Rudiments de LaTeX

Jérémy Dousselin

17 juillet 2024

Une courte introduction aux essentiels de LaTeX.

Table des matières

1 Introduction

LaTeXpermet à la fois le traitement de texte, mais aussi la mise en forme des mathématiques. Par défaut, LaTeXconsidère que l'on fait du traitement de texte, et pour écrire des maths, il faut déclarer à LaTeXqu'on va écrire des maths (voir la section 4). Un autre des intérêts majeurs de LaTeX, c'est la possibilité de créer des "commandes" et des "environnements", qui définissent un style précis d'objet, réutilisable tout au long du document. Par exemple, dans un papier de mathématiques, de nombreux théorèmes sont présents. Leur mise en page, autant que leur numérotation sont automatiques. L'auteur ne va pas refaire tout le travail à chaque fois ! Deux avantages évidents à cela : premièrement, si l'on souhaite changer le style de tous nos théorèmes (par exemple, on veut les encadrer en rouge), il suffit de modifier la définition générale de l'environnement LaTeXde votre théorème, pas besoin de tous les modifier un à un. Deuxièmement, la numérotation automatique permet de changer tout le document, automatiquement, si l'on ajoute par exemple un nouveau théorème au tout début de l'article (décalant donc tous les suivants de un). Encore une fois, on ne modifie jamais ce genre de chose à la main, un à un, c'est trop laborieux, et c'est ne pas comprendre l'intérêt de LaTeX.

1.1 Interface

Ce document a pour but de vous transmettre les bases de la rédaction d'un document en LaTeX. Nous y traiterons donc de connaissances directement liées à LaTeX, mais nous allons commencer par traiter du logiciel LaTeXet de son fonctionnement.

Il existe plusieurs logiciels pour taper du LaTeX. Nous nous baserons ici sur une installation Texmaker, couplée au logiciel MikTex. L'interface ressemble à ceci.

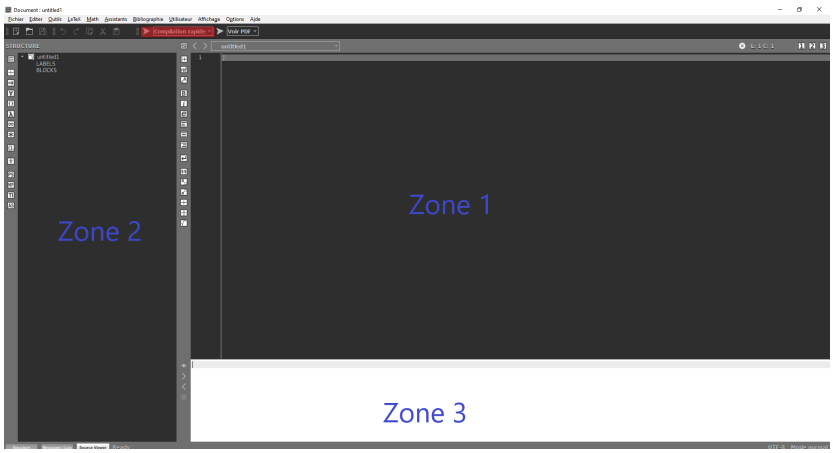

FIGURE 1 – Exemple d'interface Texmaker

Il est possible que, pour vous, l'interface soit blanche et qu'une quatrième zone s'affiche à droite. Pour passer en thème sombre, il vous suffit d'aller dans

Options→Configurer Texmaker→Éditeur→Thème Sombre.

La quatrième zone, nous y venons. Pour écrire en LaTeX, il vous faut écrire du texte dans la zone 1, avec les contraintes évoquées dans la section 2. Ce texte rédigé, il nous faut alors appuyer sur le ▶, surligné en rouge dans la figure 1, à gauche du menu déroulant "Compilation rapide". Ceci aura pour effet de compiler le texte et le code écrit dans la zone 1. Des messages peuvent alors apparaître en zone 3 : c'est la zone dans laquelle Texmaker nous dit où sont les erreurs, et en quoi ces erreurs consistent, ou simplement où Texmaker suggère des modifications (par exemple des packages non mis à jour, etc.). Une erreur, en rouge, empêchera la compilation du document. Une suggestion n'empêchera pas la compilation, mais il est préférable de ne pas en avoir. Notons que dans tous les cas, la compilation enregistre le document.

Dans le cas où la compilation réussit, c'est le PDF issu de la compilation qui s'affiche en zone 4. Parfois, comme dans mon cas, il est préférable de "détacher" cette zone 4 de l'interface principale de Texmaker. C'est par exemple utile lorsque vous travaillez sur deux écrans, un pour le code et un pour visualiser le résultat. Pour faire cette séparation, il vous faut décocher la case

Options→Configurer Texmaker→Commandes→Intégré à la fenêtre.

Notons par ailleurs qu'un clique droit sur ce PDF propose l'option "cliquer pour aller à la ligne correspondante". Ceci a pour effet de vous emmener directement à l'endroit du code LaTeXqui a servi à compiler l'endroit de votre clique droit.

La zone 2 est simplement le récapitulatif des différentes strates de votre document. La stratification en section/sous-sections/etc. est détaillée dans la sous-section 2.4.

