DÉBUTER LaTeX

OU TOUT CE QUE J'AURAIS AIMÉ SAVOIR EN DÉBUTANT AVEC LaTeX

TABLE DES MATIÈRES

1 Petit syllabus

1.1 C'est quoi ?

Ce support de formation accompagne 3h de formation pratique. Ici, vous trouverez les informations de base pour débuter avec LaTeX, et vous pourrez vous servir de ce support comme mémo. En plus de ce support, vous avez accès à des fichiers exemples pour vous faire la main, et pour commencer à utiliser LaTeX tous les jours ! Cette formation a une vocation **pratique**, donc je laisse délibérément de coté les détails 'théoriques' (cycle de production détaillé, historique, etc.). Si vous voulez les creuser, libre à vous.

Ici, je privilégie tout ce qui est libre d'accès. Je n'aborderai donc pas l'utilisation de logiciels non-libres.

Après avoir revu les basiques, j'inclue dans ce document des portions de code LaTeX, qui seront typographiées comme cela :

```
Ca c'est du code
```

N'hésitez pas à ouvrir le fichier d'exemple correspondant, qui sera indiqué, à copier-coller les bouts de code dans le fichier `.tex`, et à voir comment ca marche.

1.2 J'apprends quoi ?

Avec cette formation, vous apprendrez :
- à installer LaTeX dans différents environnements, ainsi que certains outils bien pratiques, comme par exemple *BetterBibTeX for Zotero*
- la structure de base d'un fichier LaTeX
- les commandes de base de LaTeX, celles que vous utiliserez tout le temps, et celles dont on a besoin très souvent.
- le fonctionnement de la bibliographie
- à écrire un fichier simple, comme un rapport
- les prémisses de la personnalisation sur LaTeX

Je n'aborde pas la réalisation de présentations avec Beamer. Je n'aborde pas la réalisation de figures avec Tikz et pgf – mais c'est possible !

Bien entendu, ce que je raconte ici n'est que le reflet de mon expérience.

1.3 Quelques ressources utiles

Pour du LaTeX :
- Guide par Frédéric Geraerds, un peu daté
- *LaTeX guide for beginners*, par Pedro Tiago Martins

Pour améliorer vos rédactions et présentations :
- *Shortcomings in scientific writing*, présentation par Jean-luc Dummond. Plein de très bonnes pratiques pour vos rédactions.
- *Effective page layout for the nonartist*, présentation par Jean-luc Dummond. Des astuces pour vos présentations.
- *English communication for scientists*, par *Nature Education*. Aborde la rédaction d'articles et les présentations orales, plein de trucs utiles !
- Petites leçons de typographie, par Jacques André, pour les rapports ou thèses en français (oui, les règles typographiques anglaises, américaines et françaises sont différentes !).
- *Simple rules for concise scientific writing*, par Scott Hotaling.

2 Quand utiliser LaTeX ?

LaTeX est parfait si vous voulez :
- vous concentrer sur le contenu et non sur la forme lorsque vous rédigez
- avoir une mise en forme automatique
- typographie soignée, et professionnelle si besoin
- rédiger des documents propres, avec un format et une mise en page pérennes
- rédiger des documents de quelques pages
- rédiger des documents très longs (type thèse), qui contiennent de nombreuses figures
- rédiger des documents avec une bibliographie propre (appels dans le texte, liens cliquables, doi accessibles, etc.), un index, une table des matières, et le tout en quelques lignes

– insérer des fichiers PDF tels quels, ou des figures en format PDF
– rédiger des formules mathématiques
– travailler avec d'autres personnes sans se soucier du format à partager
– gérer les commentaires, les notes *to do*, les *to-do lists*.

J'utilise donc LaTeX pour rédiger :
– thèse, rapports, articles, cours, projets
– CV
– lettres professionnelles et personnelles qui ont besoin d'avoir une apparence pro
– tous documents volumineux à imprimer (livres, thèses)

Je n'utilise par contre pas LaTeX quand :
– je travaille sur un projet court (pas plus de quelques pages) avec des gens qui ne savent pas l'utiliser **et** qui ne veulent pas annoter des PDF (voir section 7.4)
– je fais des présentations, car je préfère dans ce cas l'approche WYSIWYG[1]
– peu m'importe la mise en page du texte (dans ce cas j'utilise Markdown, on y reviendra)

[1] What You See Is What You Get, comme *Word* ou *OpenOffice*

Quelques autres exemples superbes, faits avec LaTeX
– Un livre par Jean-Luc Doumont
– Une règle de jeux de rôles
– Une reproduction de la bible de Genève de 1564 (!)
– Une reproduction d'une publicité de 1875

3 Avant de commencer

3.1 Installer LaTeX

Sous Linux, installer le package TeXlive fournit par votre distribution.

Sous Windows, installer MiKTex ou TeXLive. MiKTex me parait plus efficient à ce jour.

Sous Mac OS, installer MacTeX.

3.2 Éditeurs LaTeX

Tex, c'est du code. On peut écrire du LaTeX avec un éditeur de texte basique (bloc note sous Windows), et le compiler avec une ligne de commande. Mais c'est quand même plus sympa de travailler avec un environnement graphique, surtout si des aides à la rédaction sont incorporées ! De nombreux éditeurs sont disponibles.

Hors ligne Celui que je préfère est TeXstudio, dispo sous toutes les distributions. TexMaker est bien aussi. Lyx permet d'avoir un environnement WYSIWYG.

Ici, je ne parlerai que de l'utilisation de TeXstudio.

En ligne Overleaf est une référence. Dessus, vous pouvez stocker, partager, modifier, annoter et commenter des fichiers `tex`. Quelques universités proposent des comptes pro, bien pratiques car ils donnent accès à l'option *track-change*. Il est aussi possible de synchroniser automatiquement un projet sur overleaf et un répertoire local, en passant par Git. Sinon, enregistrer le projet manuellement.

3.3 Comment ca marche ?

LaTeX dissocie le fond et la forme. Généralement, la forme est définie avant même de commencer le corps de texte, en préambule. Toutes les options sont définies au préalable : par exemple, taille, police et couleur des titres. Dans le corps de texte, quelques commandes de forme locales sont utilisés : par exemple, pour définir un titre, ou pour mettre un mot en gras.

Pour travailler sur un document LaTeX, vous éditez un fichier `.tex`. Pour obtenir un PDF, ce fichier est ensuite compilé, et toutes les options que vous avez définies sont prises

en compte. Il existe différents compilateurs, tous disponibles dans les éditeurs comme TeXstudio, pour différents types de rendus : pdflatex, lualatex[2] , xetex. La compilation n'est pas automatique ! De nombreux éditeurs proposent une visualisation en parallèle. Ne pas oublier de compiler et visualiser pour mettre à jour votre PDF.

[2] LuaLaTeX permet de lire de l'UTF-8 directement, au contraire de PDFLaTeX. On peut donc inclure des symboles (comme €) sans avoir à trouver la commande LaTeX.

LaTeX est basé sur l'utilisation d'extensions, ou *packages*. Ces packages permettent d'ajouter de nombreuses fonctionnalités très utiles. Ils sont appelés en début de fichier.

LaTeX produit des fichiers lorsqu'il compile votre .tex, notamment :
- .pdf, bien entendu
- .log : journal d'évènement, raconte ce qui se passe pendant la compilation
- .aux : stocke les informations concernant la forme, les références, les figures.
- .toc : stocke les informations de la table des matières
- .bbl : stocke la bibliographie (voir section 6)

> Il est important de noter que les informations stockées dans les fichiers annexes nécessitent souvent 2 compilations pour 'fonctionner'. Par exemple, les références biblio, ou les titres, seront ajoutés (ou *exportés*) aux fichiers .bbl et .toc à la première compilation. Ils ne seront donc pris en compte (ou *importés*), et n'apparaitront dans votre PDF, qu'à la deuxième compilation !

