Template de thèse minimaliste

Antoine Bérut

15 février 2022

Table des matières

Introduction

De tout temps, les doctorants et doctorantes ont voulu rédiger leur thèse en LaTeX sans avoir à perdre trop de temps sur l'élaboration d'un template fonctionnel. Malheureusement, faire un template clef-en-main qui réponde aux attentes de tout le monde est illusoire. Néanmoins, je vous partage le mien, parce qu'on ne sait jamais, peut-être qu'il vous conviendra et que vous n'aurez pas à modifier trop de choses dedans pour avoir un rendu qui vous plaise ! (Et au passage je discute quelques points souvent utiles pour les thèses de sciences : comment gérer les unités, et comment faire des figures avec des sous-figures imbriquées qui sont citables dans le texte.)

Chapitre 1

Comment fonctionne ce template

On est trop souvent imprécis
lorsqu'on fait une citation.

Quelqu'un, un jour.

Dans ce chapitre je donne un bref aperçu de l'organisation du template, des packages utilisés, et des éléments modifiables pour l'adapter à votre convenance.

1.1 Un template fractionné

1.1.1 Pourquoi ?

Le template est fractionné en différents morceaux ce qui permet de compiler chaque chapitre séparément, tout en conservant la mise en page qui sera celle de la thèse finale, ce qui est très pratique si comme beaucoup de monde vous commencez par le chapitre 3 et terminez votre rédaction par l'introduction[1], mais que vous voulez quand même voir à quoi ça ressemblera avant la fin. Toutes les citations internes au chapitre \ref{} et les éléments bibliographique \cite{} seront correctement interprétés, seules les citations entre chapitres resteront vides tant que l'intégralité des chapitres ne sera pas compilée en même temps.

1.1.2 Comment ?

Les différents morceaux sont les suivants :

- dans le dossier principal : un fichier "preambule.tex" qui contient toutes les commandes à insérer au tout début du document. C'est notamment là qu'on chargera tous les packages utiles qui seront décrits dans le paragraphe 1.2. Ce fichier n'est pas à compiler.
- dans le dossier principal : un fichier "biblio.bib" qui contient l'intégralité des références bibliographiques utilisées dans la thèse.

1. Statistique purement spéculative inférée à partir d'un échantillon représentatif de deux individus.

- des sous-dossiers pour chaque chapitre contenant chacun un fichier “nom_du_chapitre.tex” et un dossier “Figures” qui contient... les figures ! Ces fichiers .tex ne contiennent que le coeur du texte et n’ont pas vocation à être compilés directement.
- dans le dossier principal (où se trouve “preambule.tex”), des fichiers “compilation_chapitre_X.tex” et “compilation_these.tex” qui comme leur nom l’indique, sont ceux à compiler si l’on veut obtenir un chapitre en particulier ou la thèse complète (compilation standard : PDFLatex + Bibtex + PDFLatex 2 fois).

1.2 Liste des packages utilisés

1.2.1 Packages indispensables

- Encodages des caractères :

```
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{lmodern}
```

  En particulier, cela assure qu’il n’y aura pas de ligatures de faites entre deux caractères successifs (comme “fi” par exemple), ce qui rendrait la recherche dans le document de mots contenants ces ligatures impossible.

- Rédiger un document en français (garantit à la fois que les noms des sections seront bien en français, mais aussi que les règles typographiques françaises seront bien respectées [1]) :

```
\usepackage[french]{babel}
```

  (remplacez “french” par “english” si vous écrivez en anglais)

- Gestion de la mise en page et des marges :

```
\usepackage{geometry}
\geometry{a4paper}
```

  Des exemples de personnalisation des marges seront donnés dans le paragraphe 1.3.1.

- Gestion des citations :

```
\usepackage{cite}
```

- Gestion des liens hypertextes :

```
\usepackage{hyperref}
\usepackage{bookmark}
```

  Un exemple de personnalisation des couleurs des liens est donné dans le paragraphe 1.3.2.

1.2.2 Packages utilitaires mais dispensables

- Choix de la “profondeur” de la table des matières (*i.e.* quels éléments sont indiqués dedans) et la “profondeur” de la numérotation des sections.

```
\setcounter{tocdepth}{2}
\setcounter{secnumdepth}{2}
```

Le numéro indique le degré de profondeur souhaité : 0 = chapitre, 1 = section, 2 = subsection, 3 = subsubsection. Ici on demande donc que les subsubsection ne soient pas numérotées et n'apparaissent pas dans la table des matières (contrairement aux chapitres, sections, et subsections).

