeur de texte moderne devrait être configuré par défaut pour utiliser ce format. TEX Live lui-même utilise ce format par défaut depuis quelques années déjà.

1.4 Générer le document final PDF

1.4.1 Choix d'un compilateur

LATEX existe depuis 1983, et a acquis sa forme actuelle (la version 2ε) en 1994. Il est donc passé par de nombreuses évolutions. Encore aujourd'hui, on rencontre plusieurs « branches cousines » de LATEX.

La plus connue est certainement PDFLATEX. Comme son nom l'indique, elle est conçue pour produire des documents PDF. Malheureusement, elle a beaucoup vieilli : sa gestion des polices de caractères est déficiente et obsolète, sa gestion des langues autres que l'Anglais est de qualité variable et mal standardisée…

[3] TEXworks est pré-installé avec TEX Live pour toutes les plateformes, mais il est rudimentaire et je recommande plutôt TEXstudio.
[4] https://www.texstudio.org/
[5] https://pages.uoregon.edu/koch/texshop/
[6] https://kile.sourceforge.io/
[7] https://wiki.gnome.org/Apps/GNOME-LaTeX
[8] http://www.xm1math.net/texmaker/ disponible aussi pour macOS et Windows.
[9] https://code.visualstudio.com/ et https://marketplace.visualstudio.com/items?itemName=James-Yu.latex-workshop
[10] Tout éditeur digne de ce nom doit proposer une fonctionnalité pour effectuer ce nettoyage.

LuaLaTeX

Dans le présent document, j'ai choisi de vous recommander LuaLaTeX. Ce compilateur est en train de devenir « le » standard, et est aujourd'hui parfaitement à maturité. C'est celui que j'utilise moi-même pour l'ensemble de mes travaux, qu'ils soient scientifiques ou non, y compris pour des documents multilingues, depuis plusieurs années.
Il est intégré à TeX Live, donc vous n'avez rien de spécifique à installer.

Dans l'immense majorité des cas, le choix du compilateur n'a pas d'effet sur la syntaxe de LaTeX.

Parmi les avantages des compilateurs modernes, citons la liberté de saisir directement les caractères régionaux de n'importe quelle langue[11] ou d'utiliser les polices de caractères installées dans votre système d'exploitation (et donc en particulier les polices TTF, le format le plus répandu).

1.4.2 Compiler un document

Votre éditeur de texte doit proposer un bouton ou un raccourci clavier pour déclencher la compilation du document, et donc la production du PDF. Quelques remarques :

- Dans certains cas, plusieurs compilations sont nécessaires. Votre éditeur devrait gérer cela automatiquement, ou au moins vous prévenir.
- Une faute de syntaxe provoque l'arrêt immédiat de la compilation. LaTeX n'essaie pas de produire le document au-delà de l'erreur.

[11] Un caractère régional est un caractère qui existe dans une langue mais pas en Anglais. Cela va des lettres accentuées aux idéogrammes, en passant par les alphabets non romains.

Chapitre 2

Le mode texte

Bien qu'il soit rarement présenté dans ce sens, LaTeX convient tout à fait pour taper du texte « pur », sans contenu scientifique. D'ailleurs, il distingue de manière tranchée le « mode texte » (sous-entendu, sans écriture mathématique) et le « mode mathématique ».

Commençons donc par discuter le mode texte. La saisie de symboles ou d'équations sera abordée dans le chapitre 3.

2.1 Structure générale d'un document

Un document est donc constitué d'un ou plusieurs fichiers textes avec l'extension `.tex`. À l'intérieur, les choses ressemblent beaucoup à du HTML : le texte est tapé au kilomètre, des commandes y sont intercalées pour intervenir sur la mise en forme.

Syntaxe

Concernant les commandes :

- Elles commencent toujours par le caractère \.
- Elles sont sensibles à la casse : respectez majuscules et minuscules.
- Si une commande prend des arguments, les arguments obligatoires sont entre accolades { et }, et les arguments facultatifs entre crochets [et].

Un *environnement* est un bloc de texte délimité par `\begin{environnement}` et `\end{environnement}`
Un document comporte un préambule et un corps de texte :

- Le préambule commence par `\documentclass` et se termine quand le corps de texte commence.
- Le corps de texte commence par `\begin{document}` (autrement dit c'est un environnement appelé *document*).

Une ligne commençant par % sera ignorée (mise en commentaire).

2.2 Le préambule

2.2.1 Vue d'ensemble

Le préambule vient en premier dans le document, c'est l'endroit où vous expliquez à LaTeX vos intentions : type de document, choix de la langue (pour l'application des règles typographiques), etc.

En outre, il existe tellement de commandes LaTeX couvrant des besoins très divers qu'il n'est pas raisonnable de les mettre toutes à disposition en même temps. Beaucoup sont alors regroupées dans des *paquetages*, typiquement de manière thématique.[1] C'est dans le préambule du document que vous devez indiquer les paquetages dont vous avez besoin.

En général le contenu de cet préambule change très peu d'un document à l'autre. Je vous propose donc ci-dessous un préambule couvrant une gamme relativement large de besoins et que vous pourrez recopier dans tous vos documents.[2]

[1] À la manière des bibliothèques dans les langages de programmation.

[2] Voir paragraphe 5.4 pour savoir comment déplacer la majeure partie de ce préambule dans un fichier unique qui sera utilisé par tous vos documents.

```
% !TeX TS-program = lualatex
% !TeX encoding = UTF-8
% !TeX spellcheck = fr_FR

\documentclass[a4paper, french, twoside]{article}
\usepackage{polyglossia}
\setdefaultlanguage{french}
\usepackage{fontspec}
\defaultfontfeatures{Ligatures={Common, TeX}}
\usepackage{mathtools, amssymb}
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage[left=2cm, right=2cm, top=2cm, bottom=2cm]{geometry}
\usepackage{enumitem}
\setlist{noitemsep}
\usepackage{tabularray}
\usepackage[pdfusetitle, colorlinks=true]{hyperref}

\title{Titre du document}
\author{Nom de l'auteur}
\date{Date du document}
```

Comme il s'agit de texte brut, les couleurs ci-dessus ne sont là que pour le confort de lecture. Les paragraphes ci-dessous détaillent le sens de ces commandes…

2.2.2 Les « macros magiques » (facultatives)

Le symbole % désignant un commentaire, les trois premières lignes sont totalement ignorées par LaTeX. Mais elles sont comprises par la plupart des éditeurs de textes spécialisés, et permettent de lui faire passer des éléments de configuration propres au document :

- `TS-program` spécifie le compilateur à utiliser.
- `encoding` signale que le document est au format Unicode.
- `spellcheck` précise la langue du document, pour que l'éditeur utilise le bon dictionnaire avec son correcteur orthographique.

Si votre éditeur ne comprend pas une macro magique, il va simplement l'ignorer (puisque c'est un commentaire).

Leur usage est totalement optionnel, mais assure que votre éditeur choisira la « bonne » configuration quand il manipulera et compilera votre document. Si vous avez besoin de jongler entre plusieurs configurations selon les documents, cela vous évitera de devoir reconfigurer votre éditeur à chaque fois.

2.2.3 Le *documentclass*

Il indique le type de document et doit se trouver avant toute autre commande. Il existe une certaine diversité de types, mais on va s'en tenir ici à deux :

- `article`, prévu pour des documents courts, sans saut de page, et utilise une numérotation légère des sections et paragraphes car il n'y a pas de chapitre.
- `book` est pour les documents longs. Il peut y avoir un prologue et un épilogue, des sauts de page voire des pages blanches (pour que les chapitres commencent toujours sur une page de droite par exemple), la numérotation des parties est plus riche… Le présent document a été fait avec cette classe.

Trois options facultatives ont été passées à la classe dans l'exemple ci-dessus :

- `a4paper` indique le format papier (A4),
- `french` indique l'usage de la langue française,
- `twoside` indique le mode recto-verso (pour le réglage de certaines marges).

2.2.4 Les paquetages

La commande `\usepackage{}` sert à déclarer un paquetage dont on veut utiliser les commandes. Dans le modèle de préambule ci-dessus, voici un aperçu de des paquetages mis en jeu :

- `polyglossia` gère les particularités typographiques des diverses langues. La commande sur la ligne suivante, `\setdefaultlanguage{}`, spécifie la langue du document, et les règles de mise en page en découlent.[3] On peut aussi changer de langue en cours de route dans le document, et les règles typographiques suivront. Dans le cas du Français, l'indentation de la première ligne d'un paragraphe est gérée automatiquement.
- `fontspec` gère l'utilisation des polices de caractères et fournit les commandes pour choisir une police ou en changer. La commande `\defaultfontfeatures{}` active l'utilisation des ligatures, conformément aux règles typographiques françaises.
- `mathtools` et `amssymb` ajoutent un grand nombre de commandes mathématiques utiles (voir section 3).
- `xcolor` fournit des commandes gérant la couleur. Voir la section 2.4.6.
- `graphicx` fournit les commandes d'insertion de figure. Voir la section 2.8.
- `geometry` permet de régler les marges. Il récupère l'option `a4paper` du *documentclass* et l'applique. Les autres options ajustent les marges.
- `enumitem` permet de configurer les listes à puces, et la commande `\setlist{noitemsep}` réduit l'espacement vertical entre puces, que je trouve trop large. Là encore, c'est purement une affaire de goût.
- `tabularray` fournit une interface moderne et efficace pour construire des tableaux.
- `hyperref` permet d'obtenir un document PDF navigable, avec des liens cliquables pointant soit vers des ressources internet, soit vers d'autres parties du document.[4]

Un univers de paquetages

La raison pour laquelle une distribution LaTeX est si volumineuse est l'énorme diversité de paquetages qui vient avec. Explorer cette jungle de possibilités fait partie du charme de LaTeX, mais on peut facilement s'y perdre. Au moins dans un premier temps, n'en abusez pas et demandez conseil.
Les paquetages LaTeX sont centralisés sur le site du CTAN (`https://ctan.org/`). Son moteur de recherche n'est pas très bon, c'est surtout un outil pratique pour trouver la documentation d'un paquetage dont vous connaissez le nom.