Pour compiler avec TeXstudio, appuyer sur la flèche verte, sur F5 ou aller dans Outils>Compilations

3.4 Commandes

LaTeX fonctionne avec des commandes. Les options de mise en forme, indiquées en préambule, servent à définir les propriétés de ces commandes. Par exemple, avant de commencer la rédaction du document, on indiquera que nos titres doivent être écrits en Arial avec une taille de 12 pt. Dans le texte, on indiquera seulement que telle suite de mot est un titre. Les propriétés de la commande 'titre' seront appliquées automatiquement.

Les commandes ont toujours la même structure, de type `\command`. Elles peuvent avoir des arguments ; dans ce cas, la commande s'applique à l'argument : `\command{argument}`. Les commandes peuvent aussi avoir des arguments optionnels, spécifiés entre crochets : `\command[option]{argument}`.

Les environnements sont des types de commandes particulier. Ils ont toujours un début et une fin, et les propriétés s'appliquent seulement à l'intérieur :

```
\begin{environment}
        blabla
\end{environment}
```

4 Les bases

4.1 Structure d'un fichier tex

Préambule

Le préambule contient toutes les options de mise en forme, les packages additionnels, etc. Dedans, aucun texte n'est écrit.

La première commande d'un préambule (et donc d'un document .tex) sert à déterminer le type du document. Les `document class` peuvent être, entre autres, `article`, `report`, `book`, `memoir`. J'utilise tout le temps la classe `article`, sauf pour les thèses et autres grands documents, pour lesquelles la classe `memoir` est plus adaptée.

Pour la classe article, toutes les options sont décrites dans ce très bon article

Ici, classe article, introduite par :
`\documentclass[a4paper,10pt,oldfontcommands,oneside]{article}`. Entre crochets se trouvent les options de la classe : taille de la page, taille de la fonte[3] par défaut, et `oneside` veut dire que j'ai un seul type de pages (`twoside` implique qu'il y a des pages de gauche ou de droite pour l'impression, ce qui peut modifier les marges, la première page étant toujours à droite). Il y a plein d'autres options, en fonctions des classes.

[3] Une fonte c'est un ensemble de caractères de même police, taille et graisse

Ensuite, les packages sont appelés.

Les différentes options de mises en forme sont définies.

Enfin, dans la classe article, il est possible de définir en préambule le titre de l'article, les auteurs, ou encore la date, et ces informations apparaitront dans le corps de texte si vous appelez `\maketitle`. Comme c'est du texte qui apparaitra en corps de texte, mais qui est défini en préambule, c'est mieux de l'indiquer à la fin du préambule.

Par exemple :

C'est le moment de commencer à faire des tests dans exemple1.tex !

```
\documentclass{article}
\usepackage[round]{natbib}  % bibliography
\title{J'écris un texte}
\author{Jean Dupond}
\date{2017}
```

Corps du texte

Le corps de texte est introduit au sein d'un environnement `document` :

```
\documentclass{article}
\usepackage[round]{natbib}  % bibliography
\title{J'écris un texte}
\author{Jean Dupond}
\date{2017}

\begin{document}
\maketitle

\section{Introduction}
Je m'appelle Jean.

\end{document}
```

Ne surtout pas oublier le `\end{document}` !

4.2 Commandes de base : mise en page, titres

Structure, environnement, style et taille des caractères Je vous invite à regarder dans `>LaTeX>...` pour avoir une idée des options. Par exemple, on utilisera
`\section{Titre}` pour créer une section, `\textbf{mot en gras}` pour mettre du texte en **gras**,
`{\footnotesize texte en petit}` pour réduire localement la taille du texte.

Pour commenter, ajouter un % en début de ligne.

Les sections sont automatiquement ajoutées à la table des matières. Pour qu'une section ne soit pas numérotée ni répertoriée, utiliser
`\subsection*{Sous-section pas numérotée}`

Table des matières La table des matières se fait automatiquement avec la commande `\tableofcontents`. La table des matières se positionne là ou vous appelez la commande : on essaye généralement de la placer après le résumé, et avant l'introduction. Un package bien pratique est `bookmarks`, qui, en combinaison avec `hyperref` (voir section 5.1), permet de créer les *bookmarks* de vos fichiers PDF.

Saut de ligne, de pages, paragraphes et espaces En LaTeX, et c'est la différence principale avec les éditeurs de texte classiques, un retour à la ligne n'équivaut pas à un retour à la ligne du texte.
Pour aller à la ligne sans indentation, utiliser `\\`.

Eh, vous avez remarqué comme les marges sont grandes dans l'exemple 1 ? Ce sont les propriétés par défaut de la classe article. Le package idéal pour modifier ces propriétés est `geometry-détails ici !`

Pour faire un nouveau paragraphe (indentation et à la ligne), sauter une ligne. En fonction des options utilisées en préambule, l'espacement de la ligne avant un nouveau paragraphe peut être plus ou moins grande.

Par exemple, le code correspondant aux lignes précédentes est :

```
En \LaTeX{}, et c'est la différence principale
avec
les
éditeurs de
texte classiques, un retour à la ligne n'équivaut
pas à un retour à la ligne du texte. \\
Pour aller à la ligne sans indentation,
 utiliser \textbackslash\textbackslash.

Pour faire un nouveau paragraphe (indentation et à la ligne),
 sauter une ligne.
```

Lorsqu'un espace vertical est nécessaire, vous pouvez utiliser les commandes `\bigskip`, `\medskip` et `\smallskip`. Ne pas oublier de sauter une ligne avant ces commandes, sinon elles n'auront pas d'effet.

Une autre manière de créer des espaces verticaux ou horizontaux de taille définie est d'utiliser `\vspace*{1cm}` ou `\hspace*{3.8cm}`.

Enfin, il est aussi possible d'utiliser les commandes d'espace élastiques `\vfill{}l et \hfill{}` pour 'remplir' (*fill*) tout l'espace sur la hauteur ou largeur de la page. Par exemple, `\hfill{}caillou\hfill{}` retourne :

caillou

et `\hfill{}caillou` :

caillou

La commande `\newpage` permet de sauter une page, `\clearpage` fait la même chose en forçant tous les flottants à apparaitre (voir section 4.3 pour savoir ce qu'est un flottant).

> Attention ! Certains commentaires spéciaux sont des commandes et sont interprétés comme du code ! C'est notamment le cas de &, %, $, #, _, {, }, ~, ^, et \. Pour les utiliser dans le texte, taper `\&`, `\%`, `\$`, `\#`, `\_`, `\{`, `\}`, `\textasciitilde`, `\textasciicircum`, et `\textbackslash`.

Ce n'est pas le cas des lettres accentuées si le bon package est utilisé (voir section 5.4)

> Un outil surpuissant pour trouver la commande d'un symbole : detexify. Dessine le symbole, il te trouve la commande !

les symboles de base sont lus par Luatex, pas par pdftex

4.3 Figures

Flottant Les figures sont ce qu'on appelle en LaTeX des flottants (*floats*), au même titre que les tableaux par exemple. L'endroit oú ces flottants sont insérés est choisit par LaTeX à partir d'un certain nombres de paramètres, que l'on peut définir ou non. Par exemple, ces paramètres incluent la taille du flottant par rapport à l'espace disponible sur la page, les espacements avant et après le flottant, le nombre total de flottants appelés sur une page. Par exemple, si un flottant est appelé en bas de page mais que la place restante n'est pas suffisante, il sera automatiquement inséré sur la page suivante. On peut aussi choisir d'insérer tous les flottants en haut des pages.