- Gestion des symboles mathématiques :

  ```
  \usepackage{amsmath,amssymb,amsfonts,amsthm}
  ```

  Ces packages rajoutent différents symboles et polices mathématiques fréquemment utilisés.

- Gestion des unités :

  ```
  \usepackage{siunitx}
  ```

  Le package a énormément d'options, pour un descriptif détaillé je vous invite à vous reporter à la documentation [2]. Quelques exemples seront donnés dans le paragraphe 1.3.3.

- Gestion des figures :

  ```
  \usepackage{graphicx}
  \usepackage{caption}
  \usepackage{subcaption}
  ```

  Le package "subcaption" permet de gérer des figures avec plusieurs sous figures (il remplace le package "subfig" qui n'est plus mis à jour) et requiert le package "caption". Un exemple est donné dans le paragraphe 1.3.4.

- Gestion des tableaux et listes :

  ```
  \usepackage{booktabs}
  \usepackage{array}
  \usepackage{paralist}
  ```

- Gestion de la table des matières :

  ```
  \usepackage[nottoc]{tocbibind}
  \usepackage{tocloft}
  ```

  La première commande assure que la bibliographie apparaisse dans la table des matières, mais que la table des matières n'apparaissent pas elle-même dans la table des matières. La seconde permet une meilleure gestion des espacements dans la table des matières.

- Gestion de la couleur :

  ```
  \usepackage{xcolor}
  ```

1.2.3 Packages cosmétiques

- Jolis en-têtes.

```
\usepackage{fancyhdr}
\usepackage{emptypage}
```

La deuxième ligne garantit que les pages “blanches” (comme celle entre la table des matières et le chapitre 1 par exemple) seront réellement blanches (*i.e.* sans en-tête ni pied de page). Des exemples de personnalisation seront donnés dans le paragraphe 1.3.5

- Jolis chapeaux de début de chapitre.

  ```
  \usepackage[Lenny]{fncychap}
  ```

  Vous pouvez personnaliser en remplaçant l’argument “Lenny” par (au choix) : Sonny, Glenn, Conny, Rejne, Bjarne, Bjornstrup [3].

- Rajout de symboles (note de musique, bullet point, etc.) :

  ```
  \usepackage{textcomp}
  ```

- Possibilités de mettre des citations en début de chapitre :

  ```
  \usepackage{epigraph}
  ```

  Un exemple est donné dans le paragraphe 1.3.6.

1.3 Exemples et personnalisation

1.3.1 Mise en page

La personnalisation de la mise en page peut se faire de la façon suivante :

```
\geometry{%
  a4paper,                % format de papier
% Définition des marges :
  left= 3cm,             % marge intérieure à la page
  right = 2cm,           % marge extérieure
  top = 3cm,
  bottom = 3cm,
% En-tête et pied de page :
  headheight=6mm,         % espace réservé à l'en-tête dans la marge top
  %headsep=3mm,            % espace entre le corps et l'en-tête
  %footskip=9mm            % espace entre le corps et le pied de page
}
```

Dans l’hypothèse où la thèse est souvent imprimée et reliée, on a spécifié ici que les marges soient plus importantes du côté intérieur des pages.

1.3.2 Choix des couleurs des liens hypertextes

À l’aide du package “xcolor” on peut définir différentes couleurs, par exemple le bleu qui est utilisé pour les liens dans ce document :

```
\definecolor{linkcolor}{rgb}{0,0,0.6}
```

On peut ensuite demander au package “hyperref” d’attribuer cette couleur (ou différentes couleurs) aux différents types de liens :

```
\hypersetup{
colorlinks=true, % colore les liens au lieu de les encadrer
urlcolor=linkcolor, % choix de la couleur des liens URL
linkcolor= linkcolor, % choix de la couleur des liens internes
citecolor=linkcolor % choix de la couleur des liens de citations
}
```

1.3.3 Gestion des unités avec SI-unitx

La package “siunitx” gère les unités de façon uniforme sur le document (et c’est l’une des rares façon de tracer un μ droit pour les unités : µm). Par exemple, si l’on veut parler de la vitesse du son dans l’air à température ambiante au lieu d’écrire “$340 m.s^{-1}$”, on écrira “\SI{340}{\meter\per\second}”, ce qui donnera $340\,\mathrm{m \cdot s^{-1}}$. Si l’on souhaite juste écrire le nom de l’unité sans mettre de chiffre devant, il suffit d’écrire “\si{\meter\per\second}” ce qui donnera $\mathrm{m \cdot s^{-1}}$.