1.2 Propreté d'une installation

Tout ceci étant dit, évoquons un instant le propreté de la gestion d'un projet. Vous l'avez remarqué, une compilation LaTeXgénère de nombreux documents annexes. Ainsi, je vous conseille la règle très simple de **un projet LaTeX= un dossier**. N'allez pas mélanger plusieurs projets dans un seul, il est ensuite très pénible de s'y retrouver. Pour améliorer la lisibilité d'un projet, on peut encore le segmenter un peu plus. Ma manière de procéder est la suivante.

1. Je crée un dossier qui contiendra tout ce qui est relatif à mon projet, disons "Projet".
2. Dans ce dossier, je crée un dossier qui sera le dossier dans lequel je mets le fichier LaTeX, disons "Projet.tex",ainsi que ma bibliographie, disons "biblio.bib" (voir la sous-section 2.5). J'appelle ce dossier disons "Compil-Projet".
3. Il est courant qu'un projet nécessite d'intégrer diverses images ou graphes (voir sous-section 3.2) : je crée donc un dossier, dans "Projet", appelé "Images". Il contiendra toutes les images de mon projet.
4. Dans la dossier "Projet", j'ajoute enfin mon fichier de commandes, disons "commandes.tex" (voir sous-section 5.2).

Ceci permet d'avoir une structure de projet propre et lisible, et je vous conseille vivement de suivre ce modèle pour ne pas vous perdre dans un méli-mélo de fichiers annexes.

2 Structure d'un document

2.1 Préambule d'un document

Un document commence par la déclaration de la classe du document. En général, ce sera

```
\documentclass[12pt]{article}
```

Ici, ce qu'il y a entre accolades désigne la nature du document (livre, article, mémoire, etc.). Vous utiliserez quasiment toujours "article". De plus, entre crochets, on peut remplir certaines options. Ici, j'ai demandé à ce que le texte soit en taille 12. Vous pouvez faire varier la taille de la police entre 10, 11 et 12pt.

Une fois la déclaration de classe effectuée, il vous faut déclarer l'environnement dans lequel vous allez taper votre document. Ceci se fait de la manière suivante :

```
\begin{document}

\end{document}
```

Le texte que l'on veut compiler doit alors se trouver à l'intérieur de cet environnement. Le document ainsi crée est un document théoriquement fonctionnel. Seulement, dans les faits, il faut encore inclure quelques suppléments pour avoir une vraie base de document LaTeX.

2.2 Les packages

Ces choses à ajouter, ce sont les packages. Ces packages sont des fichiers contenant du code nécessaire à Texmaker pour interpréter certains symboles, comme les accents par exemples. Des packages plus poussés permettent de faire des dessins, de rajouter des couleurs au document, etc. Lorsque l'on crée un document, il est préférable de n'ajouter que les packages dont nous avons besoin pour le document, mais ce n'est pas bien grave d'en avoir un ou deux en trop.

Ces packages sont à inclure **avant** le `\begin{document}`, dans une commande du type

```
\usepackage[option]{nom_du_package}
```

Ici, les crochets sont à mettre seulement si on souhaite ajouter une option au package. On peut soit les inclure directement dans notre fichier .tex, soit les inclure dans un fichier de commande (voir sous-section 5.2). Je préfère cette seconde option. Voici quelques packages utiles :

- ⋆ `fontenc` avec option `T1` : sert à gérer les accents, entre autres.
- ⋆ `babel` (avec option `francais`) : sert essentiellement à gérer le texte français.
- ⋆ `amsmath, amssymb` : servent à gérer beaucoup de symboles mathématiques.

- ⋆ `hyperref` : sert à mettre des hyperliens dans le document PDF.
- ⋆ `geometry` (avec l'option `[left=3cm,right=3cm,top=2cm,bottom=2cm]` par exemple) : sert à gérer la taille des marges du document (on peut remplacer les chiffres par ce que l'on veut, évidemment).

Il m'apparaît à peu près essentiel de mettre ces packages dans vos documents mathématiques. D'autres les accompagneront (vous pouvez déjà voir mon fichier .tex qui en contient plus), mais ceux là sont presque inévitables.

2.3 Page de garde et sommaire

LaTeXdispose d'une commande toute faite pour créer une page de garde ainsi qu'une table des matières. Premièrement, pour faire une page de garde, vous pouvez remplir les commandes

```
\author{Votre_nom}
\title{Le_titre}
```

Ces commandes peuvent être remplies avant le début du document. La commande `\date` est optionnelle : si elle n'est pas remplie, alors la date du jour sera la date prise en compte par LaTeX. Une fois ces informations rentrées par l'utilisateur, il peut appeler la commande

```
\maketitle
```

à l'endroit où il souhaite faire apparaître sa page de garde. Bien sûr, libre à vous de personnaliser cette page de garde comme bon vous semble, la commande `title` a juste le mérite de le faire directement, mais n'est pas forcément à votre goût.