Figure Le package le plus abouti pour inclure des figures est `graphicx`. L'insertion basique d'une figure se fait au sein d'un environnement en utilisant la commande suivante :

```
\begin{figure}
        \centering
        \includegraphics[width=0.75\textwidth]{image}
        \caption{Un chaton mignon.}
        \label{fig :chaton}
\end{figure}
```

La commande `\includegraphics` a plusieurs options permettant de caractériser la largeur d'une figure (ici, la figure fera 75% de la largeur du texte), la hauteur (height), ou encore l'échelle (scale). La commande `\caption` permet de créer une légende. Enfin, la commande `\label` permet d'apposer une étiquette à la figure, pour pouvoir y référer dans le texte (voir section 5.1). Cette commande permet d'incorporer une seule figure centrée sur la page.

Enfait, l'environnement figure permet de créer un flottant avec une légende et une étiquette. Il est possible d'inclure une image (`\includegraphics`) hors environnement flottant, mais ce n'est alors pas une figure.

> **Quel format** pour ma figure ? Le mieux est d'utiliser un format vectoriel, pdf ou eps, qui permet de ne pas avoir de perte de qualité. Sinon, préférer png à jpg, car jpg gère mal la compression et induit souvent des pertes de résolution.

Placement LaTeX tente de placer le contenu flottant en fonction des paramètres qu'on indique entre crochets après le `\begin` du flottant :

J'utilise toujours [ht !]

- h : là où il apparaît dans le source
- t : en haut de la page
- b : en bas de la page
- p : seul sur une page
- ! : force le placement et ignore les règles classiques

L'environnement figure devient donc :

```
\begin{figure}[ht !]
        \centering
        \includegraphics[width=0.75\textwidth]{image}
        \caption{Un chaton mignon.}
        \label{fig :chaton}
\end{figure}
```

Si la largeur de votre figure dépasse la largeur du texte, votre figure ne sera plus centrée. Pour certains cas rares, il peut être nécessaire de dépasser dans les marges et d'avoir une largeur de figure plus importante que la largeur du texte. Pour garder une figure centrée, utiliser la commande `\includegraphics` à l'intérieur de la commande `\centerline{}`.

> Pour ne pas avoir à indiquer le chemin du fichier à chaque appel de figure, la commande `\graphicspath{{path1}{path2}...{pathX}}` permet de définir tous les chemins en préambule. Notez aussi qu'il n'est pas nécessaire de spécifier l'extension du fichier de figure. Sous Windows, une telle commande serait par exemple :
> `\graphicspath{{D:/Latex/rapport1/figures/}{C:/Users/baptiste/thèse/article1/figs/}}`

Wrapfig Le package `wrapfig` propose l'environnement wrapfigure, qui permet d'insérer du texte à côté d'une figure. La figure doit être insérée avant le texte à mettre à côté, et, en fonction de la taille définie, LaTeX insèrera le plus de texte possible.

```
\begin{wrapfigure}{r}{6cm}
\includegraphics[width=5cm]{image}
\caption{Ceci est un chaton imbriqué dans du texte}
\label{fig :wrap}
\end{wrapfigure}
```

La première option correspond à la position de la figure (l pour *left* ou r pour *right*), et la seconde correspond à la largeur totale la figure à insérer (image + marge contre le texte).

la largeur totale indiquée dans l'env. wrapfig doit donc être plus grande que la largeur de l'image

Sous figures Le package `subfig` permet d'insérer des sous-figures au seins d'un même environnement figure, avec des sous-légendes et une légende principale.

```
\begin{figure}[ht !]
        \begin{center}
                \subfloat[Faut pas bosser le WE]{%
                        \includegraphics[width=5.5cm]{fig1a}
                        \label{fig1a}}
                \hspace{0.5cm}
                \subfloat[Vous demain]{%
                        \includegraphics[width=5.5cm]{fig1b}
                        \label{fig1b}}
                \caption{Des mêmes rigolos}
                \label{fig :1}
        \end{center}
\end{figure}
```

4.4 Tableaux

Pour insérer un tableau, utiliser l'environnement `tabular` du package `array`. Un tableau basique

truc	bidule	machin
one	two	three
un	deux	trois

aura cette structure :

```
\begin{tabular}{c|cc} % les 3 c correspondent au nombre de lignes
                      % le | permet de tracer une ligne verticale
\textbf{truc} & \textbf{bidule} &\textbf{ machin}\\ % les & séparent les
↪  colonnes
\hline
one & two & three\\ % les \\ font un retour à la ligne
un & deux & trois\\
\end{tabular}
```

Remplacer le `c` par un `l` ou un `r` permettra d'aligner le texte de chaque cellule à gauche ou à droite respectivement.

Bien-sur, il est conseillé d'inclure un tableau dans un environnement `table`, et d'y associer une légende et une étiquette :

```
\begin{table}[ht !]
\centering
        \begin{tabular}{c|cc}
                \textbf{truc} & \textbf{bidule} &\textbf{ machin}\\
                ↪  colonnes
                \hline
                one & two & three\\
                un & deux & trois\\
        \end{tabular}
\caption{Blabla.}
\label{tab1}
\end{table}
```

Pour avoir une dimension de colonne fixe, remplacer le `c` par `m{5cm}` :

truc	bidule	machin
one	two	three
un	deux	trois

Ici, le tableau n'est pas dans un environnement table – pas de légende ni d'étiquette

Si vous souhaitez contrôler la largeur totale du tableau (comme pour une figure), le rapport entre les largeurs des colonnes,et les largeurs des colonnes, utilisez plutôt le package (et l'environnement associé) `tabularx`, qui ajoute de nombreuses fonctionnalités et est plus flexible.

Il est aussi possible de réaliser des tableaux bien plus compliqués. Parcontre, plus le tableau est complexe, plus c'est embêtant à faire avec latex. Je vous conseille d'avoir recours à des outils qui permettent de créer des tableaux en WYSIWYG ou de les convertir à partir d'un tableau. Entres autres :

- tables generator pour créer des tableaux
- Excel2LaTeX pour convertir
- csv2latex

Enfin le package `booktab` permet d'améliorer largement l'apparence des tableaux.

Si vos tableaux sont vraiment compliqués, autant les faire avec un logiciel WYSIWYG et les importer en PDF dans un environnement tableau

4.5 Formules mathématiques

En LaTeX, on peut écrire des maths dans le texte, en l'insérant entre $, par exemple $\tan^{-1}\left(\frac{d_{i+1} - 2mh_A}{x_j}\right)$ s'écrit `$\tan^{-1} \left(\frac{d_{i+1} - 2mh_A}{x_j}\right)$`.

Vous pouvez aussi écrire les maths dans un environnement. L'environnement `equation` permet de numéroter et d'étiqueter l'équation.

$$\mathbf{K}_{\mathbf{1},\mu_\mathbf{A}}(x_j, f_i) = \tan^{-1}\left(\frac{d_{i+1} - 2mh_A}{x_j}\right) \tag{1}$$

s'écrit :

```
\begin{equation}
\mathbf{K_{1, \mu_A}}(x_j, f_i) = \tan^{-1} \left(\frac{d_{i+1} -
↪  2mh_A}{x_j}\right)
\end{equation}
```

Rajouter une étoile à `\begin{equation*}\end{equation*}` permet d'annuler la numérotation.