Le choix du séparateur entre unité (le point médian dans l’exemple ci-dessus) se fait à l’aide de la commande suivante :

```
\sisetup{inter-unit-product=\ensuremath{{}\cdot{}}}
```

Ainsi, si l’on souhaite changer le séparateur dans tout le document il suffira de remplacer “\cdot” par ce qu’on veut.

Il est possible de définir des unités personnalisées (par exemple si on a beaucoup de grandeurs en mètres par seconde, on pourra utiliser :

```
\DeclareSIUnit\vitesse{\meter\per\second}
```

Et ainsi on pourra écrire : “\SI{340}{\vitesse}” ce qui donnera $340\,\mathrm{m \cdot s^{-1}}$.

Il est également possible de choisir comment les incertitudes sont gérées à l’aide de la commande :

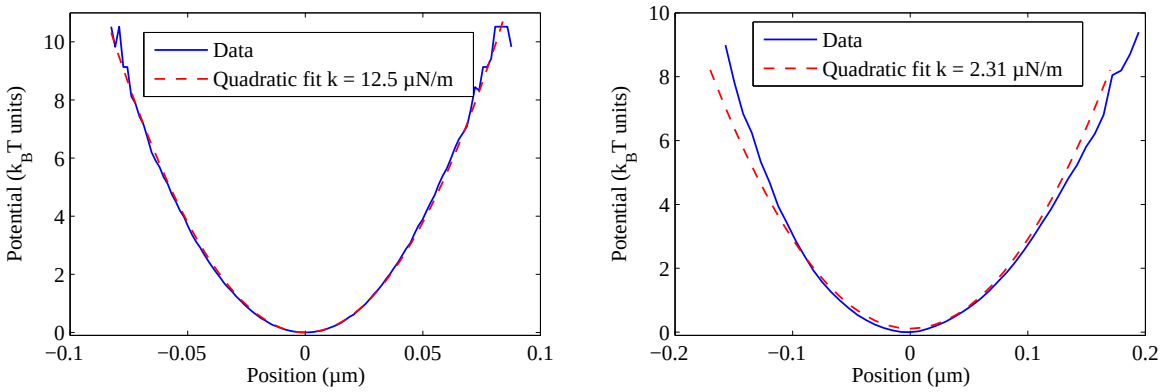
```
\sisetup{separate-uncertainty=true,multi-part-units=single}
```

Ici, il est demandé que les incertitudes soient indiquées de façon séparées de la valeur (par défaut avec un symbole $\pm$ entre les deux), et que l’unité ne soit pas répétée 2 fois (une fois pour la valeur et une fois pour l’incertitude). Si on écrit “\SI{340(5)}{\vitesse}” on obtiendra donc : $340 \pm 5\,\mathrm{m \cdot s^{-1}}$.

1.3.4 Sous-figures

L’intérêt de faire des figures avec une organisation en sous-figure directement dans le code, c’est qu’on peut faire des références à chaque sous-figure directement sans avoir besoin de rajouter à la main des (a) ou (b) après sa commande \ref{figure} ! Par exemple, dans la figure 1.1 on peut faire des renvois vers chacune des sous-figure indépendamment comme 1.1a et 1.1b. Elle est obtenue à l’aide du code suivant :

```
\begin{figure}[ht!]
	\centering
	\begin{subfigure}[c]{0.5\textwidth}
		\includegraphics[width=\textwidth]{Chapitre1/Figures/exemple_potentiel_quadratic.pdf}
		\caption{A single trap}
		\label{intro:fig:potentials_OK}
	\end{subfigure}%
	%add desired spacing between images, e. g. ~, \quad, \qquad, \hfill etc.
	%(or a blank line to force the subfigure onto a new line)
	\begin{subfigure}[c]{0.5\textwidth}
		\includegraphics[width=\textwidth]{Chapitre1/Figures/exemple_potentiel_quadratic_tilte.pdf}
		\caption{One of two traps created by an AOD}
		\label{intro:fig:potentials_tilte}
	\end{subfigure}
	\caption{Exemple de deux figures.}\label{intro:fig:potentials}
\end{figure}
```



(a) A single trap (b) One of two traps created by an AOD

FIGURE 1.1 – Exemple de deux figures.