Pour la table des matières, LaTeXcollecte tout seul toutes les entrées de sommaire (les chapitres, sections, etc.). Vous avez simplement à appeler la commande

```
\tableofcontents
```

là où vous souhaitez appeler votre table des matières. Attention, la première compilation va dire à LaTeXque vous souhaiter faire une table des matière. Elle apparaîtra donc, mais vide. Une seconde compilation la remplira. Pour changer le nom de la table des matières, on peut utiliser la commande

```
\renewcommand{\contentsname}{nom_à_choisir}.
```

2.4 Les sections

Pour un article, il n'existe que 3 types de strates dans un document : les sections, les sous-sections et les sous-sous-sections. Elles se déclarent respectivement par une commande du type

```
\section{nom_section}
\subsection{nom_sous_section}
\subsubsection{nom_sous_sous_section}
```

Ces strates sont alors numérotées : d'abord le numéro de la section en cours, puis le numéro de la sous-section en cours, et enfin le numéro de le sous-sous-section en cours. Si, pour une raison ou une autre, nous souhaitons ne pas numéroter une section, alors nous pouvons simplement luis ajouter une étoile, comme suit :

```
\section*{nom_sec}
```

Ceci aura aussi pour effet de la faire disparaître de la table des matières. Si on veut tout de même qu'elle apparaisse sur la table des matières, sans être numérotée, alors il faut écrire

```
\section*{nom_sec_sans_num}
\addcontentsline{toc}{section}{nom_à_mettre_dans_le_sommaire}
```

2.5 La bibliographie

Pour commencer, il faut créer une bibliographie. Créez un nouveau fichier LaTeX, enregistrez-le sous le nom "mabiblio.bib", par exemple, dans le dossier de votre fichier .tex. Le .bib est important : il permet de dire à Texmaker de ne pas enregistrer ce document comme un fichier .tex, mais comme un fichier bibliographique.

Ce fichier crée, notez qu'il ne faut pas le compiler : on l'enregistre uniquement, par exemple avec le symbole de disquette. Les données à entrer à l'intérieur sont les entrées bibliographiques. Il est rare de devoir les écrire soi-même, et nous les récupérons sur à peu près tous les sites de partages d'articles scientifiques. Par exemple, ici sur Arxiv :

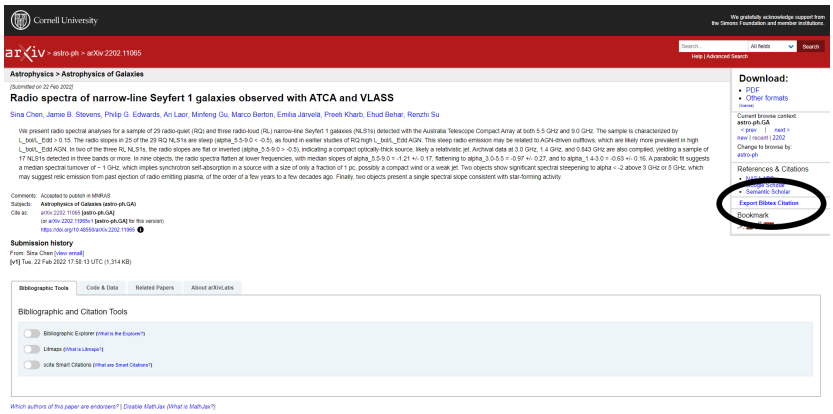

FIGURE 2 – Exporter une donnée bibliographique sur Arxiv

Ces données, une fois récupérées, ressemblent à quelque chose comme ceci

```
@article{2018,
   title={Precise Time Delays from Strongly Gravitationally Lensed
Type Ia Supernovae with Chromatically Microlensed Images},
   volume={855},
   ISSN={1538-4357},
   url={http://dx.doi.org/10.3847/1538-4357/aaa975},
   DOI={10.3847/1538-4357/aaa975},
   number={1},
   journal={The Astrophysical Journal},
   publisher={American Astronomical Society},
   author={Goldstein, Daniel A. and Nugent, Peter E. and Kasen,
Daniel N. and Collett, Thomas E.},
   year={2018},
   month={Mar},
   pages={22} }
```

La première des entrées, 2018, est la clé : c'est ce qui va permettre de citer cette référence. On peut la remplacer par ce que l'on veut. Ensuite, toutes les entrées ne nous intéressent pas, comme l'ISSN, l'URL, le DOI et le mois. On peut donc simplement modifier notre référence en

```
@article{article_phy_test,
   title={Precise Time Delays from Strongly Gravitationally Lensed
Type Ia Supernovae with Chromatically Microlensed Images},
   volume={855},
   number={1},
   journal={The Astrophysical Journal},
   publisher={American Astronomical Society},
   author={Goldstein, Daniel A. and Nugent, Peter E. and Kasen,
Daniel N. and Collett, Thomas E.},
   year={2018},
   pages={22} }
```

Une fois le fichier mabiblio.bib rempli des entrées qui nous intéressent, nous pouvons les citer dans notre document, avec la commande

```
\cite{clé}
```

Par exemple, ici nous pouvons citer l'article dont les entrées bibliographiques sont données précédemment (voir [1]). Nous ne sommes cependant pas obligé de citer toutes les entrées de notre bibliographie.

Enfin, pour faire apparaître notre bibliographie dans notre document, il faut faire appel aux deux commandes suivantes :

```
\bibliographystyle{abbrv}
\bibliography{mabiblio}
```

La première permet de choisir le style de bibliographie que nous voulons. C'est esthétique, mais vous devez choisir un style, sans quoi le fichier ne compilera pas. La seconde ligne permet simplement à LaTeXde savoir comment s'appelle votre bibliographie. Reste une dernière remarque avant de compiler. Sans rien écrire de plus, Texmaker ne fera apparaître que la bibliographie des articles que vous citez via la commande `\cite`. Si vous n'en citez aucun, alors il y aura une erreur. La méthode pour pallier à ceci, si vous voulez que l'intégralité des données de votre bibliographie apparaisse, que vous les citiez ou non, alors il vous faudra ajouter la commande

```
\nocite{*}
```

disons juste avant la commande fixant le style de votre bibliographie.