2.2.5 Méta-données du document

`\title{}`, `\author{}` et `\date{}` spécifient respectivement le titre, l'auteur et la date du document. Pour la date, vous pouvez saisir une date à la main, ou utiliser `\today` qui affiche automatiquement la date du jour. Ces informations sont récupérées par `\maketitle` (voir section 2.3.1).

2.3 Le corps de texte

2.3.1 L'environnement *document*

Le corps de texte est entièrement contenu dans un environnement *document*.[5] C'est là que vous devez saisir tout ce que vous voulez voir apparaître dans votre document, sous la forme d'un texte agrémenté de commandes pour la mise en page. Modèle :

```
\begin{document}
\maketitle

Tapez votre texte ici

\end{document}
```

`\maketitle` est optionnelle et récupère les informations `\title{}`, `\author{}` et `\date{}` du préambule pour générer un titre. L'apparence diffère selon le type de document.

2.3.2 Structure d'un document

Les parties et sous-parties de votre document sont indiquées par les commandes suivantes, de la plus générale à la plus spécifique :

[3] Pour des raisons trop complexes pour être détaillées ici, le mot `french` passé en argument de `documentclass` est ignoré par `polyglossia`. Mais il reste utile car il peut être compris par d'autres paquetages qui s'adaptent à la langue.
[4] Le présent document en contient de multiples exemples.
[5] LaTeX ignore tout ce qui se trouve au-delà, puisque par définition la fin de cet environnement marque la fin du document.

- \part{}
- \chapter{} (n'existe pas pour le type *article*)
- \section{}
- \subsection{}
- \subsubsection{}

Leur utilisation provoque l'affichage automatique d'un titre numéroté, et son ajout à la table des matières. L'apparence exacte du résultat dépend du type de document.

Gestion automatique de la numérotation

LaTeX gère automatiquement toutes les numérotations, que ce soit pour les pages, les paragraphes, les figures, les équations, les pistes à puces…
À aucun moment vous ne devez avoir à taper vous-même un tel numéro. Voir section 2.7.

2.3.3 Sections optionnelles

LaTeX dispose d'outils pour générer diverses sections optionnelles comme un prologue, un index ou une bibliographie. Mentionnons seulement ici la table des matières.

Une unique commande \tableofcontents provoque l'affichage de la table des matières, générée de manière entièrement automatique à partir des commandes de structure précédentes.

2.4 Taper du texte

Le texte est tapé au kilomètre et les espacements sont gérés par LaTeX :

- Les passages à la ligne sans laisser de ligne blanche sont ignorés.
- Une ligne laissée blanche marque un nouveau paragraphe. Plusieurs lignes blanches seront traitées comme une seule.
- De même, si vous laissez plusieurs espaces dans une ligne, elles[6] seront traitées comme une seule.

Grâce à tout cela, vous pouvez aérer votre texte en le tapant sans que cela change la mise en page finale.

2.4.1 Caractères spéciaux

Les caractères utilisés pour les commandes LaTeX ne peuvent pas être affichés directement :

```
$ & # { } _ \ ^ ~ %
```

La plupart de ces caractères peuvent être affichés en les tapant précédées de \ (exemple : \% pour %).

Le tilde ~ est l'espace insécable : il ne peut pas y avoir césure sur elle. En Français, il faut normalement en mettre une avant les symboles de ponctuation double (deux points et point virgule).[7]

Par ailleurs, le Français a quelque caractères qui ne sont pas accessibles sur tous les claviers. Si nécessaire :

Caractère	Commande	Exemple	Résultat
œ	\oe	mon \oe il	mon œil
^{e}	\ieme{}	au 19\ieme{} siècle	au 19^{e} siècle
…	\dots		

2.4.2 Style de caractère

Les commandes suivantes sont disponibles pour changer le style. Voici les principales :

```
\textit{texte en italique}        texte en italique
\textbf{texte en gras}            texte en gras
\texttt{texte machine à écrire}   texte machine à écrire
```

[6] En typographie, *espace* est féminin !
[7] En pratique, c'est rarement nécessaire, LaTeX le gère plutôt bien tout seul.

2.4.3 Taille des caractères

Les commande suivantes changent la taille des lettres à partir du point où elles sont utilisées. Il faut donc une autre commande plus loin pour revenir à la taille précédente :

`\tiny`	De plus en plus grand
`\scriptsize`	De plus en plus grand
`\footnotesize`	De plus en plus grand
`\small`	De plus en plus grand
`\normalsize`	De plus en plus grand
`\large`	De plus en plus grand
`\Large`	De plus en plus grand
`\LARGE`	De plus en plus grand
`\huge`	De plus en plus grand
`\Huge`	De plus en plus grand

Quand vous utilisez une de ces commandes, elle agit sur tout ce qui la suit. Si vous voulez circonscrire son action à un passage, délimitez ce passage entre `{` et `}`.

```
Normal, {\tiny petit}, normal
```
Normal, petit, normal

2.4.4 Changer de police de caractères

Vous pouvez utiliser les polices installées dans votre système d'exploitation. Par exemple :

```
\setmainfont{Times New Roman}
```

placé dans le préambule basculera vers une police bien connue des utilisateurs de Word. Charge à vous de vérifier que la police choisie est bien installée et contient tous les caractères dont vous avez besoin !

2.4.5 Soulignement

```
\underline{Ceci est souligné}
```
Ceci est souligné

2.4.6 Couleur

Pour changer la couleur d'un groupe de mots :

```
Ceci est \textcolor{red}{en couleur}
```
Ceci est en couleur

Les couleurs usuelles sont prédéfinies avec leur nom anglais (`red`, `yellow`, `green`, `blue`, `gray`…)

2.5 Mise en page

La justification du texte à gauche et à droite, ainsi que la césure suivant les règles de la langue utilisée, sont automatiques.

2.5.1 Listes à puces

Pour faire des listes à puces, il y a deux environnements courants.
Pour les listes à puces non numérotées :

```
\begin{itemize}
\item Blabli
\item Blabla
\end{itemize}
```

- Blabli
- Blabla

Pour les listes numérotées, remplacez `itemize` par `enumerate`. Le résultat sera alors :

```
\begin{enumerate}
\item Blabli
\item Blabla
\end{enumerate}
```

1. Blabli
2. Blabla

Il est possible d'imbriquer des listes à puces, LaTeX gérant alors automatiquement l'indentation.

2.5.2 Centrage

Pour centrer quelque chose, mettez-le dans un environnement `center` :

```
\begin{center}
Ceci est un texte centré
\end{center}
```

donne :

Ceci est un texte centré

2.5.3 Notes de bas de page

Le présent document contient de nombreuses notes de bas de page ! À l'endroit où doit apparaître le renvoi, utilisez la commande :

```
\footnote{Texte de la note de bas de page.}
```

et l'argument de la commande apparaîtra en bas de la page. La note recevra automatiquement un numéro.

2.6 Tableaux

Faire des tableaux en LaTeX est un peu déroutant au début,[8] mais c'est assez visuel en exploitant la liberté d'aller à la ligne et de laisser des espaces sans affecter la mise en page. Exemple :[9]

```
\begin{tblr}{colspec={lcc}, vline{2, 4}, hline{2}}
Colonne 1  & Colonne 2 & Colonne 3 \\
Intitulé 1 & 1         & 2 \\
Intitulé 2 & 3         & 4
\end{tblr}
```

ce qui donne :

Colonne 1	Colonne 2	Colonne 3
Intitulé 1	1	2
Intitulé 2	3	4

En clair, un tableau est un environnement appelé `tblr`. Ce dernier, dans ses arguments obligatoires, peut recevoir plusieurs paramètres. Sur cet exemple :

- `colspec` spécifie la description des colonnes. Chaque colonne est désignée par `l`, `c` ou `r` (alignement horizontal à gauche, centré ou à droite).
- Dans chaque ligne du tableau, les cases sont séparées par `&` et le passage à la ligne suivante est indiqué par `\\`.
- `vline` indique, ici, qu'il y aura un trait vertical avant les colonnes 2 et 4. Pour dire « toutes les colonnes », utilisez `vlines` à la place.
- `hline` indique, ici, qu'il y aura un trait horizontal avant la ligne 2. Pour dire « toutes les lignes », utilisez `hlines` à la place.

[8] Certains éditeurs de texte proposent un assistant de saisie pour ça.

[9] En fait, LaTeX propose l'environnement `tabular` par défaut, mais j'ai choisi de vous faire utiliser le paquetage `tabularray`, qui fournit l'environnement `tblr`, plus moderne et permettant une séparation du fond et de la forme.

2.7 Références

Si vous voulez faire référence à quelque chose situé ailleurs dans le texte, placez `\label{mot-clef}` à côté de ce quelque chose. Cela ne provoque aucun affichage. Bien sûr le mot-clef doit être unique. Ensuite :

- `\ref{mot-clef}` est placé à l'endroit où la référence apparaît. Selon la position du `\label{}` correspondant, il provoque l'affichage du numéro de section, numéro de figure, etc. Exemple :

```
\section{Entrer en mode mathématique}\label{sec:math}
Pour les équations, voir la section \ref{sec:math}.
```

  qui donne : « Pour les équations, voir la section 3. »
- `\pageref{mot-clef}` fonctionne comme `\ref{mot-clef}` mais affiche le numéro de page correspondant.
- `\eqref{mot-clef}` remplace `\ref{mot-clef}` pour les équations (voir section 3.4.8).

Les références seront automatiquement cliquables pour naviguer rapidement dans le document PDF grâce au paquetage `hyperref`.

2.8 Insertion de figures

Quand vous écrivez deux paragraphes l'un après l'autre, LaTeX peut librement les disposer sur une ou plusieurs pages, mais il en préservera évidemment l'ordre.
À l'inverse, un *flottant* est un objet que LaTeX peut déplacer plus librement. En pratique, ce sont surtout les figures que l'on fait flotter. L'idée est que les figures sont numérotées, de sorte qu'on peut s'y référer par leur numéro même si elles sont placées plus loin.