1.3.5 En-têtes et pieds de pages

Le package “fancyhdr” permet de choisir comment seront les en-têtes et les pieds de page. Il faut définir le style pour les pages qui sera utilisé pour la majorité des pages du document (et qui s’appelle “fancy”) :

```
\pagestyle{fancy}
\fancyhf{} % assure que les entête et pieds de page sont vides au départ
\fancyhead[LE]{\leftmark}
\fancyhead[RO]{\rightmark}
\fancyfoot[LE,RO]{\thepage}
```

Les commandes par défaut sont les suivantes :

- L = left, R = right, C = center, E = even pages, O = odd pages
- \thepage : adds number of the current page.
- \thechapter : adds number of the current chapter.
- \thesection : adds number of the current section.
- \chaptername : adds the word "Chapter" in English or its equivalent in the current language.

- \leftmark : adds name and number of the current top-level structure (for example, Chapter for reports and books classes ; Section for articles) in uppercase letters.
- \rightmark : adds name and number of the current next to top-level structure (Section for reports and books ; Subsection for articles) in uppercase letters.

Ici on a donc demandé que : sur les pages paires en haut à gauche, on ait le nom du chapitre en cours, sur les pages impaires en haut à droite on ait le nom du paragraphe en cours, et en bas de toutes les pages on a le numéro de la page. On remarquera que par défaut les noms des chapitres, sections et paragraphes sont inscrits en majuscules dans les en-têtes, ce que je trouve personnellement assez moche. Pour y remédier, on peut redéfinir les commandes \leftmark et \rightmark (code à insérer avant la commande \fancyhf) :

```
\renewcommand{\chaptermark}[1]{\markboth{\chaptername \ \thechapter.\ #1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection.\ #1}}
```

Il faut ensuite définir le style des pages “spéciales” comme celles de début de chapitre (qui s'appellera “plain”) :

```
\fancypagestyle{plain}{
\fancyhf{}
\fancyfoot[RO,RE]{\thepage}
\renewcommand{\headrulewidth}{0pt}
\renewcommand{\footrulewidth}{0pt}}
```

En particulier, il est spécifié que pour ces pages, la barre horizontale qui sépare l'en-tête du corps du texte soit d'épaisseur nulle.

1.3.6 Citation en début de chapitre

Que serait une thèse sans une citation en début de chapitre ?

Le package “epigraph” permet d'ajouter des citations avec la syntaxe suivante :

```
\epigraph{\textit{Texte de la citation.}}{Auteur}
```

Si le résultat ne vous satisfait pas, vous pouvez également utiliser le package “quotchap” avec la syntaxe suivante :

```
\begin{savequote}
Texte de la citation.
\qauthor{Auteur}
\end{savequote}
```

1.3.7 Intégration de code

Merci à Michaël Rollin pour cet ajout !

Il est parfois utile de pouvoir citer du code de façon lisible dans sa thèse.

Le package “listings” permet d'intégrer des blocs de codes avec une personnalisation de la colorisation syntaxique. Pour en faire appel :

```
\begin{lstlisting}
Ton code ici
\end{lstlisting}
```

Par défaut voici un exemple de la syntaxe donnée avec du python :

```python
# Création du circuit quantique
q = QuantumRegister(nb_qubits)
c = ClassicalRegister(nb_qubits)

qc = QuantumCircuit(q, c)

# Pairing aléatoire
i = 0
while i < nb_qubits:
    cible_qbit = random.choice(random_circuit)
    qc.rx(math.pi / 2, q[cible_qbit])
    random_circuit.remove(cible_qbit)
    if len(random_circuit) != 0:
        target_qbit = random.choice(random_circuit)
        qc.cx(q[cible_qbit], q[target_qbit])
        random_circuit.remove(target_qbit)
    i += 2

qc.measure(range(nb_qubits), range(nb_qubits))
```

1.3.8 Automatisation / Compilation avec GitHub

Merci à Michaël Rollin pour cet ajout !

Cela peut être une bonne idée d'utiliser logiciel de gestion de versions tel que git lors de la rédaction de sa thèse. Ce dernier permet de conserver un historique des modifications, et de revenir facilement en arrière en cas de besoin. En le couplant avec un service d'hébergement en ligne, tel que GitHub ou GitLab, on s'assure d'avoir toujours une copie de sauvegarde en ligne en cas de problème (pensez à sauvegarder régulièrement vos fichiers de thèse !).