Ceci étant fait, on compile le document normalement. Ensuite, on change de "compilation rapide" à "Bibtex", et on compile notre document .tex. On revient ensuite en "compilation rapide" et on compile deux fois. La bibliographie devrait apparaître !

3 Le traitement de texte

3.1 Les généralités

Lorsque l'on saisi du texte dans Texmaker, il n'y a rien de particulier à signaler si ce n'est pour les saut de ligne. Un unique retour à la ligne dans le document .tex revient juste à un simple espace. Par contre, un double retour à la ligne dans le .tex a pour effet de créer un nouveau paragraphe.

On observe donc un incrément, comme ce qui vient de se passer pour cette phrase. Si, parfois, on ne souhaite pas voir d'incrément, il suffit d'appeler la commande

```
\noindent
```

Une petite note sur ce qui permet de commenter du texte : dans Texmaker, on peut commenter son code grâce au symbole %.

3.2 Inclusion d'images

Pour travailler avec des images (en format jpeg, png, pdf, etc.), il existe un environnement `figure`, **qui fonctionne très très mal**. Il nécessite le package `graphicx`. Je vous explique cependant comment l'utiliser, mais des erreurs de position d'image interviendront.

Comme je l'ai détaillé plus haut, lors de la création propre d'un projet LaTeX, les images ne sont pas forcément dans le même dossier que le fichier .tex. Pour appeler une image ailleurs que dans le dossier du .tex, il faut se munir de son chemin dans

l'ordinateur, sous la forme par exemple [1] :

```
C:/Users/dousseli1/Desktop/nom_image
```

Ensuite, pour appeler cette image, on écrit

```
\begin{figure}[!h]%l'option !h permet de gérer un bon placement
%de l'image
\centering %pour centrer l'image et la description
\includegraphics[scale=1]{{C:/Users/dousseli1/Desktop/image}} %pour
%appeler l'image, avec une échelle (=scale) valant 1
\caption{mon_titre} %pour donner un nom à la figure
\end{figure}
```

Évidemment, vous n'avez pas à recopier le texte commenté. Au lieu de donner la taille de l'image en terme d'échelle, on peut aussi lui donner l'option

```
\includegraphics[width=x\textwidth]{C:/Users/dousseli1/Desktop/image}
```

Ici, `x` représente simplement un coefficient dilatant l'image. On peut par exemple remplacer `x` par 0.7. Cette commande ne se soucie donc plus de l'échelle, mais plutôt de la largeur de l'image, qui doit donc valoir `x` fois la taille d'une ligne de texte.

Plutôt que de préciser à chaque fois le chemin vers l'image, si nous avons déjà un dossier tout prêt où se trouvent toutes nos images, alors on peut directement dire à Texmaker l'endroit où toutes nos images se trouvent. Pour cela, on écrit au début de notre document

```
\graphicspath{{C:/Users/dousseli1/Desktop/}}
```

Notez le fait qu'il y ait des doubles accolades. Après cette commande, il suffit simplement d'entrer le nom de l'image dans la commande `includegraphics`. On trouve donc une commande comme par exemple

```
\includegraphics[width=0.7\textwidth]{image}
```

Dernière note concernant les chemins : supposons que nous avons un dossier D contenant un dossier "Document", qui contient le .tex, et un dossier "Images". Alors le dossier "Images" est juste 'un dossier en arrière' par rapport à notre .tex. Texmaker accepte alors des commandes du genre

```
\graphicspath{{../Images/}}
```

1. On peut aussi utiliser le code suivant sans l'environnement **figure**, sans la commande **centering** et sans la commande **caption**. La figure ne sera alors pas référencée comme figure, il s'agira juste d'une image, que l'on peut toujours centrer ou légender. C'est la manière facile d'éviter l'environnement `figure`.

Il l'interprète comme 'on va dans le dossier parent de celui contenant le .tex, et on redescends vers le dossier "Images"'.

Le problème, si l'on évince `figure`, c'est que l'on ne peut plus faire référence aux images : et oui, car il n'y a plus d'environnement d'images, elles sont "volantes" dans le document. Vous trouverez à suivre un code pour un environnement d'inclusion de figures de ma composition, qui lui fonctionne.

```
\newcounter{fig0}
\makeatletter
\newenvironment{fig}[1]{
 \refstepcounter{fig0}
 \protected@edef\@currentlabelname{figure \arabic{fig0}}
 \newcommand{\copie}{#1}
 \begin{center}
}
{
 \\
 Figure \arabic{fig0} : \copie
 \end{center}
}
\makeatother
```

Un bon exercice serait de comprendre son fonctionnement, chose possible après la lecture de la section 5.

3.3 Quelques commandes/environnements utiles

- ⋆ `\hfill, \vfill` : la première pousse le texte tout à droite, la seconde tout en bas. Elles remplissent l'espace horizontalement/verticalement, d'où leur nom.
- ⋆ `\newpage` : saute une page
- ⋆ `\newline, \\` : saute une ligne, sans pour autant créer un nouveau paragraphe.
- ⋆ `\par` : crée un nouveau paragraphe.
- ⋆ `\textbf{ }` : met le texte dans la commande en gras.
- ⋆ `\begin{center}` : centre ce qui est dans l'environnement.
- ⋆ `\underline{ }` : souligne ce qui est donné en argument.
- ⋆ `\textit{ }` : met ce qui est dans la commande en italique.