Pour insérer une figure, la commande fondamentale est `\includegraphics`. En général, cette commande est placée dans un environnement `figure` qui la rend flottante :[10]

```
\begin{figure}[!htb]
\centering
\includegraphics[width=4cm]{misa.jpg}
\caption{\label{fig:uneréférence}Misa.}
\end{figure}
```

où, dans cet exemple, `misa.jpg` est le nom de fichier de l'image, qui doit se trouver dans le même dossier que le fichier LaTeX.[11] Ce code donne la figure 2.1.

FIG. 2.1 : Misa.

Vous pouvez insérer des figures aux formats PDF, JPEG, PNG et TIFF. La syntaxe est :

- L'option `!htb` informe LaTeX de vos préférences pour faire flotter la figure. Je vous encourage à toujours l'utiliser exactement ainsi : `htb` veut dire « here, top, bottom » et indique, par ordre de préférence, où LaTeX doit s'efforcer de placer la figure (ici, en haut de la page, en bas de la page).[12]

[10] Rassurez-vous, tous les éditeurs de texte permettent la saisie automatique de tout ou partie des commandes (voir paragraphe 5.3).
[11] Voir paragraphe 5.1.2 pour un moyen de concentrer les images dans un dossier séparé.
[12] LaTeX suit une règle typographique qui limite l'occupation de la surface d'une page par des figures et refuse par défaut de mettre trop de figures sur une page. Alors, `!` permet justement d'outrepasser cette règle.

- `\centering` demande de centrer la figure sur la largeur de la page.
- L'option `width` permet de spécifier la largeur de la figure. Si vous préférez, il y aussi une option `height` pour spécifier la hauteur.
- `\caption` s'occupe automatiquement de la numérotation de l'image et de l'affichage d'une légende.
- `\label{}` sert à pouvoir faire référence à la figure ailleurs dans le document (voir section 2.7).

Si vous voulez vraiment imposer de manière stricte le placement d'une figure, c'est facile : il suffit qu'elle ne soit pas flottante. Voir section 4.1.2.

Chapitre 3

Le mode mathématique

LaTeX distingue fortement les passages mathématiques et non mathématiques :

- LaTeX utilise ses propres polices en mode mathématique, indépendamment de celles du mode texte.
- La typographie est différente en « mode texte » et en « mode mathématique », conformément aux règles internationales de typographie.
- Les commandes mathématiques ne sont pas disponibles en mode texte.
- Un certain nombre de caractères, en particulier l'alphabet grec, sont disponibles par des commandes spécifiques en mode mathématique, par exemple `\alpha` pour α.

L'écriture de formules n'est pas aussi compliquée qu'elle en a l'air. Il suffit de souvent de prononcer la formule à haute voix pour savoir dans quel ordre écrire les commandes.

3.1 Conventions typographiques

La typographie d'une équation

Un petit aperçu des règles en usage :

- Par défaut, un caractère est une variable et est automatiquement écrit en italique. Exemple : x.
- Plusieurs caractères à la suite ne sont pas un mot, mais un produit de variables, donc l'espacement est élargi. Comparez par exemple : ijk (mode texte) et ijk (mode mathématique).
- Un nom d'opérateur est un mot et ne doit pas être en italique, donc LaTeX fournit des commandes pour tous les opérateurs usuels. Par exemple pour le cosinus, `\cos` donne : cos.
- Ne pas laisser de ligne blanche. Le passage à la ligne doit être annoncé par `\\`.
- Les espaces sont ignorées, mais il y a des commandes d'espacement (voir section 3.3).

Le point concernant les lignes blanches est important ! En mode mathématique, une ligne vide n'est pas ignorée, elle est traitée comme une erreur de syntaxe et donc bloque la compilation. Si vous voulez réellement laisser un espacement vertical entre deux éléments dans ce mode, utilisez la commande `\vspace{1cm}` (la valeur est un exemple).

3.2 Entrer en mode mathématique

3.2.1 Dans le fil du texte

Pour insérer des symboles mathématiques ou de petites équations dans le fil du texte, on entre en mode mathématique avec `$` et on en sort avec un second `$`. Exemple :

```
Traçons la courbe $y=x^2$ pour $x$ variant entre 0 et 2.
```

donne : Traçons la courbe $y = x^2$ pour x variant entre 0 et 2.

3.2.2 Équations isolées du texte

Il faut ouvrir un environnement mathématique, le plus souvent `align`.[1] Exemple :

[1] Il existe de nombreux environnements mathématiques, mais celui-ci devrait couvrir la grande majorité des cas.

```
\begin{align}
f(x) = \int_0^1 g(x,y) dy
\end{align}
```

donne une équation horizontalement centrée :

$$f(x) = \int_0^1 g(x,y)dy \tag{3.1}$$

Notez le numéro en bout de ligne à droite : `align` génère automatiquement une numérotation pour chaque ligne d'équation. Si vous n'en voulez pas, utilisez `align*` à la place.

Rappelons en outre que les espaces placés dans le code source (autour du égal, avant le dy, etc) sont totalement ignorés en mode mathématique. Ils ne sont là que pour le confort visuel de l'humain.

3.3 Espacement

Par défaut en mode mathématique, LaTeX élimine toute espace horizontale. Pour forcer une telle espace, de la plus étroite à la plus large :

`a\,b`	$a\,b$
`a\ b`	$a\ b$
`a\quad b`	$a \quad b$
`a\qquad b`	$a \qquad b$

3.4 Écrire des expressions mathématiques

3.4.1 Les commandes de base

Il existe un nombre incalculable de commandes mathématiques ! Limitons-nous à l'essentiel :

- Exposant : `x^2` donne x^2 (si l'exposant fait plus d'un caractère, il faut le mettre en accolades)
- Indice : `x_2` donne x_2 (si l'indice fait plus d'un caractère, il faut le mettre en accolades)
- Exposant et indice : `x^2_3` donne x^2_3
- Symbole somme : `\sum_{i=0}^n a_n` donne $\sum_{i=0}^{n} a_n$
- Intégrale simple : `\int_a^b f(x)dx` donne $\int_a^b f(x)dx$
- Intégrale double ou triple : `\iint` ou `\iiint`
- Fraction : `\frac{1}{2}` donne $\frac{1}{2}$
- Racine carrée : `\sqrt{2}` donne $\sqrt{2}$
- Opérateurs usuels : saisir leur nom comme une commande : `\sin`, `\log`, `\arcsin`, etc
- Lettres grecques : saisir leur nom comme une commande : `\xi`, `\delta`, etc. Mettre la première lettre en majuscule pour avoir la majuscule : `\Gamma`, `\Lambda`, etc. Deux exceptions : le phi « tout rond » habituel est `\varphi` (φ) et le epsilon avec deux arrondis est `\varepsilon` (ε).
- Vecteur court : `\vec{v}` donne $\vec{v}$. Les lettres i et j devant perdre leur point dans un vecteur, on a les commandes `\imath` et `\jmath`, par exemple `\vec{\jmath}` donne $\vec{\jmath}$.
- Vecteur long : `\overrightarrow{OM}` donne $\overrightarrow{OM}$
- Barre courte : `\bar{x}` donne $\bar{x}$
- Barre longue / distance algébrique : `\overline{AB}` donne $\overline{AB}$
- Flèches simples : `\to` donne $\to$, `\leftrightarrow` donne $\leftrightarrow$
- Flèches doubles : `\Rightarrow` donne $\Rightarrow$, `\Leftrightarrow` donne $\Leftrightarrow$
- Dérivées en notation de Newton : `\dot{x}` donne $\dot{x}$ et `\ddot{x}` donne $\ddot{x}$
- Produit vectoriel : `\wedge` donne $\wedge$ (notation française), `\times` donne $\times$ (notation internationale)
- Inégalités larges : `\geqslant` donne $\geqslant$ et `\leqslant` donne $\leqslant$
- Très grand devant : `\gg` donne $\gg$; très petit devant : `\ll` donne $\ll$

3.4.2 Jeux de caractères

La commande `\mathcal{}` donne accès aux majuscules calligraphiques : `\mathcal{C}` donne $\mathcal{C}$.
La commande `\mathbb{}` donne accès aux majuscules pour les ensembles : `\mathbb{Z}` donne $\mathbb{Z}$.

3.4.3 Les délimiteurs

Les délimiteurs usuels sont :[2]

- parenthèses : tapées directement au clavier, (x)
- crochets : tapés directement au clavier, $[x]$
- accolades : `\lbrace` (ouvrante) et `\rbrace` (fermante), ainsi `\lbrace x\rbrace` donne $\{x\}$
- valeur absolue ou module : `\lvert x\rvert` donne $|x|$
- norme : `\lVert x\rVert` donne $\|x\|$
- moyenne : `\langle x\rangle` donne $\langle x\rangle$

Si leur taille par défaut ne convient pas, les commandes `\left` et `\right` fournissent une taille dynamique, qui s'adapte au contenu. Par exemple :[3]

```
\lvert A\rvert
=\left(
        \frac{m\omega_0}{\pi\hslash}
\right)^\frac{1}{4}
```

donne :

$$|A| = \left(\frac{m\omega_0}{\pi\hbar}\right)^{\frac{1}{4}}$$

3.4.4 Encadrer une formule

En mode mathématique, `\boxed{}` encadre son argument. Attention, celui-ci doit être une unique équation. Par exemple :[4]

```
\boxed{
R=\sqrt{x^2+y^2}
}
```

donne :

$$\boxed{R = \sqrt{x^2 + y^2}}$$

3.4.5 Les matrices

Pour saisir des matrices, la syntaxe est la même que pour un tableau, sauf qu'on ne précise pas le nombre de colonnes (et il faut être en mode mathématique). La syntaxe est donc :

```
\begin{environnement}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{environnement}
```

Selon ce que vous mettez pour `environnement`, voilà le résultat :

`matrix`	`pmatrix`	`bmatrix`	`vmatrix`	`Vmatrix`
$\begin{matrix}1&0&0\\0&1&0\\0&0&1\end{matrix}$	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$	$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$	$\begin{vmatrix}1&0&0\\0&1&0\\0&0&1\end{vmatrix}$	$\begin{Vmatrix}1&0&0\\0&1&0\\0&0&1\end{Vmatrix}$

[2] La plupart des noms de délimiteurs contiennent soit un `l` soit un `r`, pour « left » et « right ». Cela aide LaTeX à déterminer quel délimiteur va avec quel autre. Cela est donc utile même si les deux délimiteurs sont identiques.

