Introduction au langage LaTeX

Nicolas Delestre

Table des matières

1 Produire son premier PDF **2**
- 1.1 Décrire un document pour le composer . . . 2
- 1.2 Du fichier source au PDF . . . 3
- 1.3 Créer et compiler un premier document . . . 4
  - 1.3.1 Un fichier complet . . . 4
  - 1.3.2 La compilation et son résultat . . . 5
  - 1.3.3 Les fichiers produits par la compilation . . . 5

2 Comprendre la syntaxe et structurer le document **6**
- 2.1 La syntaxe du fichier source . . . 6
  - 2.1.1 Commandes et arguments . . . 6
  - 2.1.2 Environnements . . . 7
  - 2.1.3 Groupes . . . 7
  - 2.1.4 Commentaires . . . 7
  - 2.1.5 Un exemple de portée locale . . . 7
- 2.2 La classe et les options du document . . . 8
  - 2.2.1 Le rôle de la classe . . . 8
  - 2.2.2 Les options usuelles . . . 9
  - 2.2.3 Les packages . . . 9
- 2.3 Les titres et les paragraphes . . . 9
  - 2.3.1 Une hiérarchie de titres . . . 9
  - 2.3.2 Paragraphes et espaces dans le source . . . 10
- 2.4 Le titre et la table des matières . . . 11
  - 2.4.1 Déclaration et composition du titre . . . 11
  - 2.4.2 Une table des matières calculée . . . 12

3 Rédiger le texte courant **14**
- 3.1 Les espaces et les caractères réservés . . . 14
  - 3.1.1 Du texte source aux paragraphes . . . 14
  - 3.1.2 Les caractères qui ont un rôle syntaxique . . . 14
- 3.2 Les listes . . . 15
  - 3.2.1 Les listes non ordonnées . . . 16
  - 3.2.2 Les listes ordonnées . . . 16
  - 3.2.3 Les listes de descriptions . . . 16
  - 3.2.4 L'imbrication des listes . . . 16
- 3.3 Les notes et les commandes usuelles . . . 17
  - 3.3.1 Les notes de bas de page . . . 17
  - 3.3.2 L'emphase et la forme des caractères . . . 17
  - 3.3.3 Les changements locaux de taille . . . 17
  - 3.3.4 Le texte littéral . . . 18

4 Insérer tableaux et figures, puis les référencer **20**
4.1 La construction des tableaux . . . 20
4.1.1 Les colonnes et les cellules . . . 20
4.1.2 La largeur des colonnes . . . 21
4.1.3 Les extensions pour les tableaux . . . 21
4.2 L'inclusion des images . . . 22
4.2.1 Le fichier graphique et son chemin . . . 22
4.2.2 Les dimensions et les transformations . . . 23
4.2.3 Les formats matriciels et vectoriels . . . 23
4.3 Les flottants, les légendes et les références . . . 23
4.3.1 Le placement des flottants . . . 24
4.3.2 La légende et l'étiquette . . . 24
4.3.3 Les références et les listes . . . 25

5 Composer des mathématiques **27**
5.1 Les modes mathématiques . . . 27
5.1.1 Les formules dans le texte . . . 27
5.1.2 Les formules hors texte . . . 27
5.1.3 Les écritures usuelles . . . 28
5.2 Les équations et leurs références . . . 29
5.2.1 Les équations numérotées . . . 29
5.2.2 Les équations non numérotées . . . 29
5.2.3 Les étiquettes et les références croisées . . . 30

6 Organiser un document en plusieurs fichiers **31**
6.1 Du fichier unique au fichier maître . . . 31
6.1.1 Les limites d'un document monolithique . . . 31
6.1.2 L'insertion du contenu d'un fichier . . . 32
6.2 Un projet de document à deux auteurs . . . 32
6.2.1 Un projet complet . . . 32
6.2.2 Les chemins relatifs . . . 34
6.2.3 Une compilation globale . . . 34
6.2.4 La répartition des contributions . . . 35
6.2.5 Les conventions communes . . . 35

7 Découvrir les usages spécialisés **36**
7.1 Le code source et le pseudocode . . . 36
7.1.1 L'inclusion d'un programme avec listings . . . 36
7.1.2 La description d'une procédure en pseudocode . . . 37
7.2 Les dessins avec TikZ . . . 38
7.2.1 Les nœuds et les flèches . . . 38
7.2.2 Les formes et les couleurs . . . 39
7.3 L'interaction avec Python . . . 39
7.3.1 Le calcul et son résultat dans le texte . . . 40
7.3.2 La chaîne de compilation et ses dépendances . . . 40

8 Personnaliser et poursuivre son apprentissage **42**
8.1 Définir ses commandes et ses environnements . . . 42
8.1.1 Une commande sans argument . . . 42
8.1.2 Une commande avec un argument . . . 42
8.1.3 Un environnement personnalisé . . . 43
8.1.4 La redéfinition et sa portée . . . 44

8.2 Quelques packages complémentaires 44
8.2.1 Les marges, les en-têtes et les pieds de page 45
8.2.2 Les polices et les symboles mathématiques 45
8.2.3 Les adresses et les liens hypertextes 46
8.3 La documentation et les ressources en ligne 46
8.3.1 La consultation depuis la ligne de commande 46
8.3.2 CTAN, la FAQ francophone et les guides de référence 47

Conclusion générale **48**

Avant-propos

Ce polycopié a été élaboré à partir de mon support de cours.