Notez que certains symboles en LaTeXn'apparaissent pas simplement en les écrivant dans la zone de saisie de texte. C'est le cas des symboles suivants :

$$\#, \{, \}, \$, \%, \_, \backslash.$$

Pour les faire apparaître, il vous faut les protéger avec un \, en écrivant par exemple `\{`. Seul le \ n'apparaît pas comme ça : il faut utiliser la commande

```
\backslash
```

4 Le mode maths

4.1 Le mode maths

Lorsque l'on veut taper des maths, il faut intégrer notre texte dans le mode maths. Le mode maths se distingue du mode texte, le mode normal, car il s'ouvre et se ferme à l'aide du symbole $. Ainsi `$x$` produira un "x" en mode maths (x), alors que `x` produira un "x" de texte (x).

La police n'est pas la seule différence. Certaines commandes ne peuvent compiler que dans un mode maths, comme par exemple `\mathbb{R}`. Cette commande, si elle n'est pas encadrée par une paire de $, causera une erreur de compilation. Le mode maths entre $ est réservé pour un usage courant, lorsque l'on écrit des maths dans une ligne de texte normal. Notons que le mode maths contient autant de caractères qu'on le veut. Ainsi, on écrire

```
$\forall x\in\mathbb{R}$
```

plutôt que

```
$\forall$ $x$ $\in$ $\mathbb{R}$
```

On peut aussi rentrer en mode maths via divers environnement détaillés en sous-section 4.3, ainsi qu'avec le syntaxe

```
\[mes_maths\]
```

Dans ce cas, pas besoin de $: la présence des `\[` assure l'entrée et la sortie du mode maths. Ce mode maths-ci a pour effet de centrer ainsi que de changer légèrement la présentation des maths tapées dedans, en moins condensées. Par exemple, la somme $\sum_{k=0}^{n} a_k$ est tapée entre $, alors que cette même somme tapée entre crochets donne

$$\sum_{k=0}^{n} a_k.$$

Dans tous les cas, il est bon de noter que le mode maths ne fait pas apparaitre les espaces et met le texte en italique. On évite donc de mettre du texte dans un mode maths. Seulement, parfois, on ne peut pas faire autrement. On dispose donc de la commande

```
\text{mon_texte}
```

qui aura pour effet d'intégrer du texte dans un mode maths, espaces compris, et sans italique. Pour forcer un espace en mode maths, on peut utiliser les choses suivantes :

- ⋆ `~` : espace insécable. Donne $a\ b$.
- ⋆ `\,` : petit espace. Donne $a\,b$.
- ⋆ `\;` : plus grand espace. Donne $a\;b$.

⋆ `\quad` : encore un plus grand espace. Donne $a \quad b$.

⋆ `\qquad` : le plus grand des espaces ici donnés. Donne $a \qquad b$.

Pour finir, j'attire votre attention sur le fait qu'un espace en mode maths, bien que non affiché par LaTeX, est parfois nécessaire. C'est notamment le cas pour bien séparer le nom des commandes. Ainsi

```
a\qquadb
```

donnerait une erreur, car `\qquadb` n'est pas une commande existante. Il faut écrire

```
\qquad b
```

4.2 Les environnements de théorème

Dans la suite, le package `amsthm` sera utile. On peut créer des environnements avec la commande

```
\newtheorem{nom_environnement}{nom_affiché}
```

Cette commande écrite, on peut désormais utiliser un nouvel environnement, avec la syntaxe

```
\begin{nom_environnement}[option]

\end{nom_environnement}
```

Sur le PDF, on verra alors un paragraphe appelé 'nom_affiché' apparaître. Par exemple, avec 'Théorème' pour 'nom_affiché', on obtient

Théorème 4.2.1 (Infinité des nombres premiers)**.** *Il existe une infinité de nombres premiers.*

Le titre du théorème : c'est ce que l'on met dans l'option. On est donc pas obligé de mettre une option.

Le package intègre aussi un environnement de preuve, déjà tout fait, sous le nom `proof`. On peut donc écrire

Démonstration. Par l'absurde, s'il y en a un nombre fini, alors on considère leur produit $+1$ et le résultat suit assez clairement. □

On observe une numérotation sur le théorème, mais pas sur la preuve. Pour avoir un environnement de théorème sans numérotation, il faut utiliser la commande

```
\newtheorem*
```

Enfin, on peut souhaiter une numérotation particulière des théorèmes. Par exemple, on peut souhaiter une numérotation en fonction de la section/sous-section/sous-sous-section dans laquelle se trouve ledit théorème. Pour cela, il faut le spécifier dans la création de l'environnement, comme suit

```
\newtheorem{theorem}{Théorème}[section]
```

Dans notre exemple, la numérotation est faire selon la sous-section (c'est bien le premier théorème de la seconde sous-section de la quatrième section).

4.3 Esthétisme des équations

Il existe deux environnements deux autres environnements qui permettent une "jolie" mise en page des équations. Le premier, c'est l'environnement

```
\begin{equation}
```

Cet environnement va numéroter les équations et les centrer, comme le mode maths `\[ \]` (à la numérotation près). L'environnement est déjà en mode maths.