[3] Remarquez que j'ai étalé le code sur plusieurs lignes. Ce n'est pas nécessaire, mais c'est, comme toujours, pour le confort de lecture du fichier source.

[4] Là encore, l'étalement de la commande sur plusieurs lignes est optionnel.

3.4.6 Du texte dans les équations

Pour pouvoir taper du texte « normal » (c'est-à-dire typographié en mode texte) au milieu d'une équation, il y a `\text{}`.

```
n=\pm\sqrt{
        1-\frac{\omega_p^2}{\omega^2}
}\qquad\text{à basse fréquence}
```

donne :

$$n = \pm\sqrt{1 - \frac{\omega_p^2}{\omega^2}} \qquad \text{à basse fréquence}$$

3.4.7 Groupes d'équations avec alignement vertical

L'environnement mathématique `align` permet d'avoir plusieurs équations qui se suivent avec un alignement vertical, souvent sur le signe égal. C'est le caractère `&` qui indique sur quoi faire l'alignement, et `\\` qui indique le saut de ligne vers l'équation suivante.

```
\langle\mathcal{P}\rangle
&=\iint_S\langle\vec{\Pi}\rangle.d\vec{S}\\
&=\frac{\omega^4p_0^2}{12\pi\varepsilon_0c^3}
```

donne :

$$\begin{aligned}\langle\mathcal{P}\rangle &= \iint_S \langle\vec{\Pi}\rangle.d\vec{S}\\ &= \frac{\omega^4 p_0^2}{12\pi\varepsilon_0 c^3}\end{aligned}$$

3.4.8 Références aux équations

La commande `\label{}` présentée en section 2.7 peut être placée à côté d'une équation. Ensuite, l'équivalent de `\ref{}` pour y faire référence est `\eqref{}` et s'utilise de la même façon.

Si un environnement `align` contient plusieurs lignes d'équations, chaque ligne reçoit son propre numéro. Veillez alors à placer `\label{}` sur la bonne ligne.

3.4.9 Tableaux mathématiques

En mode mathématique, l'environnement `tblr` vu en section 2.6 reste utilisable, son contenu étant alors automatiquement en mode mathématique aussi.[5]

```
\begin{tblr}{colspec={ccc}, hline{2}}
f(x) & f'(x) & f''(x)\\
x^3 & 3x^2 & 6x\\
\cos(x) & -\sin(x) & -\cos(x)
\end{tblr}
```

qui produit le résultat :

$f(x)$	$f'(x)$	$f''(x)$
x^3	$3x^2$	$6x$
$\cos(x)$	$-\sin(x)$	$-\cos(x)$

(3.2)

[5] Historiquement, le paquetage `mathtools` provoque la mise à disposition de l'environnement `array`, qui sert justement à faire des tableaux en mode mathématique. Là encore, j'ai choisi de préconiser plutôt `tblr`.

3.4.10 Résultats regroupés par une accolade

Pour présenter une série de résultats avec une grande accolade à gauche, utilisez l'environnement `dcases`.

```
x=(-1)^n
\quad\Rightarrow\quad
\begin{dcases}
x=1  & \text{si $n$ est pair}\\
x=-1 & \text{si $n$ est impair}
\end{dcases}
```

donne :

$$x=(-1)^n \quad \Rightarrow \quad \begin{cases} x=1 & \text{si } n \text{ est pair} \\ x=-1 & \text{si } n \text{ est impair} \end{cases}$$

L'utilisation du `&` et de ce qui le suit est optionnelle.

Chapitre 4

Des cerises sur le gâteau...

Ce chapitre compile quelques commandes un peu plus avancées, qui répondent à des besoins plus spécifiques sans technicité excessive.

4.1 Compléments sur l'insertion de figures

4.1.1 Options d'affichage

En section 2.8, nous avons vu la commande `\includegraphics{}` qui réalise l'insertion d'un fichier image, avec ses principales options `width` et `height`. Voici quelques autres options :

```
\begin{figure}[!htb]
\centering
\includegraphics[angle=45, scale=0.8]{midori.jpg}
\caption{\label{fig:uneréférence}Tatematsu Midori.}
\end{figure}
```

Ceci donne la figure 4.1.

FIG. 4.1 : Tatematsu Midori.

L'option `angle=45` fait tourner la figure de 45° dans le sens inverse des aiguilles d'une montre et `scale=0.8` provoque une mise à l'échelle, ici une réduction à 80 % de la taille d'origine.

4.1.2 Figure non flottante

Comme c'est l'environnement `figure` qui rend une figure flottante, il suffit de ne pas l'utiliser si vous avez besoin de maîtriser strictement la position d'une figure :

```
\includegraphics[width=4cm]{misa.jpg}
```

La seule subtilité est que, si vous avez besoin d'attribuer une légende et une référence à la figure, la commande `\caption` ne fonctionne qu'avec un flottant. Donc ici il faut la remplacer par `\captionof` :

```
\includegraphics[width=4cm]{misa.jpg}
\captionof{figure}{\label{fig:uneréférence}Misa.}
```

4.1.3 Plusieurs figures côte-à-côte

Si vous voulez placer plusieurs figures dans le même environnement `figure`, pour qu'elles soient côte-à-côte ou les unes en dessous des autres, ajoutez à votre préambule ces deux lignes :

```
\usepackage{subcaption}
\captionsetup[sub]{subrefformat=brace}
```

Ceci met à votre disposition un environnement `subfigure` qui s'utilise à l'intérieur de l'environnement `figure`, un par figure. Chaque `subfigure` a sa propre largeur, ses règles d'alignement et sa numérotation. Une commande `subref` est alors fournie pour la numérotation des sous-figures.

Voici l'exemple de deux figures côte-à-côte :

```
\begin{figure}[!htb]
\centering
        \begin{subfigure}[b]{0.45\textwidth}
        \centering
        \includegraphics[height=4cm]{nita.jpg}
        \caption{\label{fig:nita}}
        \end{subfigure}
\quad
        \begin{subfigure}[b]{0.45\textwidth}
        \centering
        \includegraphics[height=4cm]{tosin.jpg}
        \caption{\label{fig:tosin}}
        \end{subfigure}
\caption{\subref{fig:nita} Nita Strauss. \subref{fig:tosin} Tosin Abasi.}
\end{figure}
```

Cela donne les figures 4.2a et 4.2b.

(a)

(b)

FIG. 4.2 : a) Nita Strauss. b) Tosin Abasi.

Bien sûr, taper une telle liste de commandes peut vous paraître lourd, mais en pratique votre éditeur devrait tout taper pour vous, soit via un assistant, soit via une macro (voir paragraphe 5.3). En résumé :

- Notez que chaque figure reçoit un « sous-numéro » sous la forme d'une lettre.
- Chaque figure se trouve dans un environnement `subfigure`.
- L'argument optionnel `b` (bottom) indique que la figure doit être placée le plus bas possible (bord inférieur de la figure sur le bord inférieur de l'environnement).
- L'argument obligatoire `0.45\textwidth` signifie « 45 % de la largeur du texte », ajustez le chiffre à votre goût. Cela correspond à l'espace disponible pour la figure, pas à la largeur de la figure elle-même.[1]

[1] Notez dans cet exemple que, au total, les deux figures n'occupent que 90 % de la largeur du texte. Je garde en effet un peu de marge des deux côtés dans un souci esthétique.

- Entre chaque image, vous pouvez mettre les commandes de votre choix. J'ai mis `\quad` pour espacer un peu les figures, mais vous pouvez par exemple mettre `\\` pour aller à la ligne, ce qui mettra les figures l'une en dessous de l'autre.
- La généralisation à plus de deux figures est immédiate, sachant que vous pouvez panacher `\quad` et `\\` (ou toute autre commande du même genre) entre chaque.

4.2 Valeurs numériques et unités

Si vous avez besoin d'écrire des valeurs numériques avec ou sans unité, utilisez le paquetage `siunitx`. Chargez-le en ajoutant dans votre préambule :

```
\usepackage[output-decimal-marker={,}, exponent-product=.,, group-digits=true]{siunitx}
```

Les arguments optionnels servent à s'adapter au Français. Ce paquetage fournit plusieurs commandes, utilisables à l'identique en mode texte ou en mode mathématique. Avec elles, les puissances de 10 s'écrivent avec la lettre e, comme dans un langage de programmation classique.

- Valeur numérique sans unité : `\num{1,3e-2}` donne $1{,}3.10^{-2}$
- Unité seule : `\unit{m.s^{-1}}` donne $\mathrm{m\,s^{-1}}$
- Valeur numérique avec unité : `\qty{1,52e-6}{\micro m}` donne $1{,}52.10^{-6}\,\mu\mathrm{m}$

Il y a quelques unités préprogrammées bien utiles, par exemple :

- Degrés Celsius : `\qty{25}{\degreeCelsius}` donne $25\,^{\circ}\mathrm{C}$
- Pourcentage : `\qty{42}{\percent}` donne $42\,\%$

Pour les angles en degrés, utilisez `ang{}` : `\ang{34;12;9}` donne $34^{\circ}12'9''$, ou `\ang{25}` donne simplement 25°.

4.3 Insérer un programme

Si vous avez tapé un programme dans un langage usuel (C, Python, etc) et que vous voulez l'insérer dans un document LaTeX, chargez le paquetage `listings`. Il est capable d'insérer aussi bien du code tapé dans le document source LaTeX que du code situé dans un autre fichier, en ajoutant des options cosmétiques comme la coloration syntaxique ou la numérotation des lignes.

Tout d'abord, il faut charger ce paquetage en ajoutant dans votre préambule :

```
\usepackage[procnames]{listings}
\lstset{language=Python,
        basicstyle=\ttfamily,
        keywordstyle=\color{red},
        commentstyle=\color{gray},
        stringstyle=\color{red},
        identifierstyle=\color{blue}
        }
```

Seule la première ligne est indispensable. Les suivantes sont optionnelles, mais permettent d'améliorer le rendu :

- `language` sélectionne le langage, pour permettre l'analyse correcte de la syntaxe.
- `basicstyle` sélectionne une police de caractères de type « machine à écrire » (à chasse fixe).
- Les lignes suivantes choisissent les couleurs pour, respectivement, les commandes, les commentaires, les chaînes de caractères et les variables.