Par défaut, les maths sont écrits en italique. Or, par convention, les vecteurs, matrices et tenseurs doivent être droits et en gras, et les fonctions usuelles en style droit. Pour changer l'italique par défaut, utiliser `$\mathrm{K_{1, \mu_A}}$` pour du droit ou `$\mathbf{K_{1, \mu_A}}$` pour du droit et gras.

Quelques basiques dont vous aurez surement l'utilité :

- `$\mu$`, `$\delta$`, `$\rho$` μ, δ, ρ en mode math, il y a plein de symboles, dont les lettres grecques.

- `$\mu_A$` pour faire un indice μ_A, `$d_{i+1}$` entre crochets si indice de plus d'un caractère d_{i+1}
- `$\mu^A$` pour faire un exposant μ^A
- `$\frac{a}{b}$` pour écrire une fraction $\frac{a}{b}$
- `$\sqrt[n]{x}$`, `$\sin$`, `$\ln$` pour les fonctions standards $\sqrt[n]{x}$, sin, ln, etc.
- `$\left($`, `$\right]$` permettent d'écrire des parenthèses, crochets ou accolades de grande taille.

Pour le reste, je vous laisser chercher par vous même !

Enfin, pour les plus longues formules (ou pour insérer des formules dans un environnement à deux colonnes), il peut être intéressant d'utiliser l'environnement `split` (fourni avec le package `amsmath`). Par exemple,

$$\begin{aligned}\mathbf{K_{1,\mu_A}}(x_j, f_i) = \frac{2}{\pi} \sum_{m=1}^{+\infty} \frac{m \cdot \mu_C \cdot \mu_A}{(\mu_C - \mu_A)^2} K_A^m \Bigg[& \tan^{-1}\left(\frac{d_{i+1} + 2mh_A}{x_j}\right) \\ & + \tan^{-1}\left(\frac{d_{i+1} - 2mh_A}{x_j}\right) \\ & - \tan^{-1}\left(\frac{d_i + 2mh_A}{x_j}\right) \\ & - \tan^{-1}\left(\frac{d_i - 2mh_A}{x_j}\right)\Bigg].\end{aligned} \tag{2}$$

s'écrit

```
\begin{equation}
\begin{split}
\mathbf{K_{1, \mu_A}}(x_j, f_i) = {}&
\frac{2}{\pi} \sum_{m=1}^{+\infty} \frac{m \cdot \mu_C \cdot
↪  \mu_A}{(\mu_C - \mu_A)^2} K_A^{m} \biggr[ \tan^{-1}
↪  \left(\frac{d_{i+1} + 2mh_A}{x_j}\right) \\
& \hspace{2cm} + \tan^{-1} \left(\frac{d_{i+1} - 2mh_A}{x_j}\right) \\
& \hspace{2cm} - \tan^{-1} \left(\frac{d_{i} + 2mh_A}{x_j}\right) \\
& \hspace{2cm} - \tan^{-1} \left(\frac{d_{i} - 2mh_A}{x_j}\right)
\biggr].
\end{split}
\end{equation}
```

> Un outil surpuissant pour digitaliser une formule écrite à la main ou copiée, et l'importer sous LaTeX : Mathpix Snip. Écris ta formule à la main (ou copie-colle la), et hop c'est dans LaTeX !

> Pour écrire du code, le package minted est la référence. Il permet d'incorporer et de colorer du code source, dans le texte, dans un environnement, ou à partir d'un fichier. Plus de 500 langages sont supportés, et plusieurs styles existent. Dans ce document, j'utilise minted pour incorporer du code source LaTeX, avec le style tango. **Attention !** Pour utiliser minted, il faut rajouter un '-shell-escape' à vos options de compilation.

dans Options > Configurer TeXstudio > Compilations

5 Rédaction et début de personnalisation

5.1 Liens, hyperliens, appels dans le texte, références

Le système de référence de LaTeX permet de manipuler la numérotation de n'importe quel objet de manière symbolique – c'est à dire que la numérotation se fait automatiquement. On ne gère donc que l'apposition d'étiquettes aux objets à numéroter. Ensuite, on peut référer au numéro de l'objet étiqueté, ou à la page sur laquelle cet objet se trouve.

Pour utiliser une référence, il suffit donc seulement d'apposer une étiquette à un objet avec la commande `\label{bla}` (l'étiquette est `bla`). Ensuite, on fait référence au numéro de l'objet avec `\ref{bla}`. On peut faire référence à la page de l'objet avec `\pageref{bla}`.

On peut apposer des étiquettes à de nombreux objets : titres, flottants, équations, items de listes, etc. Pour un titre, l'étiquette doit être apposée après la commande titre. Pour un flottant, l'étiquette doit être apposée après la commande `\caption`.

Par exemple :

```
\section{Seismic processin}
\label{s :seismic_proc}

\begin{figure}
...
\caption{Seismic profile}
\label{fig :seismic_prof}
\end{figure}
```

> Pour les étiquettes, surtout pour les gros documents (style thèse), essayez d'être cohérent·es : toutes les étiquettes de sections commencent par `s :`, celles des figures par `fig :` par exemple.

Le package `hyperref` permet de transformer en hyperlien toutes les références, y compris celles de la table des matières (qui se font automatiquement). Le package hyperref permet aussi d'inclure des liens url, ou des liens à des fichiers locaux, avec la commande : `\href{adresse}{texte du lien}`. On peut aussi utiliser juste `\url{adresse url}` pour avoir l'adresse dans le texte directement.

En plus, ce package permet de paramétriser toutes les propriétés des liens. Par exemple, on peut choisir la couleur de chaque type de lien (url, figure, hyperlien, etc.). Pour aller plus loin, tout est très bien expliqué ici[4] .

[4] Ca par exemple c'est un hyperlien url

5.2 Entêtes et pieds de page

On peut spécifier le style d'en-tête ou pied de page à l'aide de la commande `\pagestyle{style}`, le `style` étant :

- `empty` ni entête, ni pied de page
- `plain` c'est le style par défaut appliqué aux pages de titre notamment, le pied de page contient les numéros de pages. Personnalisable.
- `fancy` suivant le style de document, un certain nombre d'informations est inséré automatiquement dans l'entête et le pied de page (par exemple dans le style report en recto-verso, est inséré en entête : soit le titre du chapitre en cours, soit le titre de la section en cours). Mais c'est généralement ce style qu'on personnalise.
- `mystyle` ou n'importe quel autre nom, à personnaliser

Là le style c'est fancy !

La commande `\thispagestyle` permet de changer ou de spécifier le style de la page courante.

Les en-têtes et pieds de pages sont gérés par le package `fancyhdr`. Pour chaque style de page, on peut modifier le style de l'entête ou pied de page correspondant (s'il y en a un). Par exemple, le style `fancy` de ce document est :

Pour les couleurs, voir section 5.6

```
\fancyhead[R]{ \fontsize{9pt}{1em}      % en-tête droit
                \color{gray} Théa Ragon }
\fancyhead[L]{ \fontsize{9pt}{1em}      % en-tête gauche
                        \color{blue} Formation \LaTeX{}
                        \hspace{0.1cm} $\cdot$ \hspace{0.1cm} Géoazur
                        ↪  2021 }
\renewcommand{\headrule}{\vskip-9pt \color{blue}\hrulefill }
```

Et pour le `footer` :

```
\fancyfoot[R]{\fontsize{9pt}{1em}\color{gray}\thepage}
```

> On peut tout faire avec `fancyhdr` : inclure les dates, numéros des sections, chapitres, équations ou figures dans les en-têtes ou pieds de page. Tout est déjà très bien expliqué ici.