Le second environnement, le plus utile à mon avis, c'est

```
\begin{align}
```

Cet environnement permet de centrer et surtout aligner les équations. Pour fixer "une ancre", dire à Texmaker l'endroit d'alignement, on utilise une esperluette &. Bien penser à sauter une ligne à chaque fois que l'on veut sauter une ligne dans l'environnement d'alignement ! Par exemple, le code

```
\begin{align}
f(x)&=g(x)
\\ &=h(x)
\\ &=x^2+1.
\end{align}
```

donnerait

$$\begin{aligned} f(x) &= g(x) && (1)\\ &= h(x) && (2)\\ &= x^2+1. && (3) \end{aligned}$$

Bien sûr, on peut éviter la numérotation en usant des versions étoilées de ces environnements, à savoir

```
\begin{align*}
```

et

```
\begin{equation*}
```

Petite remarque concernant la taille des parenthèses. Dans certaines expressions, les parenthèses encadrent des symboles "grands". Il est disgracieux de ne pas adapter la taille des parenthèses à leur contenu. Ainsi, on évitera des choses comme

$$(\int_0^1 f(t)dt)^2.$$

Au lieu de simplement mettre des parenthèses autour de notre expression, on peut encadrer notre expression comme suit

```
\left(expression\right)
```

La présence des `\left` et `\right` dit à Texmaker de faire apparaître des parenthèses dont la taille est adaptée à l'expression qu'elles contiennent, ce qui donne dans notre exemple

$$\left(\int_0^1 f(t)dt\right)^2.$$

Ces commandes ne vont pas l'une sans l'autre ! Si on veut mettre un symbole de parenthèsage -adapté à la taille d'une expression- à gauche, mais rien à droite, il vous faudra combler l'absence d'un `\right` par la commande `\right.` (notez bien le point). Cette commande délimite pour LaTeXl'expression à parenthèser, sans rien afficher à droite. Par exemple,

$$f(x) = \begin{cases} x & \text{si } x \geq 0, \\ 0 & \text{sinon.} \end{cases}$$

Ici, c'est l'accolade qui est adaptée à la taille de l'expression, et pourtant rien est affiché à droite.

4.4 Quelques commandes/environnements utiles

- ⋆ `_{ }` : permet de mettre en indice, par exemple a_i.
- ⋆ `^{ }` : permet de mettre en exposant, par exemple a^i.
- ⋆ `\sum` : affiche le symbole somme Σ.
- ⋆ `\prod` : affiche le produit $\prod$.
- ⋆ `\int` : affiche l'intégrale $\int$.
- ⋆ `\circ` : affiche le rond de la composition $f \circ g$.
- ⋆ `\times` : affiche un "joli" produit $\times$.
- ⋆ `\forall` : affiche le quel que soit $\forall$.
- ⋆ `\exists` : affiche le symbole d'existence $\exists$.
- ⋆ `\mathcal{ }` : donne la version cursive de certaines lettres majuscules, comme $\mathcal{A}$.

- ⋆ `\mathbb{ }` : donne la version "doublée" de certaines lettres majuscules, comme $\mathbb{A}$.
- ⋆ `\sigma` : une lettre grecque avec initial minuscule produit la minuscule grecque, ici σ. Il en va donc de même avec λ, θ, ρ, etc.
- ⋆ `\Sigma` : comme avant, mais en majuscule, par exemple Σ, Θ, Λ, etc. Ne fonctionne pas pour toutes les lettres grecques, notamment celles dont la majuscule est une lettre latine, comme α ou β, dont les majuscules sont A et B.
- ⋆ `\implies` : produit une implication $\Longrightarrow$.
- ⋆ `\in` : produit le symbole d'appartenance $\in$.
- ⋆ `\equiv` : produit le symbole de congruence $\equiv$.
- ⋆ `\iff` : produit le symbole d'équivalence $\Longleftrightarrow$.
- ⋆ `\subset` : produit le symbole d'inclusion $\subset$.
- ⋆ `\not` : nie le symbole qui lui succède, par exemple $\notin$ ou $\not\subset$.
- ⋆ `\neq` : produit le symbole "différent de" $\neq$.
- ⋆ `\geq, \leq` : produisent les symboles de relation d'ordre $\geq, \leq$.
- ⋆ `\pm, \mp` : produisent les symboles $\pm, \mp$.
- ⋆ `\infty` : produit le symbole de l'infini ∞.
- ⋆ `\frac{ }{ }` : produit la fraction dont le numérateur est le premier paramètre et le dénominateur est le second, par exemple $\frac{a}{b}$.
- ⋆ `\exp, \log, \sin, \cos, \tan, \sqrt` : leur fonction est assez claire je pense.

5 Commandes et environnement

En toute généralité, je vous invite à utiliser et étudier les environnements crées dans le document d'exemple.

5.1 Les compteurs

Vous l'avez vu, LaTeXdispose de nombreux environnements liés à une numérotation. Pour garder en mémoire cette numérotation, il fait appel à des compteurs. Nous pouvons nous aussi créer des compteurs. La commande à suivre est

```
\newcounter{nom_compteur}[autre_compteur]
```

L'option autre_compteur permet au compteur nouvellement crée, ici nom_compteur, d'être réinitialisé à chaque fois que autre_compteur est incrémenté. On dispose alors des commandes suivantes sur les compteurs

- ⋆ `\setcounter{compteur}{nombre}` : fixe la valeur de compteur à nombre.