Ensuite, à l'endroit où vous voulez insérer le programme (situé dans le même dossier que votre document LaTeX), tapez la seule ligne :

```
\lstinputlisting[language=Python, frame=single, numbers=left]{recherche_zéro.py}
```

L'option `frame` ajoute un cadre autour du programme, et `numbers` ajoute une numérotation des lignes (ici sur la gauche). Voici le résultat de la commande :[2]

[2] Il est possible de configurer `listings` pour la coloration syntaxique, voir sa documentation.

```
1  import scipy.optimize as so
2
3  def f(x,K,C0):
4      return x*(C0-x)**2-K
5
6  K1 = 1e-4.8
7  xmin, xmax = 0, 0.01
8
9  # brentq renvoie le premier zéro trouvé dans [xmin, xmax]
10 racine = so.brentq(f, xmin, xmax, args=(K1,))
11 print("racine␣:", racine)
```

4.4 Ouvertures

Les possibilités de LaTeX sont vastes, et ce qui précède ne fait qu'en effleurer la surface. À un moment, vous aurez certainement un besoin qui n'est pas couvert de manière satisfaisante par le préambule proposé dans ce document.

Voici quelques pistes d'exploration. Vous trouverez les documentations sur https://ctan.org/.

- LaTeX dispose de son propre langage de programmation spécialisé dans la production de graphes et schémas, avec des modules dédiés pour de nombreux domaines scientifiques,[3] mais aussi de capacités de calcul (pour tracer des courbes) ou de constructions géométriques (intersections, projetés…) Il s'agit de `TikZ`.[4]
- `circuitikz` remplace la bibliothèque de tracé de circuits électriques de `TikZ` par une autre plus puissante et plus simple à utiliser (il n'est pas nécessaire de connaître `TikZ` pour l'utiliser.)
- `tcolorbox` s'appuie sur `TikZ` pour produire des encadrés élégants, comme ceux qui parsèment le présent document.
- `amsthm` fournit des commandes pour mettre en page des énoncés de définition ou de théorème à la manière d'un article de revue.
- `mhchem` est très utile pour la chimie, il permet d'écrire des formules brutes ou développées, ou encore des équations bilans, de manière très légère.
- `fancyhdr` permet de personnaliser les en-têtes et pieds de page du document. Le présent document utilise les en-têtes et pieds de page par défaut, donc ne chargez ce paquetage que s'ils ne vous conviennent pas.
- `lastpage` permet de déterminer le numéro de la dernière page, typiquement pour numéroter les pages sous la forme « page 2 sur 8 » (altérer la numérotation des pages se fait typiquement à l'aide de `fancyhdr`, mentionné ci-dessus).
- Pour faire des diapositives animées, Beamer propose un *documentclass* adapté avec des commandes de temporisation (pour la révélation progressive du texte).
- `url` permet de typographier correctement des adresses internet et de les rendre cliquables (comme c'est le cas un peu partout dans le présent document).
- Des paquetages comme `xsim` ou `answers` servent à concevoir des feuilles d'exercices avec corrigés de manière modulaire : inclure ou non les corrigés, exclure des exercices, etc. Je n'ai pas d'expérience personnelle avec ces paquetages.
- `esint` permet d'écrire de nombreuses sortes d'intégrales. D'abord il fournit `\iint` et `\iiint`, qui remplacent les commandes de base du même nom et produisent des résultats plus élégants. Ensuite, il permet de saisir des intégrales plus « exotiques », comme les intégrales sur des contours ou des surfaces fermées.

Priorité aux paquetages

Devant un nouveau besoin, votre premier réflexe doit être de chercher un paquetage approprié et non de programmer une solution à la main. TeX Live vient avec plus de 5000 paquetages ! Si vous avez un besoin, il est hautement probable que d'autres l'ont eu aussi et qu'un paquetage approprié existe déjà. Programmer à la main présente des risques d'effet de bord non négligeables quand on ne maîtrise pas le fonctionnement interne de LaTeX.

4.5 Insérer du code Lua

Lua est un petit langage de programmation généraliste qui a trouvé son chemin dans la version la plus moderne du compilateur LaTeX, LuaLaTeX. Sa syntaxe n'a rien à voir avec celle de LaTeX, donc faire cohabiter

[3] Mais aussi pour des usages plus fantaisistes comme la reproduction d'enluminures ou la conception de feuilles de personnage !
[4] Lisez « TikZ pour l'impatient » sur http://math.et.info.free.fr/TikZ/bdd/TikZ-Impatient.pdf.

les deux langages dans le même fichier nécessite d'inclure le code Lua dans des environnements LaTeX dédiés.

Voici le code complet d'un exemple (le préambule est tout de même épuré afin de faciliter la lecture) :

```
% !TeX TS-program = lualatex
% !TeX encoding = UTF-8
% !TeX spellcheck = fr_FR

\documentclass[a4paper, french]{article}
\usepackage{polyglossia}
\setdefaultlanguage{french}
\usepackage{fontspec}
\usepackage{luacode}

\directlua{
    a = 1.2
    b = 2
}

\begin{document}
Le résultat est \directlua{
    h = a+b^2
    tex.print(h)
    }
qui doit être inférieur à 10.
\end{document}
```

Le PDF résultant contient une seule ligne : « Le résultat est 5.2 qui doit être inférieur à 10. »

- Il faut charger le paquetage `luacode`.
- Le code Lua doit se trouver en argument de la commande `\directlua` et peut comporter plusieurs lignes. Attention à bien respecter les règles de syntaxe de Lua, qui diffèrent de celles de LaTeX.
- S'il y a plusieurs commandes `\directlua`, elles se comportent comme autant de morceaux d'un code unique, donc des variables définies dans l'une sont accessibles dans les autres. Sur cet exemple, j'ai défini des valeurs numériques dans la première, placée dans le préambule, et je les utilise dans la seconde, placée au fil du texte dans le corps du document.
- Dans le code Lua, la commande `tex.print()` transmet son argument à LaTeX, autrement dit provoque son affichage dans le PDF.
- Pour un rendu plus joli des nombres, pensez au paquetage `siunitx` décrit en section 4.2.

Bien sûr, on peut réaliser des choses bien plus riches car Lua est un langage complet, qui dispose de toutes les structures de base (boucles, branchements, etc) ainsi que de bibliothèques scientifiques performantes. LaTeX seul étant notoirement lent et peu précis pour les calculs (surtout flottants), Lua peut vous ouvrir de nouveaux horizons.[5]

4.6 Générer une présentation avec Beamer

Beamer est une classe de document spéciale pour réaliser des diapositives, en vue d'une présentation. Ce genre de document obéit à des conventions d'apparence très différentes de celles d'un document papier :

- densité de texte beaucoup plus faible (plus d'espaces, caractères plus gros),
- lisibilité globale immédiate, ce qui favorise les polices de caractères dites « sans serif » (sans empattement),
- centrage vertical du contenu des diapositives.

La classe de document Beamer vient avec une configuration adaptée à ces contraintes.

4.6.1 Le préambule

Vous pouvez reprendre le préambule proposé au paragraphe 2.2, avec seulement quelques modifications :

[5] Ainsi, certaines parties du paquetage `TikZ` deviennent notablement plus rapides si vous compilez avec LuaLaTeX, même si vous ne tapez aucun code en Lua (`TikZ` le fait pour vous).

- dans \documentclass, déclarez la classe beamer à la place de article, et supprimez l'option twoside qui n'a aucun sens pour une présentation,
- supprimez les options passées à hyperref et geometry, pour ne pas interférer avec les pré-réglages faits par Beamer,
- supprimez le paquetage enumitem et la commande \setlist{noitemsep}, là aussi parce que Beamer apporte sa propre gestion des listes à puce.

Ce qui donne (les lignes non modifiées ne sont pas répétées ici) :

```
\documentclass[a4paper, french]{beamer}
(...)
\usepackage{hyperref}
(...)
\usepackage{geometry}
(...)
```

Beamer vient avec avec un certain nombre de modèles de présentation appelés *thèmes*. Pour ce document je ne vais en mentionner que deux :

- CambridgeUS est sobre, avec une présentation bien aérée.
- Marburg est plus élaborée, avec un bandeau latéral affichant le plan de la présentation en mettant en surbrillance la partie en cours.

Le choix du thème se fait en ajoutant une commande dans le préambule :

```
\usetheme{CambridgeUS}
```

4.6.2 Structure d'une présentation

Une présentation est subdivisée de manière plus simple qu'un document papier :

- Le document est découpé en parties, avec l'habituelle commande \section.
- Chaque partie est découpée en sous-parties, avec l'habituelle commande \subsection.
- Chaque sous-partie est découpée en diapositives, délimitées par un environnement frame.

Voici un exemple de diapositive :

```
\begin{frame}{Titre de cette diapositive}
Le contenu de cette diapo est saisi avec la syntaxe habituelle.

\medskip

\begin{itemize}
\item premier point de la liste à puces
\item second point
\end{itemize}

\bigskip

\begin{figure}[!htb]
\centering
\includegraphics[width=5cm]{lovebites.jpg}
\caption*{Asami et Miyako.}
\end{figure}

\end{frame}
```

Le résultat est montré figure 4.3. Remarquez que les bandeaux occupant le haut et le bas de la diapositive sont générés automatiquement avec le thème.

- En haut à gauche : le titre fourni par la commande \section précédente.