Dans le cas où l'on utilise un tel dispositif[2], on peut vouloir automatiser la compilation de ses fichiers latex en PDF pour ne pas avoir à l'effectuer manuellement à chaque modification apportée. C'est ce que nous vous proposons ici dans un template qui utilise les "GitHubAction". En bref : à chaque fois que vous effectuerez un "push" vers le serveur distant, un script sera exécuté, qui compilera les fichiers PDF et les ajoutera directement au serveur via un push automatisé (les fichiers PDF seront donc récupérables directement sur le serveur, ou lorsque vous effectuerez un "pull" sur votre machine[3]).

2. Le but de ce chapitre n'est pas de vous apprendre à utiliser git et/ou github, si cela vous intéresse, il existe de nombreux tutoriels en ligne, ainsi que des formations dispensées directement au sein des laboratoires, n'hésitez pas à demander à votre service informatique s'ils peuvent vous aiguiller sur le sujet.

3. La compilation automatique peut prendre quelques minutes, ne vous inquiétez pas si vous ne voyez pas les fichiers apparaître instantanément.

Pour l'utiliser dans votre repository :

- Copiez collez le dossier *.github* à la racine de votre repo (comme c'est le cas dans l'exemple ici).
- Modifiez le fichier *.github/workflows/compile.yml*, pour le faire correspondre à vos besoin.

Ci dessous, une brève description des fonctions contenues dans le fichier *compile.yml*. La variable *location_latex* sert à indiquer l'endroit où sont stockées vos sources .tex (ici, elles sont dans le dossier *latex_files*) :

```
location_latex: "latex_files"
```

Ensuite, les packages nécessaires à l'installation de l'environnement latex sont déclarés, si des packages spécifiques sont nécessaires pour compiler votre document, ajoutez-les à cette liste (ils ne seront pas installés sur votre ordinateur, ils serviront juste à l'exécution du script en ligne) :

```
- name: Install latex
  run: sudo apt-get install -yq texlive texlive-latex-extra texlive-lang-
    all
- name: Install extra package
  run: sudo apt-get install -yq texlive-science
```

Pour chaque fichier pour lequel vous souhaitez générer un PDF, il vous suffit de créer un bloc comme celui ci-dessous :

```
- name: Compile Chapitre 1
  uses: ./.github/actions/compile
  with:
    document_tex: "compilation_chapitre1.tex"
    location_file: ${{ env.location_latex }}
```

Ce bloc permet de copier les fichiers générés finaux pour les rendre accessibles dans le dossier de votre choix (ici on copie et renomme le chapitre 1 et le fichier complet dans le sous-dossier *example* du repository) :

```
- name: Move examples
  run: |
    cp ${{ env.location_latex }}/compilation_chapitre1.pdf examples/
    explanations_packages.pdf
    cp ${{ env.location_latex }}/
    compilation_these_avec_page_de_garde_et_abstract.pdf examples/
    complete_thesis.pdf
```

Enfin, ce bloc indique au script d'effectuer un push sur le serveur avec les nouveaux fichiers générés :

```
- name: Pull modif
```

```
2     run: git pull
3   - uses: stefanzweifel/git-auto-commit-action@v4
4     with:
5       commit_message: Compile latex
6       file_pattern: ${{ env.location_latex }}/*.pdf examples
7       push_options: '--force'
```

Si le document latex comporte des erreurs, la compilation automatique échouera, on peut alors avoir accès aux logs sur le serveur GitHub (pour trouver ce qui a causé l'erreur de compilation).

Remarque : Si vous ne souhaitez pas utiliser cette compilation automatique, vous pouvez tout à fait faire sans (par exemple en supprimant le sous-dossier *.github* après avoir téléchargé ou cloné ce repository).

Chapitre 2

Un autre chapitre pour l'exemple

Blablabla.

Chapitre 3

Un troisième chapitre

Blablabla.

Chapitre 4

Chapitre 4

Blablabla.

Chapitre 5

Dernier chapitre

Blablabla.

Conclusion

Et voilà !

Bibliographie

[1] D. Flipo, “Documentation sur le module babel-french.” http://daniel.flipo.free.fr/frenchb/frenchb-doc.pdf.

[2] J. Wright, “siunitx - a comprehensive (si) units package.” ftp://ftp.dante.de/ctan%3A/macros/latex/exptl/siunitx/siunitx.pdf.

[3] U. Lindgren, “Fncychap v1.34.” http://ctan.mines-albi.fr/macros/latex/contrib/fncychap/fncychap.pdf.