5.3 Listes

Il existe trois types de listes principaux en LaTeX, les listes non-numérotées, les listes numérotées et les listes dont les items démarrent par un mot en gras. Ces listes sont gérées par 3 environnements : `itemize`, `enumerate` ou `description`. Il est tout à fait possible d'imbriquer des listes, de n'importe quel type. Pour mettre des éléments dans une liste, facile, il suffit d'ajouter des `\item`. Par exemple,

```
\begin{itemize}
\item ceci est le premier item
\item ceci est le 2eme
\end{itemize}
```

donne :

- ceci est le premier item
- ceci est le 2eme

Un autre exemple :

```
\begin{description}
\item[\TeX] The \TeX{}book
\item[\LaTeX] deux livres importants :
\begin{enumerate}
\item \LaTeX{} : A Document preparation
System
\item The \LaTeX{} Companion
\end{enumerate}
\end{description}
```

> Comme pour tout, il est possible de modifier le style des listes. Le package `enumitem` permet de modifier le style des listes non numérotées. Il permet de changer le symbole utilisé par défaut pour chaque item, avec la commande `\renewcommand{\labelitemi}{\textendash}` à mettre en préambule. Ici, `labelitemi` correspond au premier niveau d'une liste imbriquée (le deuxième serait `labelitemii`) et `\textendash` est un –.
> `\setlist{nosep}` permet d'enlever tout espace entre les lignes, ou `\setlist{itemsep=0.3cm}` permet de le spécifier.

5.4 Langages

Les caractères et les règles typographiques (par exemple, les espacements autour des :) sont différentes en fonction des langages écrits. LaTeX gère toutes les règles typographiques automatiquement ; il suffit de lui spécifier pour quel langage.

Le package le plus populaire pour utiliser un langage qui n'est pas l'anglais est `babel`. Pour l'utiliser, inclure `\usepackage[francais]{babel}` (ou une autre langue) au préambule, et `\selectlanguage{french}` ou `\selectlanguage{english}` dans le corps de texte, lorsque vous souhaitez changer de langue.

L'autre option est `polyglossia`, qui fonctionne de manière similaire. En 2021, il semble que les deux packages soient aussi performants l'un que l'autre.

5.5 *Templates*, *templates* de journaux scientifiques

Une fois que vous avez compris les bases (et à la fin de la formation, ce devrait être le cas !), le plus aisé pour commencer est d'utiliser un *template*. Dans un *template* LaTeX, toute la forme est déjà paramétrée – il ne vous reste plus qu'à remplir.

Il existe deux principaux répertoires de *templates*, celui d'Overleaf et celui de Latex-Templates. Il existe des *templates* pour tout : livres, posters, calendriers, lettres, thèses, CV, rapports, journaux, présentations, etc. Il y en a des moches et des jolis, vous trouverez forcément un *template* pour vous !

Une fois devenu·es familier·es avec votre *template* préféré, vous pourrez commencer à le modifier et à le personnaliser. Pour ca, stackexchange et google sont vos amis !

L'un des gros avantages de LaTeX est qu'on peut écrire des articles scientifiques sans se soucier de la mise en forme du journal visé ; il suffit ensuite d'incorporer le corps du texte dans le *template* du journal. En effet, la grande majorité des journaux scientifiques proposent aussi leurs *templates*, qui sont trouvables sur Overleaf ou sur leurs sites respectifs. Souvent, ces *templates* sont basés sur des classes spécifiques (par exemple, la classe AGU ou Elsevier), qui sont elles même basées sur la classe article.

5.6 Style un peu plus avancé

Ici je vous donne quelques astuces pour commencer à personnaliser vos documents LaTeX.

Police Il est très facile de changer la police utilisée, pour le corps de texte ou pour certains éléments seulement, comme les titres (c'est le cas dans ce document). L'option la plus simple est d'utiliser une police qui est déjà sous package. Toutes les polices disponibles sont listées dans ce catalogue. Ici par exemple, j'utilise `\usepackage[opentype]{sourcesanspro}`.

À compiler avec LuaTeX !

Une fois que le package est chargé, il suffit d'utiliser `\sourcesanspro`, ou encore `\sourcesansprobold` ou `\sourcesansprolight` pour changer la fonte du texte localement, à l'intérieur d'un environnement ou entre `{}`.

Il est aussi possible de changer la police par défaut en utilisant `\setmainfont{sourcesanspro}` en préambule.

Opentype est un format de fonte (otf) qu'il est préférable d'employer si disponible

Vous pouvez incorporer n'importe quelle police installée sur votre ordinateur avec le package `fontspec`. Dans ce cas, vous pouvez changer la police par défaut directement, mais pour changer de fonte localement il faut redéfinir les commandes, en utilisant : `\newfontfamily\myfont[Ligatures=TeX]{sourcesanspro}` par exemple. Je trouve ca plus simple d'utiliser le package directement.

Couleurs On gère les couleurs avec le package `xcolor`. Quelques couleurs sont déjà définies de manière native, comme darkgray ou orange. Il est possible de charger 400+ couleurs prédéfinies avec `\usepackage[svgnames,dvipsnames]{xcolor}`. Pour changer la couleur du texte, soit utiliser `\textcolor{darkray}{texte}`, soit `{\color{darkgray} texte}`[5] .

[5] Ne pas oublier les {} !

Il est aussi possible de définir ses propres couleurs avec, en préambule, `\definecolor{name}{model}{color-spec}`, le mode étant RGB (entre 0 et 255), rgb (entre 0 et 1), cmyk ou html. Par exemple ici, j'ai défini la couleur bleue par : `\definecolor{myblue}{RGB}{1,78,158}`.

Style des titres Avec le package `titlesec`, il est très facile de personnaliser vos titres. Par exemple ici, j'utilise :

```
\titleformat*{\section}{\fontsize{14pt}{1em}\bfseries
↪ \sourcesanspro\color{myblue}}{}{}
\titleformat*{\subsection}{\fontsize{12pt}{1em}\bfseries\sourcesanspro}{}{}
```

6 Bibliographie

6.1 Comment ca marche ?

LaTeX utilise BibTex pour gérer les références biblios. Les références sont stockées, avec un format particulier, dans un fichier (souvent .bib). Dans ce fichier .bib, chaque référence est caractérisée par une clé (par exemple, 'nom_année'). Dans notre fichier .tex, pour citer dans le texte, on appelle la clé des références qui nous intéressent. BibTex gère ensuite automatiquement la création d'une bibliographie, au format souhaité, qui contient l'ensemble des références citées.

Le format basique d'une référence listée dans le fichier .bib (ici, la clé est 'smith_2017') :

```
@article{smith_2017,
        author = {Smith, John},
        title = {Some paper},
        year = {2017},
        journal = {Journal of Research},
        volume = {2},
        number = {1},
        pages = {1--69}
}
```

Le package permettant de gérer la bibliographie le plus utilisé est natbib. Après l'avoir importé, on peut définir le style de bibliographie souhaité. Par défaut, il s'agit souvent d'apalike (format APA). Il est aussi possible d'importer un style particulier (fichier .cls), ce qui est souvent le cas lorsque l'on utilise des templates d'articles.

Dans le texte, pour citer avec natbib, on utilisera les commandes `\citet{smith_2017}` (dans le texte, année entre parenthèses) ou `\citep` (tout entre parenthèses). Pour ajouter du texte avant ou après, dans les parenthèses, mettre entre crochet. Par exemple, `\citep[blabla1][blabla2]{smith_2017}`.