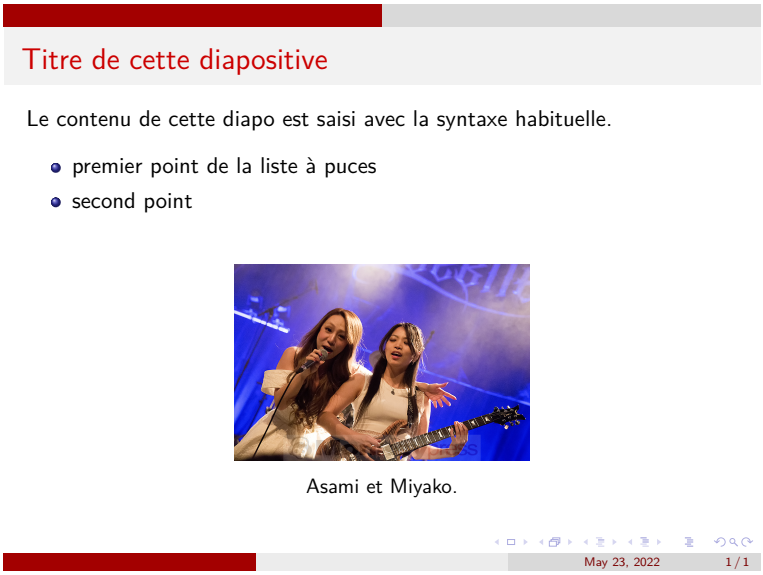


FIG. 4.3 : Exemple de diapositive avec le thème CambridgeUS.

- En haut à droite : le titre fourni par la commande `\subsection` précédente.
- Le titre de la diapo est passé en option de `\begin{frame}`.
- Le contenu est tapé comme d'habitude, mais vous constatez l'espacement et le choix de police adapté à la présentation.[6]
- En bas à gauche : le nom fourni par la commande `\author` dans le préambule.
- En bas au milieu : le titre global fourni par la commande `\title` dans le préambule.
- En bas à droite : la date fournie par la commande `\date` dans le préambule, et le numéro de la diapo.

4.6.3 Apparition progressive du contenu

Beamer permet d'afficher progressivement le contenu d'une diapositive.[7]
Voici une variation de la diapositive précédente avec les ajouts nécessaires.

```
\begin{frame}{Titre de cette diapositive}
Le contenu de cette diapo est saisi avec la syntaxe habituelle.

\medskip

\begin{itemize}
\item<1-> premier point de la liste à puces
\item<2-> second point
\end{itemize}

\bigskip

\begin{uncoverenv}<3>
\begin{figure}[!htb]
\centering
\includegraphics[width=5cm]{lovebites.jpg}
\caption*{Légende de la figure.}
\end{figure}
\end{uncoverenv}

\end{frame}
```

Les deux syntaxes utilisables y apparaissent :

[6] Notez aussi l'usage de la commande `\caption*` pour la figure, ce qui génère une légende sans numérotation, ce qui est courant dans ce type de document.

[7] Techniquement, chaque environnement `frame` produit autant de diapositives qu'il y a d'étapes pour l'affichage progressif, ce qui donne l'illusion d'une unique diapositive dont le contenu apparaît peu à peu.

- Pour un élément d'une liste à puces, il suffit de mettre `<n->` après `\item` afin que cet élément apparaisse à la n^{e} étape.
- Quand on veut que tout un bloc de contenu apparaisse à la n^{e} étape, il suffit de l'insérer dans un environnement `uncoverenv` suivi de `<n->`.

Sur cet exemple, la diapositive va donc apparaître avec seulement son titre et le premier élément de la liste à puces (étape 1), puis le deuxième élément de la liste apparaît (étape 2), puis la figure apparaît (étape 3).

Chapitre 5

Mettre LaTeX à sa sauce

Comme expliqué au début de ce document, le but ici est de rester dans le périmètre de ce que LaTeX permet « sans bidouiller ». Néanmoins, cela n'interdit pas un minimum de personnalisation et de confort d'utilisation.

5.1 Document source en plusieurs fichiers

5.1.1 Plusieurs fichiers sources

Quand un document devient volumineux, tout stocker dans le même fichier peut devenir désagréable. La stratégie est alors d'avoir un *document maître* contenant le préambule, et l'environnement `document` mais ce dernier, au lieu de contenir le corps du document, ne contient que la liste des autres fichiers dans lesquels se trouve vraiment le corps du document. Par exemple :

```
\begin{document}
\include{fichier1}
\include{fichier2}
\end{document}
```

LaTeX va alors chercher les fichiers nommés `fichier1.tex` et `fichier2.tex` et insérer leur contenu pour fabriquer le PDF.

> Les fichiers sources autre que le document maître ne doivent pas contenir d'environnement `document`. Lors de la compilation, ils sont traités les uns après les autres, un saut de page étant inséré entre chaque. Il est classique (mais pas obligatoire) de dédier chaque fichier à un chapitre.

Veillez à ne compiler que le document maître (puisque c'est lui qui contient le préambule). Certains éditeurs détectent cette situation automatiquement, d'autres ont besoin d'une « macro magique » (comme celles décrites section 2.2.2). Au début de chaque fichier sauf le document maître, mettez :

```
% !TeX root = document-maître.tex
```

en adaptant le nom du document maître bien sûr. Ainsi, si vous demandez la compilation du fichier, votre éditeur identifiera correctement le document maître et pourra lancer la compilation sur le bon fichier automatiquement.

5.1.2 Dossiers d'images

Si vous avez beaucoup d'images, ou si ces images sont partagées entre plusieurs documents, il peut être utile de les regrouper dans un dossier unique et de dire à LaTeX d'aller automatiquement les chercher là. Placez dans le préambule, par exemple :

```
\graphicspath{{images/}}
```

pour que le sous-dossier `images/` soit utilisé. Si vous piochez dans plusieurs dossiers :

```
\graphicspath{{images1/}{images2/}}
```

Le `/` à la fin du nom de chaque dossier est obligatoire.

5.2 Créer une commande personnalisée

\newcommand permet, comme son nom l'indique, de définir vos propres commandes. Son premier argument est le nom de la commande, son second est son contenu. Elle doit se trouver dans le préambule.

Il ne doit pas exister de commande portant déjà ce nom, sans quoi la compilation échouera avec une erreur.

5.2.1 Commande sans argument

Par exemple, vous pouvez souhaiter économiser de la saisie pour certains mots :

```
\newcommand{\cad}{c'est-à-dire}
```

Si le contenu de la commande est destiné à être utilisé en mode mathématique, vous pouvez en informer LaTeX avec \ensuremath. Par exemple, pour saisir plus vite le symbole de l'ensemble des entiers naturels :

```
\newcommand{\Nint}{\ensuremath{\mathbb{N}}}
```

LaTeX vérifiera alors si le mode mathématique est actif, et l'activera automatiquement si ce n'est pas le cas.

5.2.2 Commande avec arguments

\newcommand prend alors un argument facultatif qui est le nombre d'arguments. Dans son contenu, faites référence à ces arguments par `#1`, `#2`, etc.

Créons par exemple une commande qui prend deux arguments pour écrire une dérivée partielle :

```
\newcommand{\dpa}[2]{\ensuremath{\frac{\partial #1}{\partial #2}}}
```

Ainsi, \dpa{f}{x} donne $\frac{\partial f}{\partial x}$

5.3 Programmation de macros dans TeXstudio

Automatiser la saisie d'un morceau de code (appelé aussi *code snippet*) est une fonction courante dans la plupart des éditeurs convenablement conçus.

Voyons l'exemple de TeXstudio, qui est l'éditeur que je vous recommande. Reprenez par exemple la série de commandes permettant d'insérer une figure (paragraphe 2.8). Vu sa longueur, il est raisonnable de souhaiter que l'éditeur en saisisse un maximum à votre place.

Allez dans le menu Macros, cliquez sur « Éditer les macros ». Vous arrivez sur la fenêtre représentée figure 5.1.

Il y a trois champs importants :

- Le « nom » apparaîtra dans le menu Macros, une fois la macro finalisée.
- Le plus important est le « déclencheur » : si vous tapez les quatre caractères `sub1` dans l'éditeur, ils seront instantanément remplacés par tout ce qui se trouve dans « Code LaTeX ».

C'est aussi simple que ça !

Entre ceci et la capacité de TeXstudio à auto-compléter toute commande ou référence que vous êtes en train de taper à l'aide de la touche Tabulation, il y a moyen d'accélérer considérablement votre vitesse de saisie.

5.4 Classe personnalisée basique

Limitons-nous à la création d'une classe qui n'a pas d'autre vocation que de regrouper l'essentiel de votre préambule, afin de le rendre commun à tous vos fichiers. Ainsi, si vous apportez une modification à votre préambule, celle-ci sera automatiquement répercutée sur tous vos documents (moyennant une recompilation bien sûr).

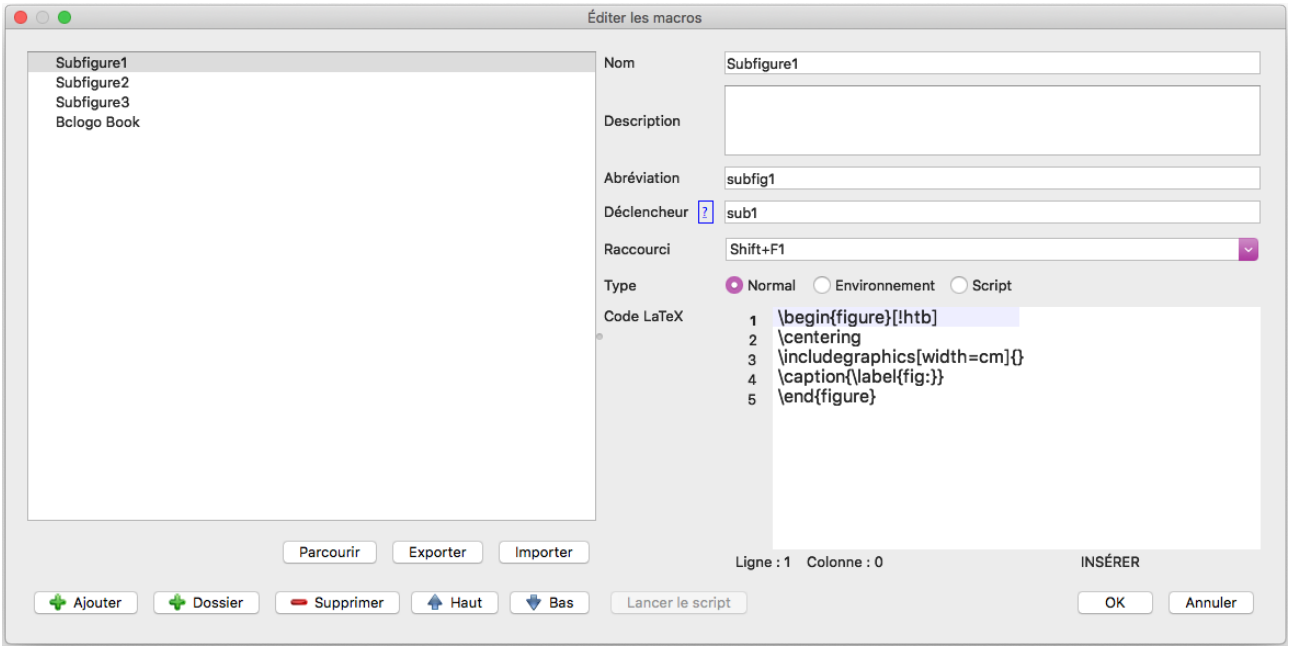


FIG. 5.1 : Boîte de dialogue d'édition de macro dans TeXstudio.