- ⋆ `\refstepcounter{compteur}` : incrémente la valeur de compteur de 1, tout en permettant à ce compteur de servir dans les références.
- ⋆ `\arabic{compteur},\roman{compteur},\Roman{compteur}, \alph{compteur},\Alph{compteur}` : ces commandes permettent, respectivement, de donner le compteur sous forme de chiffres arabes, chiffres romains minuscules, chiffres romains majuscules, lettres latine minuscules et lettres latine majuscules.

5.2 Créer une commande, un environnement

Dans la suite, nous allons créer des commandes et des environnements. Je vous conseille de disposer d'un document "commandes.tex", dans votre dossier de projet, dans lequel vous mettrez toutes les commandes et environnements crées. On y mettra les compteurs, les packages ainsi que toutes les commandes de modification du document (modification des interlignes, du système de référence, etc.). Ce document ne présente ni `\documentclass[]{}`, ni `\begin{document}`. Il ne faut pas non plus le compiler, juste l'enregistrer. Pour ensuite que notre .tex principal l'utilise, il nous suffit d'intégrer la commande

```
\input{chemin_vers_commandes.tex}
```

au tout début de notre document, entre le `\documentclass[]{}` et `\begin{document}`. Pour un point traitant des chemins en LaTeX, je vous renvoie à la sous-section 3.2.

Pour créer une nouvelle commande, on utilise

```
\newcommand{\nom_commande}{effet}
```

On peut par exemple créer une commande pour remplacer l'instruction `\mathbb{N}`. Pour cela, on écrit

```
\newcommand{\NN}{\mathbb{N}}
```

et désormais, l'écriture de `\NN` (en mode maths) donne $\mathbb{N}$. On peut aussi souhaiter que notre commande prenne en compte des options, un paramètre auquel elle s'applique. Pour cela, on ajoute une option comme suit

```
\newcommand{\nom_commande}[1]{effet_sur_#1}
```

La syntaxe ici déclare que votre commande prend un argument (non-optionnel), et que l'on appelle cet argument dans l'accolade "effet_sur" en écrivant #1. S'il y avait deux arguments[2], alors il aurait fallu écrire `[2]` au lieu de `[1]` dans la définition ci-dessus, et on ferait appel au premier argument via un #1, et au second via un #2. Pour signaler à LaTeXle caractère d'un argument (qui sera alors nécessairement le premier des arguments), on peut rajouter une paire de crochets comme suit :

2. Au plus, on peut signaler 9 arguments.

```
\newcommand{\test}[3][]{#2 : \textbf{#1,#3}}
```

Cette définition est à comprendre comme suit : elle prend trois arguments, dont le premier est optionnel. Ce que la commande retourne, c'est alors le deuxième argument, deux points, et le premier argument, le troisième argument, les deux en gras. Lors de l'appel d'une commande, les arguments se signalent entre accolades pour les obligatoires, et entre crochets pour l'éventuel optionnel. Dans le cas ci dessus, un exemple typique serait `\test[Galois]{Cauchy}{Hadamard}` Si jamais on ne souhaite rien mettre dans l'argument optionnel, alors pas besoin de mettre des crochets vides. Par contre, si on ne souhaite rien mettre dans un argument obligatoire (peu probable, mais pourquoi pas), il faut tout de même mettre une paire d'accolades vides. Par exemple `\test{}{Hadamard}` Ici les deux premiers arguments sont vides, et le troisième vaut "Hadamard".

En pratique, disons que l'on veut par exemple souligner et mettre en gras du texte. Si on appelle `\sg` un telle commande, on pourrait donc écrire

```
\newcommand{\sg}[1]{\textbf{\underline{#1}}}
```

Ceci produit donc, quand j'écris `\sg{bonjour}`, le texte suivant : **bonjour**. Pour changer l'effet d'une commande, on utilise la syntaxe

```
\renewcommand{\ancien_nom_commande}{nouvel_effet}
```

Comme vous avez pu le remarquer, Texmaker propose une complétion automatique du texte. Pour des noms de commandes/environnements longs qui ne sont pas proposés, vous pouvez les ajouter. Il vous suffit d'aller dans 'Utilisateur→Personnaliser complétion' pour ajouter vos nouvelles commandes dans la complétion automatique.

Pour créer un environnement (ce sont les blocs de la forme `\begin...\end`), le principe est exactement le même. On utilise la commande

```
\newenvironment{nom_environnement}[nombre_arguments]{debut}{fin}
```

Le fonctionnement des arguments est identique à celui exposé ci-dessus. Ce que vous entrez entre les accolades "début" sera l'action de l'environnement au début de l'environnement, alors que ce que vous entrez entre les accolades "fin" se produira à la fin de l'environnement. Tout ce que vous écrirez dans l'environnement ira entre les deux. Par exemple, on peut écrire

```
\newenvironment{prop}[1]{
 \refstepcounter{glob}

 \addvspace{0.3cm}\noindent
 \textbf{Proposition \Roman{section}.\alph{glob}} (#1) :\it
 }
 {
  \hfill$\star$

  \addvspace{0.3cm}
 }
```

L'environnement peut être compris comme suit. C'est un environnement dépendant d'un compteur se réinitialisant à chaque section. L'environnement commence par sauter une ligne[3] et afficher "Proposition" en gras souligné, suivi du numéro de la section en chiffre romain majuscule, d'un point, du numéro de la proposition en lettre minuscule, de l'option entre parenthèses et de deux points. Ensuite, on écrit ce que l'on souhaite. L'environnement se termine alors par l'affichage d'une étoile tout à droite de la ligne en cours, suivi d'un saut de ligne. Par exemple, on obtient :

Proposition V.a (Exemple) : *Voici une première proposition.* ⋆

ou encore

Proposition V.b (Re Exemple) : *Voici une seconde proposition.* ⋆

Attention une remarque concernant les environnements : l'argument optionnel ne peut pas être utilisé dans la seconde paire d'accolades, à la fin de l'environnement. On peut par exemple ruser en copiant le contenu des arguments optionnels dans la première paire d'accolades, dans une commande qui servira de variable de texte, et utiliser cette copie dans la seconde (c'est ce que l'on a fait en créant l'environnement `fig0` dans la sous-section 3.2).