À l'endroit ou l'on veut introduire la bibliographie, il suffit d'appeler notre fichier .bib avec la commande `\bibliography`.

En résumé :

```
\usepackage{natbib}
\bibliographystyle{apalike}
\begin{document}
        Blablabla \citep{smith_2017,doe_2021}.
        \bibliography{biblio.bib}
\end{document}
```

> Attention ! Dans certains templates (GRL par exemple), ce n'est pas natbib qui est utilisé, et les commandes pour citer changent donc.

6.2 LaTeX et outils bibliographiques

On va se concentrer ici sur Zotero, parce que c'est libre d'accès. Zotero est un gestionnaire de bibliographie lié à votre navigateur. Avec cet outil, vous pouvez gérer, conserver, annoter, classer, trier, taguer, exporter,... votre bibliographie, et le tout quasi-automatiquement. Si c'est pas déjà fait, installez le !

De manière native, il est possible d'exporter une bibliographie .bib directement (clic droit, exporter,...). Mais les clés ne sont pas gérées automatiquement. Et c'est bien plus pratique de n'avoir rien à faire, non ? Avec Better BibTex, les clés sont gérées et l'export est automatique dès qu'une référence est ajoutée à Zotero (entre autres). Vous pouvez utiliser aussi le 'drag-and-drop' pour insérer une bibliographie rapidement.

Après l'avoir installé, paramétrer l'export automatique dans `Édition>Préférences>Better Bibtex>Synchronisation`.

> Je conseille d'exporter la bibliographie intégrale dans un seul fichier qui sera appelé dans l'ensemble de vos documents `tex`. Pour, par exemple, les articles, lorsque seule une partie des références est nécessaire, il est possible d'utiliser l'outil bibtool pour créer un fichier .bib tronqué :
> `bibtool -x filename.aux -o filename.bib`, `filename` étant le nom du fichier .tex.

> Lorsque le fichier `.bib` se met à jour, il faut plusieurs compilations pour que les fichiers auxiliaires s'actualisent. Souvent, il faut compiler latex deux fois, la bibliographie deux fois, et compiler latex encore une fois. Répéter les ce cycle autant de fois que nécessaire.

7 En pratique

7.1 Organisation des fichiers et dossiers

On l'a déjà vu, une compilation LaTeX produit un PDF, mais aussi de nombreux fichiers auxiliaires. Si vous mettez toutes les figures dans le même dossier, c'est le bordel complet.

Il y a autant de manières de s'organiser que de personnes, mais ici je vous partage ma pratique. Chaque projet de document (un article, une thèse, ou une lettre) a son dossier réservé, dans lequel ne sera placé que ce qui concerne la rédaction.

Pour un projet simple, je place dans ce dossier le fichier `tex` qui contiendra le préambule et ma rédaction. Si j'ai un fichier `tex` supplémentaire (comme les suppléments d'un article), il sera placé là aussi. Toutes les figures seront placées dans un sous-dossier `fig`. Toutes les anciennes versions seront placées dans un sous-dossier `old`.

```
.
`-- project
|-- myproject.tex
|-- myproject.aux
|-- ...
|-- fig
|   |-- fig1.pdf
|   `-- fig2.pdf
`-- old
   |-- myproject_v-1.tex
   `-- myproject_v-2.tex
```

Lorsque le projet est plus complexe, et qu'il inclue plusieurs fichiers `tex` (plusieurs chapitres par exemple), alors chaque chapitre a son dossier dédié, dans lequel seront inclus les dossiers `old` et `fig`.

```
.
`-- project
|-- main.tex
|-- main.aux
|-- ...
|-- intro
|   `-- intro.tex
|-- chap1
|   |-- chap1.tex
|   |-- ...
|   |-- fig
|   |   |-- fig1.1.pdf
|   |   `-- fig1.2.pdf
|   `-- old
```

```
|       |-- chap1_v-1.tex
|       `-- chap1_v-2.tex
|-- chap2
|   |-- chap2.tex
|   |-- ...
|   |-- fig
|   |   |-- fig2.1.pdf
|   |   `-- fig2.2.pdf
|   `-- old
|       |-- chap2_v-1.tex
|       `-- chap2_v-2.tex
`-- conclu
    |-- conclu.tex
    `-- old
       `-- conclu_v-1.tex
```

Avec ce type d'organisation, tous les fichiers auxiliaires sont dans le dossier principal, mais ce n'est pas si gênant puisque vous n'accédez au `.tex` et au PDF qu'avec TeXstudio ou presque.

> Il est aussi possible de spécifier un sous-dossier dans lequel tous les fichiers auxiliaires seront créés (+ le PDF final). Pour ce faire, il suffit de rajouter `-output-directory=sousdossier` à la commande de compilation (`Options > Configure TeXstudio > Compilations` sous TeXstudio). Parcontre, je ne suis pas sûre que ce système soit très stable pour des gros projets.

7.2 Markdown

Les bases

Markdown est un langage de balisage léger, c'est à dire qu'il utilise une syntaxe simple, conçue pour être aisée à saisir avec un éditeur de texte simple, et facile à lire dans sa forme non formatée. En gros, très différent de LaTeX qui est peut être assez complexe à lire pour des novices. Enfait, vous avez surement déjà écrit en Markdown si, par exemple, vous formattez du texte sur Whatsapp ou Messenger. Écrire `*bold*` ou `_italic_` (pour **bold** ou *italic*), c'est du Markdown ! Markdown est aussi le language employé pour la documentation sur Github par exemple.

Avec Markdown, on peut écrire des documents avec une mise en page simple, sans se prendre la tête sur la forme. Markdown gère les sections multiples, les références biblio (se basant sur un fichier .bib), les étiquettes, l'insertion de code source, les figures simples, et même quelques symboles et formules mathématiques simples. On peut l'exporter facilement en PDF. Bref, tout ce dont on a besoin pour faire des documents simples. Oui, vous l'aurez compris, Mardown c'est simple.

> Comment écrire du Markdown ? Oui certes, on peut le faire avec n'importe quel éditeur de texte. Mais c'est plus sympa d'avoir une interface pratique et étudiée pour non ? Mon éditeur Mardown préféré est Zettlr : c'est beau, c'est pratique, c'est efficace. Pour un éditeur encore plus puissant, qui permet le travail collaboratif en ligne et la gestion de version, tournez -vous vers Atom ou Visual Studio Code. Vous pouvez aussi regarder du côté de Mark Text si Zettlr ne vous convient pas. Sinon, des éditeurs plus classiques comme Pycharm gèrent aussi le Markdown.

Markdown et LaTeX

Alors, pourquoi je parle de Markdown dans une formation pour LaTeX ? Déjà, Mardown est très utile lorsque peu vous importe la forme de ce que vous rédigez. Si vous rédigez des notes pour vous même, je vous conseille de le faire en Markdown – et donc ni en LaTeX, ni en *Word* ou *OpenOffice*.

Ensuite, et c'est pour ca que Markdown est intéressant avec LaTeX : il est possible d'utiliser du Markdown dans un environnement LaTeX ! Tout ca se fait grâce au `package{markdown}`. Il permet d'inclure du Markdown dans un document LaTeX, qui sera automatiquement mis en forme avec les spécifications indiquées en préambule du fichier `tex`. Par exemple, le code ci dessous donne le paragraphe suivant.

```
\begin{markdown}
##### Ceci est écrit en Markdown

Pareil qu'en \LaTeX{}, les paragraphs sont déparés par des lignes.

Je peux écrire facilement en *italique*, **gras**, ou `monospace`.
 Je peux faire des listes (sans m'embarasser avec des
  ↪  environnements !!!!) :

* un
* deux
* ou trois
```

Ceci est écrit en Markdown Pareil qu'en LaTeX, les paragraphs sont déparés par des lignes.