5.4.1 Création du fichier de classe

Disons que votre classe est de type article et qu'elle s'appelle `monarticle`. Dans votre éditeur LaTeX, créez un nouveau document et enregistrez-le sous le nom `monarticle.cls`. Son contenu est :

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{monarticle}[Ma classe personnalisée, v2, du 5 avril 2022]
\LoadClass[a4paper, french, twoside]{article}

\RequirePackage{polyglossia}
\setdefaultlanguage{french}
\RequirePackage{fontspec}
\defaultfontfeatures{Ligatures={Common, TeX}}
\RequirePackage{mathtools,amssymb}
\RequirePackage{color}
\RequirePackage{graphicx}
\RequirePackage[left=2cm,right=2cm,top=2cm,bottom=2cm]{geometry}
\RequirePackage{enumitem}
\setlist{noitemsep}
\RequirePackage{tabularray}
\RequirePackage[pdfusetitle, colorlinks=true]{hyperref}
```

En résumé :

- La ligne `\ProvidesClass` sert à nommer votre classe, accompagnée d'un petit descriptif optionnel.
- La ligne `\LoadClass` précise le style de document sur lequel votre classe est basée.[1] On y retrouve essentiellement ce qui se trouvait à l'origine sur la ligne `\documentclass`.
- Pour déclarer un paquetage, on utilise `\RequirePackage` au lieu de `\usepackage`, avec une syntaxe par ailleurs identique.
- Les autres commandes sont inchangées.

5.4.2 Où placer le fichier de classe ?

Si vous le placez dans le même dossier que votre document LaTeX, il sera correctement utilisé. Ceci est pratique si vous voulez partager votre document avec quelqu'un d'autre : vous joignez le fichier de classe au document pour que le destinataire puisse compiler lui-même le PDF.

[1] Il est bien sûr possible de créer une classe *ex nihilo*, mais c'est une tout autre histoire !

Mais ce n'est pas évidemment pas la bonne solution si vous voulez que votre fichier de classe soit commun à tous vos documents. Pour cela, il doit se trouver un dossier précis.

- Sous macOS, à partir de votre dossier personnel, mettez-le dans `Bibliothèque/texmf/tex/latex/`
- Sous Windows ou Linux, à partir de votre dossier personnel, mettez-le dans `texmf/tex/latex`

5.4.3 Que reste-il dans le document LaTeX ?

Avec la classe ci-dessous, le préambule décrit au paragraphe 2.2 se réduit à :

```
% !TeX TS-program = lualatex
% !TeX encoding = UTF-8
% !TeX spellcheck = fr_FR

\documentclass{monarticle}

\title{Titre du document}
\author{Nom de l'auteur}
\date{Date du document}

\begin{document}
\maketitle

Ceci est le corps du document.