6 Label et références

À chaque section/sous-section/etc., à l'environnement `figure`, aux environnement d'équation et aux environnements de théorème, on peut associer "un label". Un label est une sorte d'étiquette que l'on utilise pour faire appel, ailleurs dans le document, à l'objet labelisé. Par exemple, je peux ici faire référence à la figure 2. Pour associer un label, il suffit d'écrire

```
\label{nom_label}
```

dans l'environnement en question. Pour faire référence à une équation, on appelle la commande

```
\eqref{nom_label_equation}
```

Pour faire appel à autre chose, on utilise la commande

```
\ref{nom_label}
```

Pour faire apparaître les références à des labels, deux compilations sont nécessaires.

Pour terminer, on peut souhaiter changer la manière dont la référence apparait dans un document. Par exemple, vous pouvez souhaiter qu'une référence à une section ne

3. C'est l'enchainement du retour à la ligne et de `addvspace` qui permet ce saut de ligne (sans indentation grâce au `noindent`. Avantage de cette commande par rapport à `vspace` : les espaces verticaux ne sont ici pas cumulables, ce qui évite diverses situations peu élégantes.

face apparaître que son numéro, ou alors "section numéro_section", ou que sais-je. Pour formater ce qu'une référence affiche, on importe le package `fncylab`. Vous pouvez alors écrire la commande

```
\labelformat{figure}{figure #1}
```

Ici, ce sont les références aux environnement `figure` que j'ai formatées de sorte à faire apparaître "figure numéro_figure" au moment de la référence. Ce `\labelformat` doit être renseigné au tout début du document, après les packages par exemple.

7 Erreurs courantes

Parmi les erreurs fréquentes, on trouve

1. Répéter des mises en forme à la main, plusieurs fois au cours du document, sans utiliser d'environnement ou commandes pour automatiser la chose.
2. Ne pas ponctuer ses équations.
3. Oublier le mode maths sur certaines quantités dans le texte. Par exemple, quand je dis "Soit E un ensemble", E désigne une quantité mathématique, pas juste une lettre (ça voudrait dire quoi, E ?), elle se doit donc d'être en mode maths.
4. Faire des références "à la main", sans utiliser les notions de label et références.
5. Centrer des maths avec un environnement `center`, alors que `\[ \]` s'en charge très bien.
6. Des erreurs esthétiques, parmi lesquelles
   (a) trop d'espaces (il est disgracieux de forcer LaTeXà faire un espace vertical, en général. Quand vous changez d'idée dans le texte, vous entamez un nouveau paragraphe avec un double retour à la ligne, mais il est rare d'utiliser \\ au sein d'un paragraphe ;
   (b) faire des alignements verticaux interminables, on essaye de lisser un peu horizontalement. Il n'y a pas de règle précise, c'est plutôt une question de ce qui est beau, et donc subjectif : essayez au moins de trouver un équilibre ;
   (c) mettre des couleurs exubérantes à tout va ;
   (d) il est plus élégant de mettre les quantités mathématiques usuelles et "générales" en texte droit, pas italique. Par exemple, sin, c'est assez moche. On écrira sin. Si la quantité n'est pas déjà "droitisée" dans LaTeX, on utilise `\textrm{ }` pour le faire. Par exemple, on écrira $\mathrm{SO}_2(\mathbb{R})$ plutôt que $SO_2(\mathbb{R})$.

L'esthétisme est difficile à codifier, inspirez-vous des exemples que l'on vous donne !

Diverses aides et remarques

En règle général n'hésitez surtout pas à aller sur internet pour chercher solutions à vos problèmes en LaTeX, c'est ce que tout le monde fait. Plus généralement, des sites comme Overleaf (https://fr.overleaf.com/) possèdent beaucoup de tutoriels, en plus de vous permettre de compiler vos fichiers en ligne.

Par ailleurs, je vous conseille de bien lire les messages d'erreur qui apparaissent en bas de l'interface de Texmaker. En général, ils indiquent avec précision ce qui empêche la compilation.

Pour l'écriture de figure géométriques, je vous renvoie aux packages tikz et tkz-euclide, ainsi qu'à leur documentation.

Je vous conseille l'usage de l'application/site internet Detexify : elle permet, en traçant un symbole sur votre téléphone, d'obtenir le code LaTeXdes symboles ressemblant à votre tracé.

On évite de mettre des accents/espaces dans les noms de nos fichiers et dossiers. Par exemples, les accents empêchent Texmaker d'utiliser l'option "Cliquer pour aller à la ligne correspondante".

Références

[1] D. A. Goldstein, P. E. Nugent, D. N. Kasen, and T. E. Collett. Precise time delays from strongly gravitationally lensed type ia supernovae with chromatically microlensed images. *The Astrophysical Journal*, 855(1) :22, 2018.