Je peux écrire facilement en *italique*, **gras**, ou `monospace`. Je peux faire des listes (sans m'embarasser avec des environnements !!!!) :

- un
- deux
- ou trois

Il est aussi possible d'inclure du LaTeX directement dans l'environnement Markdown (ajouter l'option *hybrid* au package). Par exemple, pour inclure une figure, une formule, un environnement spécifique, une citation, ou autre. Qu'est ce que ca veut dire ? Que **vous pouvez écrire vos documents LaTeX en Markdown**. Le Markdown, c'est plus simple à écrire (cool pour débuter !), et plus facile à partager (car sans la syntaxe assez lourde de LaTeX). Et ca, ca peut être très utile !

Ou une note en marge.

La seule différence de la syntaxe Markdown est pour les citations : `\citep{key1,key2` est remplacé par `[@key1 ; @key2]` et `\citet{key1` par `@key1`.

Collaborer avec Markdown

Voir paragraphe 7.4

Quelques astuces pratiques pour écrire en Markdown

Autocomplétion des références : L'autocomplétion dépend des éditeurs. Dans Zettlr, il suffit d'indiquer le fichier de références `.bib`, comme expliqué ici. Avec avec Visual Studio Code et Atom, il faut installer des packages supplémentaires, comme indiqué ici et là.

Authorea : Authorea, c'est comme Overleaf mais ca supporte le Markdown, en plus du LaTeXet de plein d'autres langages. L'interface Markdown est WYSIWYG, les formules LaTeXcomprises, l'autocomplétion des références est OK, il est possible de faire des commentaires, et de suivre les modifications. Voir Fig. 1 pour un exemple sympa.

> Pour aller plus loin en Markdown, vous pouvez regarder cet article par Overleaf, et cette *cheatsheet*.
>
> Enfin, il est aussi possible d'écrire toute une thèse, un livre, ou un gros projet en R Markdown. Ca rend très bien, mais je ne vais pas m'étendre ici. Si ca vous intéresse, regardez du côté du *R Markdown Cookbook*

Jusqu'ici, j'écris toujours en Markdown !

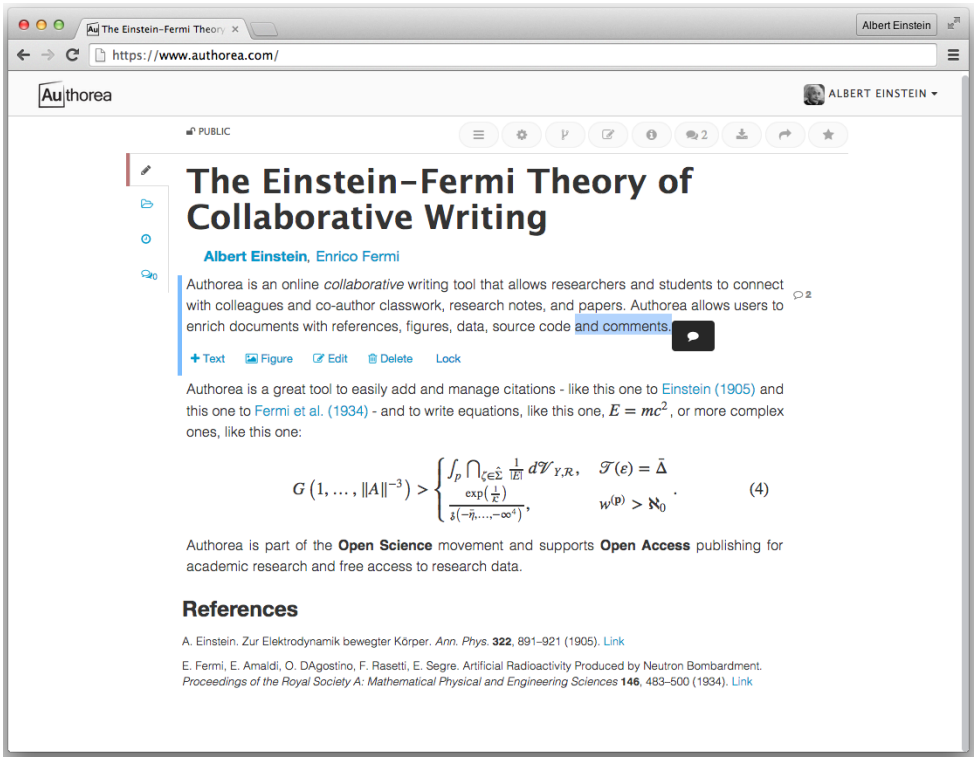

FIG. 1 : Un imprécran d'Authorea.

7.3 Suivi des modifications

Dans ce paragraphe je parle seulement du suivi des modifications (donc en gros afficher les différences), pas de la prise en compte des commentaires, corrections, etc. Il existe plusieurs outils pour afficher les modifications sur un document créé sous LaTeX. On peut les regrouper en deux types différents : ceux qui affichent les modifications sur le .tex, et ceux qui les affichent sur le PDF. Les deux vous seront utiles, mais pas pour les mêmes utilisations.

Afficher les modifications sur un fichier .tex directement peut être très utile lors que vous travaillez avec des collaborateur·ices qui utilisent aussi LaTeX. L'option que je préfère est la version *track change* d'Overleaf, mais qui n'est parcontre pas libre d'accès (de nombreuses universités incluent cependant cette option).
Si cette option n'est pas possible pour vous, je recommande d'utiliser les options de comparaison de fichiers de TeXstudio, accessibles dans
`Fichier>SVN>Montrer la différence entre deux fichiers`.

Pour afficher les changements apparus dans le PDF directement, ce qui est très utile pour soumettre des révisions d'articles par exemple, le plus simple est d'utiliser latexdiff (inclus dans TeXlive). Latexdiff permet de créer un nouveau fichier `tex`, avec `latexdiff old.tex new.tex > diff.tex`. Ce fichier `tex` doit ensuite être compilé pour obtenir le PDF avec modifications affichées (suppression, ajouts, etc.). Parcontre, latexdiff nécéssite l'utilisation d'un invite de commandes, même sur Windows !

> LaTeXdiff n'est pas installé dans les distributions LaTeX sous Windows par défaut. Pour l'installer, il faut d'abord installer Perl, et dézipper les fichiers latexdiff dans le dossier `Perl > bin` (dans `C :` par défaut).

Il est aussi possible d'utiliser les packages change ou trackchanges, mais ceux-ci nécessitent d'utiliser des commandes à l'intérieur du document `tex` pour déclarer les modifications, ce que je trouve particulièrement lourd.

7.4 Travail collaboratif

Le conseil à suivre absolument : **rédiger directement en LaTeX** ou Markdown, pour un **énorme** gain de temps. Que la rédaction concerne les rapports, les articles, ou (et surtout) la thèse.

Ensuite, deux options possibles. Soit vos collaborateur·ices utilisent déjà LaTeX, soit ils ou elles ne l'utilisent pas. Bien entendu, le plus simple dans tous les cas est de suggérer les modifications directement sur le PDF, mais c'est moins pratique lorsque les suggestions sont nombreuses (puisque vous devrez faire les corrections manuellement).