\end{document}
```

Quoi qu'il en soit, les macros magiques doivent rester dans le document LaTeX, puisqu'elles sont destinées à l'éditeur et non au compilateur.

Si vous voulez automatiser même la saisie du peu qui reste, TeXstudio (et d'autres éditeurs) permet la création de « modèles » (ou templates) que vous pourrez charger à la demande.

Chapitre 6

Au passé et au futur...

Il est temps de brosser un portrait un peu plus large de ce qu'est LaTeX, et de l'univers logiciel auquel il appartient.

6.1 Une brève histoire de TeX

6.1.1 L'œuvre de Donald Knuth

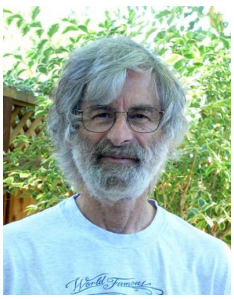
Dans les années 1970, les outils pour typographier un document à l'aide d'ordinateurs étaient très rudimentaires, surtout lorsqu'on voulait y inclure du contenu scientifique, comme des symboles mathématiques. C'est en faisant face à cet état de fait que le mathématicien Donald Knuth a imaginé un langage de programmation dont le compilateur pourrait prendre le contenu et en faire un document parfaitement mis en page.

C'est ainsi qu'est né TeX. Son nom même illustre ses capacités révolutionnaires à l'époque : il permettait par exemple de typographier des lettres grecques au sein d'un document anglophone. Et mieux, il permettait de jouer avec la mise en page de chaque lettre séparément. D'où ce logo, $\tau\varepsilon\chi$ où les lettres présentent des hauteurs et des espacements volontairement fantaisistes.

TeX est arrivé à maturité à la fin des années 1980 et est, depuis, en mode maintenance : plus aucune nouvelle fonctionnalité ne lui est ajoutée, et ses seules évolutions sont des corrections de bugs. Son numéro de version est depuis lors une écriture décimale du nombre π, chaque correction de bug ajoutant un chiffre.[1]

Il ne faut surtout pas prendre cette absence d'évolution pour un défaut. Knuth a pensé TeX comme un système abstrait aux caractéristiques universelles. Par exemple, il peut gérer des longueurs de l'ordre de la longueur d'onde de la lumière visible.

(a)



(b)

Fig. 6.1 : a) Donald Knuth. b) Leslie Lamport.

6.1.2 La contribution de Leslie Lamport

La grande généralité — et la grande abstraction — de TeX le rend par contre très lourd d'utilisation.[2] Cela a poussé Leslie Lamport à combiner les commandes TeX pour former des *macros*, c'est-à-dire des commandes élaborées à la portée plus limitée mais mieux adaptées pour simplifier la vie d'un utilisateur « normal ». Ce système de macros porte son nom : Lamport's TeX… autrement dit, LaTeX.

[1] Au moment où j'écris ces lignes, la dernière version en date est la 3,141 592 65 sortie en 2014.

[2] Si vous pratiquez la programmation, vous pouvez voir TeX comme l'assembleur des systèmes de typographie.

La première version de LaTeX est sortie au début des années 1980, mais les efforts sur son développement se sont rapidement dispersés, donnant naissance à de multiples implémentations incompatibles entre elles. La situation a heureusement été rectifiée en 1994, avec la publication de la version 2ε, qui a essentiellement unifié LaTeX. C'est aussi à ce moment que Lamport a confié les rênes de LaTeX au groupe qui deviendra le projet LaTeX 3.

On peut dire que, encore aujourd'hui, c'est la version du système de macros que tout le monde utilise. On ne peut par contre pas dire que c'est un système figé comme TeX car son environnement logiciel, lui, évolue constamment pour accompagner les progrès du monde technologique : compilateurs, paquetages, etc. Par exemple, sa capacité à produire des documents au format PDF ou à manipuler l'encodage universel UTF8 ne date que de la fin des années 1990, peu après l'apparition de ces standards. Le compilateur PDFLaTeX, qui reste le plus utilisé aujourd'hui, date de cette période, mais son développement est désormais ralenti.

6.1.3 Le projet LaTeX 3

Comme son nom l'indique, le projet LaTeX 3 a pour vocation de donner un successeur à LaTeX 2ε. C'est un travail de longue haleine pour éliminer autant que possible sa *dette technologique*, autrement dit ses limitations et ses lourdeurs dues aux circonstances de son développement remontant à une époque révolue.

En pratique, certaines améliorations offertes par ce projet sont déjà disponibles car elles ont été intégrées à des paquetages utilisables par LaTeX 2ε. C'est dans cette lignée que sont nés les deux compilateurs modernes :

- LuaLaTeX, recommandé dans le présent document et qui est le candidat préconisé pour devenir le compilateur « officiel » de LaTeX. Son point fort est l'intégration d'un moteur Lua, et donc la possibilité de réaliser très efficacement certaines traitements et calculs avec ce langage.
- XeLaTeX, particulièrement adapté à l'écriture de documents polyglottes où coexistent plusieurs systèmes d'écritures (alphabet roman, idéogrammes…)

Ces deux compilateurs ont beaucoup de choses en commun car ils reposent toujours sur la syntaxe de LaTeX 2ε et utilisent tous les deux le paquetage `fontspec` pour gérer la saisie et la manipulation de polices de caractères, ainsi que les paquetages du projet LaTeX 3.

> Notez toutefois que, entre autre du fait des progrès de LuaLaTeX, le compilateur XeLaTeX est entré en mode maintenance. Cela signifie qu'il continuera à être utilisable dans le futur, mais qu'il ne recevra plus de nouvelles fonctionnalités. À moins d'avoir des besoins très spéficiques, il vaut donc mieux se focaliser sur LuaLaTeX.

6.2 Une invitation à la prudence

6.2.1 Inertie et souplesse, et les pièges entre les deux

LaTeX est un outil très souple : si vous avez un besoin, il est hautement probable qu'il existe un paquetage pour le satisfaire. Et s'il ne vous convient pas, vous pouvez toujours revenir aux commandes TeX fondamentales pour obtenir à peu près n'importe quel résultat.

C'est aussi un outil dont le développement a beaucoup d'inertie. Il est encore possible aujourd'hui, avec une installation récente, d'utiliser l'ancien compilateur des années 1990 bien qu'il soit incompatible avec la plupart des standards modernes.

> Ceci signifie que, au gré des forums et des sites consacrés à LaTeX, on trouve des conseils *très* divers, et qu'il faut rester prudent avant de suivre un conseil trouvé ici ou là.
> Des gens continuent aujourd'hui à utiliser des techniques que des paquetages, ou des évolutions naturelles des compilateurs, ont rendu obsolètes depuis des années. Si cela peut se justifier pour qui a un « passif » de centaines, voire de milliers, de documents qu'il serait fastidieux de faire migrer vers des solutions pourtant plus simples et plus modernes, c'est aussi une peau de banane pour un nouvel utilisateur de LaTeX. Il serait dommage que vous vous enfermiez dans des pratiques dépassées.

C'est pour cela que le présent document a un parti pris très clair : vous fournir un préambule clef en main pour vous permettre d'aller le plus vite possible vers l'utilisation de LaTeX. J'ai donc fait des choix là où LaTeX offre de multiples manières de procéder, et les solutions non retenues ne sont pas discutées. Il s'agit donc d'*une* manière d'utiliser ce système pour laquelle je peux vous promettre que :

- elle vise la simplicité pour la grande majorité des usages, au détriment de la souplesse ;

- elle est à jour au niveau des outils préconisés et ne traîne aucune dette technologique.

Mais il vous appartient de déterminer si elle vous convient. Après avoir utilisé LaTeX un certain temps à partir de ce document, ou si vous avez déjà de l'expérience sur le sujet, vous ajusterez vos choix.

6.2.2 Les mauvaises habitudes à ne pas prendre

Que l'on s'entende : quand je parle de « mauvaises habitudes », j'ai en tête la situation d'une personne qui débute en LaTeX et n'a donc pas d'historique à préserver. Un tel utilisateur a tout intérêt à embrasser d'emblée les évolutions modernes de ce logiciel. Dans ce cadre, certaines anciennes manières de procéder sont devenues obsolètes et peuvent être qualifiées de mauvaises habitudes.

Ainsi, si vous cherchez sur internet comment réaliser ci ou ça avec LaTeX, méfiez-vous ! Voici quelques pistes pour reconnaître des conseils qu'il vaut peut-être mieux éviter de suivre.

Les paquetages à éviter

En général, ce sont les paquetages utilisés qui vendent la mèche :

- `pstricks` : cet ancêtre de `TikZ` n'est compatible avec le format PDF qu'au prix de bidouilles pas toujours fiables.
- `fontenc` et `inputenc` : remplacés par `fontspec` dans les compilateurs modernes comme LuaLaTeX.
- `babel` : bien qu'encore utilisable, il est conseillé de le remplacer par `polyglossia` avec les compilateurs modernes.
- À l'inverse, si un document règle les marges sans passer par `geometry`, c'est mauvais signe pour la qualité de sa conception, évitez-le.

Le format de sortie : fuyez le Poscript !

Les compilateurs LaTeX produisant des formats comme le DVI ou le Postscript, mais ces formats appartiennent à un lointain passé. Ils sont incompatibles avec des fonctions élémentaires comme les liens cliquables, et les logiciels capables de les lire sont rares et anciens. Ils sont en outre très inefficaces (les documents obtenus sont inutilement volumineux).

Attention aux primitives TeX

Les primitives TeX sont des commandes « bas niveau » qui sont utilisées par le moteur TeX, et non par LaTeX. Mixer les deux est dangereux car cela peut engendrer des résultats inattendus et difficiles à comprendre pour qui n'est ne connaît pas bien le fonctionnement interne du système.

Les deux plus connues, qu'il faut donc reconnaître et éviter, sont :

- `\def` sert à créer une commande TeX. En LaTeX, c'est `\newcommand` qu'il faut utiliser, comme expliqué au paragraphe 5.2.
- `$$` sert à ouvrir ou fermer le mode mathématique en TeX, mais son utilisation peut empêcher certaines commandes LaTeX de détecter le mode mathématique. Utilisez plutôt l'environnement `align` décrit section 3.2.2.

6.2.3 Comment migrer ?

Si vous avez des habitudes passées en LaTeX et que vous voulez passer sur un compilateur moderne comme LuaLaTeX, l'essentiel du travail de migration se passe dans le préambule, et il est probable que votre corps de texte pourra être réutilisé (presque) en l'état.

1. Vérifiez que votre document est enregistré en Unicode, ou convertissez-le.
2. Comme expliqué section 6.2.2, remplacez `fontenc` et `inputenc` par `fontspec`. Remplacez aussi `babel` par `polyglossia`.
3. Configurez votre éditeur pour qu'il appelle le bon compilateur (par exemple avec une macro magique).
4. Si vous insérez des figures au format Postscript (`.eps`), LuaLaTeX est capable de les convertir à la volée à condition de charger le paquetage `epstopdf-pkg`. Mais ce format étant extrêmement ancien, il vaut probablement mieux les convertir une fois pour tout en PDF (l'outil `epstopdf` est inclus dans votre distribution LaTeX).
5. Remarque annexe de la précédente, si vous utilisiez `pstricks` pour faire des figures et que vous n'avez pas le temps de les refaire avec un autre outil comme `TikZ`, chargez le paquetage `luapstricks`. Cela assurera automatiquement le portage par LuaLaTeXavec un taux de réussite très correct.

Une fois la migration terminée, vous êtes tranquille pour très longtemps. LaTeX évolue lentement, donc il ne sera pas nécessaire de « mettre à jour » vos habitudes fréquemment (personnellement, je rafraîchis mon préambule pour suivre les évolutions récentes environ deux fois par décennie).

Rappelez-vous que vous n'êtes pas obligé de migrer tous vos documents ! Grâce aux macros magiques, vous pouvez continuer à utiliser votre ancienne configuration éprouvée pour les anciens documents, et une nouvelle configuration dépoussiérée pour les nouveaux documents, et tout cela dans le même éditeur.

Chapitre 7

LaTeX en ligne : Overleaf

Il existe plusieurs solutions en ligne pour saisir et compiler des documents LaTeX sans avoir à installer le logiciel chez soi. C'est évidemment un peu moins performant et on perd la liberté du choix de l'éditeur, mais ça reste pratique. Cela ouvre également des possibilités intéressantes comme l'édition collaborative.

Au moment où j'écris ces lignes, la solution qui me semble la plus efficace est Overleaf.[1] Ce chapitre montre comment mettre en œuvre les conseils du présent document avec cet outil en ligne.

La figure 7.1 montre l'interface d'Overleaf (susceptible d'évoluer, bien entendu !)

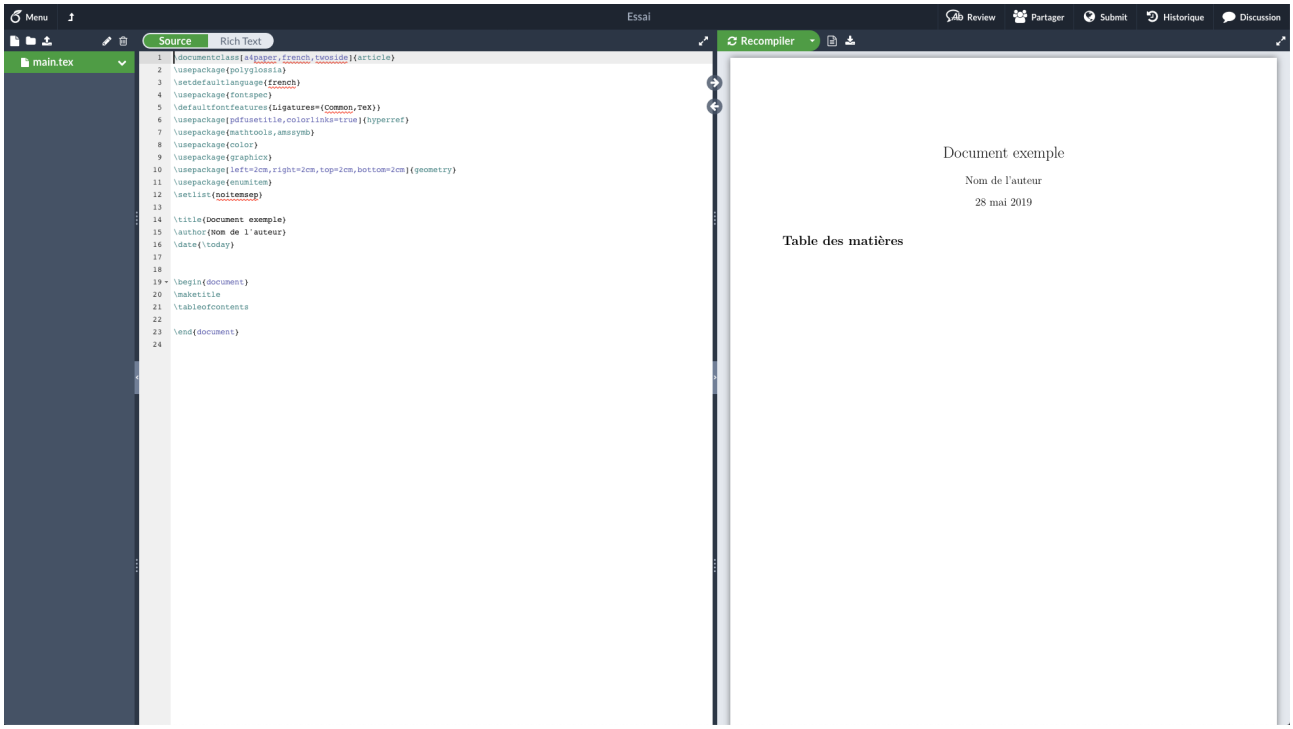


FIG. 7.1 : Page principale d'Overleaf.

1. Overleaf ne reconnaît pas les macros magiques, donc il faut configurer la langue et le compilateur dans le menu (en haut à gauche) :
   - Compilateur : LuaLaTeX
   - Correcteur orthographique : French
2. Il ne vous reste plus ensuite qu'à copier-coller le préambule que je vous ai proposé dans ce document (à la place de celui par défaut).
3. La génération du document PDF se fait avec le bouton « Recompiler ».
4. À côté de ce bouton se trouve aussi le bouton pour télécharger le PDF.

[1] https://www.overleaf.com/