Collaborateur·ices utilisent LaTeX Dans ce cas, l'option que je préfère est de partager la source sur Overleaf. Ainsi, tou·tes les collaborateur·ices ont accès à la source et peuvent la modifier, et on peut même suivre les modifications si on a accès à l'option *track-change*.

Sinon, comme suggéré plus haut, simplement utiliser les options de TeXstudio.

Collaborateur·ices utilisent *Word* ou *Open office* (et l'option annotation de PDF n'est pas envisageable).

La solution la plus simple, c'est d'utiliser *Word* ou *Open office* pour rédiger votre document, mais seulement si c'est techniquement possible ! Par exemple, ce n'est pas possible pour un rapport long, une thèse, etc. Solution envisageable seulement pour les articles ou documents de quelques pages donc.

Une solution consiste à partager le corps du texte LaTeX (donc le .tex sans le préambule, ou seulement les parties qui vous intéressent) sous *Word* ou *Open office*. Toutes les commandes seront déjà en place (figures, références, étiquettes, titres), et vos collaborateur·ices pourront procéder comme ils en auront l'habitude. Vous n'aurez qu'à réimporter (copier-coller) ensuite la nouvelle version dans votre fichier `tex`. Pour les figures, qui n'apparaitront pas dans le .doc, il suffit de partager le PDF. *Toutefois, dans ce cas, le document n'est pas mis en page correctement.*

La solution que je privilégie, et que je vous conseille, par exemple pour une thèse, est d'écrire le corps du texte en Markdown. Vous trouverez un exemple détaillé dans le fichier `ex_markdown.tex`. Dans ce document `.tex`, j'incorpore du Markdown avec la commande `\markdownInput{text.md}`. Vous écrivez alors votre texte directement en Markdown (et en LaTeX) dans le document `text.md`, grâce à un éditeur Markdown spécifique ou directement dans Word. Eh oui ! Avec le plugin Writage pour Word, vous pouvez ouvrir, éditer et convertir des documents `.md` en `.docx`. Si vous utilisez openoffice, la solution la plus pratique est d'utiliser Pandoc, avec la commande `pandoc -s -o text.odt text.md` pour convertir de `.md` en `.odt`..

La solution Authorea, qui permet de partager un document Markdown ou LaTeX en ligne, avec une interface type Word, qui permet l'ajout de commentaires et le suivi des modifications. Je l'ai seulement testé, mais *a priori* ca à l'air très pratique et très puissant. Ca supporte aussi l'exportation en PDF, word, ou autre.

Biensur, que vous écriviez en LaTeX ou en Markdown, les citations bibliographiques auront toujours le format commande + clé (`\citep{nom_année}` ou `@nom_année`), mais la différence avec un (Nom, année) ne devrait pas perturber lecteurs et lectrices.

> Si vous utilisez des logiciels de gestion de versions (comme git ou subversion), il est tout à fait possible de versionner et de voir les modifications de vos documents `tex` ou Markdown. Je vous invite à vous référer à ce wikibook.

7.5 Outils pratiques pas encore mentionnés

Todonotes Un de mes packages préférés. Permet d'incorporer des *todo* notes, et de faire des *todo lists*!

```
\usepackage[colorinlistoftodos,french]{todonotes}
\todo[inline,author=Jean,color=orange]{À écrire.}
```

Jean : À écrire.

Théa
faire un truc

showkeys Visualiser sur le PDF les labels et les références.

```
\usepackage[draft,color]{showkeys}% draft or final
```

refcheck Signaler sur le PDF les labels inutilisés (doit être chargé à la fin du préambule)

```
\usepackage{refcheck}
```

siunitx Met en forme nombres et unités.

```
\usepackage{siunitx}
\SI{6}{m.kg/(s^3.A)}
```

exercise, exsheets Pour rédiger des exercices et exporter les réponses dans des fichiers séparés (entre autres)

```
\usepackage{exercise,exsheets}
```

infobulle Pour inclure des 'infobulles'.

```
\usepackage{exercise}
```

Pandoc Convertis un document de n'importe quel format (ou presque, LaTeX, markdown, word...) en n'importe quel format.

microtype améliore la typographie LaTeX pour la rendre juste parfaite. Parcontre, il semble ne pas être stable avec LuaTex. Les extensions sont infinies, mais les options conseillées sont généralement :

```
\usepackage[
protrusion=true,
activate={true,nocompatibility},
final,
tracking=true,
kerning=true,
spacing=true,
factor=1100]{microtype}
\SetTracking{encoding={*}, shape=sc}{40}
```

7.6 Les packages que j'importe par défaut

Si vous n'utilisez pas un package, commentez le.

```
\usepackage{polyglossia} % the french language, or any other language
\setmainlanguage[]{french}
\usepackage[round]{natbib}  % bibliography

%---- floats
\usepackage{graphicx}  % figures
\usepackage{subfig}  % subfigures
\usepackage{wrapfig}  % wrapped figures
\usepackage[font=footnotesize,labelfont={bf}]{caption}  % captions
\usepackage{array}  % arrays
\usepackage{float}  % Improved interface for floating objects
\usepackage{pdfpages}  % include PDF files
```

```
\usepackage{tabularx}  % nicer arrays
\usepackage{colortbl}  % color array rows or columns

%---- lists
\usepackage{enumitem}  % lists
\renewcommand{\labelitemi}{\textendash}  % change 1st items label
\setlist{nosep}  % no space in between items

%---- basic style
\usepackage[headsep=0.8cm,
top=2.5cm,bottom=2.5cm,
left=3cm,right=3cm]{geometry}  %left= inner et right=outer, set margin
↪  properties
\usepackage{titlesec}  % sections formatting
\usepackage{titletoc}  % table of content
\usepackage{fancyhdr}  % headers and footers
\let\footruleskip\undefined  % remove footruleskip

%---- more advanced style
\usepackage[
                protrusion=true,
                activate={true,nocompatibility},
                final,
                tracking=true,
                kerning=true,
                spacing=true,
                factor=1100]{microtype}  % nicer spacing
\usepackage[opentype]{sourcesanspro}
\usepackage{setspace}  % setting the spacing between lines
\usepackage[colorinlistoftodos,french]{todonotes}  % todo
\usepackage{marginnote}  % notes in the margin
\usepackage{newunicodechar}  % for the interpunct
\newunicodechar{·}{\kern-.25em\textperiodcentered\kern-.25em}

%---- colors
\usepackage[svgnames]{xcolor}  % define colors
\definecolor{myblue}{RGB}{1,78,158}
\definecolor{myblue2}{RGB}{70,96,151}

%---- code, maths, markdown
\usepackage{minted}  % include source code
\usemintedstyle{tango}  % minted color style
\usepackage{amsmath}  % maths
%\usepackage[hashEnumerators,smartEllipses,hybrid,citations,underscores=false]{markdown}
↪  %loads enumerate which is in conflict with enumitem. If loaded,
↪  replace \usepackage{enumitem} by :
%                       \usepackage[loadonly]{enumitem}
%                       \newlist{myitem}{itemize}{2}
%                       \setlist*[myitem]{nosep,label=\textendash}

%---- hyperlinks etc.
\usepackage{url}  % URL links
\usepackage[backref=page,bookmarks=true,
        bookmarksopen=true,colorlinks=true,
        linkcolor=myblue,citecolor=myblue,
        hidelinks]{hyperref}  % hyperlinks, cross-referencing, etc.
\hypersetup{colorlinks,
        linkcolor=myblue,
```

```
        urlcolor=myblue,
        citecolor=myblue2}
\usepackage{bookmark}  % PDF bookmarks
```