La rédaction de ce document a été réalisée avec l'aide de *Codex (GPT-6.1 Sol Moyen)*, un modèle d'intelligence artificielle générative développé par OpenAI. Il a été utilisé comme outil de rédaction initiale à partir de mes supports de cours. Les textes produits ont été relus, discutés et modifiés au cours de la rédaction. Le contenu pédagogique, les choix de notions abordées, leur organisation générale et les supports ayant servi de sources restent issus de mes enseignements.

Cette mention a pour objectif de rendre explicite l'utilisation d'un système d'intelligence artificielle générative dans la production du document et de distinguer cet usage des sources pédagogiques et scientifiques sur lesquelles le contenu s'appuie.

Chapitre 1

Produire son premier PDF

Ce chapitre présente d'abord les approches WYSIWYG et WYMIWYG, la séparation entre le contenu et sa présentation, ainsi que les rôles de la langue et du type de document. Il décrit ensuite la chaîne de compilation et les rôles de l'éditeur de texte, de la distribution LaTeX et du lecteur PDF. La dernière section détaille un premier fichier source complet, sa compilation et le résultat obtenu, puis présente les fichiers générés.

1.1 Décrire un document pour le composer

Dans un traitement de texte à interface graphique, l'auteur peut agir directement sur une représentation des pages : il sélectionne du texte, choisit une taille de caractères et observe le résultat. Cette approche est souvent désignée par le sigle *WYSIWYG*, pour *What You See Is What You Get*, soit « ce que vous voyez est ce que vous obtenez ».

Cette proximité entre la rédaction et son résultat visuel peut conduire l'auteur à travailler simultanément sur le fond et sur la forme. Au lieu de se concentrer sur ses idées et leur organisation, il interrompt sa rédaction pour déplacer une image, ajuster un espacement ou modifier une taille de caractères. Ces ajustements peuvent devenir une source de perte de temps, notamment lorsqu'une modification du contenu oblige à reprendre la mise en page.

La facilité des modifications locales favorise aussi des usages qui nuisent à l'uniformité du document. Un auteur non spécialiste peut mettre chaque titre en forme séparément, sans utiliser de styles ni de modèle commun. Des éléments de même nature risquent alors de présenter des tailles, des espacements ou des alignements différents. Les traitements de texte disposent de styles et de modèles pour éviter ces incohérences, mais leur utilisation demande une démarche que la mise en forme directe n'impose pas.

Cette approche tend enfin à confier à une même personne des tâches qui relèvent de métiers distincts. L'auteur rédige le contenu et en construit l'organisation ; dans une chaîne éditoriale, l'éditeur définit les choix de présentation et coordonne la mise en forme, tandis que l'illustrateur conçoit les représentations graphiques. La composition et la mise en page mobilisent ainsi des compétences spécifiques, que l'auteur ne possède pas nécessairement. Séparer ces responsabilités permet à l'auteur de se concentrer sur le texte tout en s'appuyant sur des règles de présentation établies pour l'ensemble du document.

Avec LaTeX, l'auteur travaille sur une description textuelle du document. Il indique, par exemple, qu'un passage constitue un titre de section, puis laisse le système déterminer sa présentation à partir de règles communes. Cette approche est désignée par le sigle *WYMIWYG*, pour *What You Mean Is What You Get* : l'auteur exprime le rôle des éléments et le logiciel en compose le rendu. Le fichier qui contient cette description est appelé le *fichier source* ; son nom se termine habituellement par `.tex`.

La présentation d'un texte dépend notamment des conventions typographiques de sa langue. Les usages français et anglais diffèrent, par exemple, par les guillemets, les espaces associés à

certains signes de ponctuation et la présentation des listes. Les réglages d'un traitement de texte ou d'un modèle peuvent correspondre à des conventions anglophones : une liste non numérotée peut ainsi être introduite par des puces rondes, alors que la composition française emploie couramment des tirets. La langue du correcteur orthographique ne détermine pas nécessairement l'ensemble de ces choix de présentation. Les possibilités d'adaptation dépendent du logiciel et du modèle utilisés.

Dans LaTeX, la déclaration de la langue permet d'appliquer un ensemble cohérent de conventions sans régler séparément chaque élément. Le package `babel`, configuré pour le français, adapte notamment la coupure des mots, certains espacements de ponctuation, les intitulés automatiques et, dans sa configuration usuelle, les marqueurs des listes non numérotées. Une configuration anglaise applique les conventions correspondantes. Cette automatisation repose sur la configuration linguistique chargée ; elle ne traduit pas le contenu du document.

La structure et la mise en forme dépendent également de la nature du document : un article scientifique, un rapport technique et un livre ne suivent pas nécessairement la même organisation. Un traitement de texte généraliste laisse habituellement accessibles les mêmes niveaux de titres, quel que soit le texte rédigé ; un modèle peut toutefois en encadrer l'utilisation. Dans LaTeX, le type de document est déclaré par le choix d'une *classe*, qui définit à la fois des règles de présentation et les commandes de structuration disponibles. La classe standard `article` propose des sections, mais pas de commande de chapitre, tandis que les classes `report` et `book` proposent aussi des chapitres. Une commande de structuration absente de la classe n'est donc pas utilisable telle quelle : le choix de la classe délimite une organisation adaptée au document. La langue et la classe répondent ainsi à deux questions complémentaires : quelles conventions linguistiques appliquer et quelle nature de document composer.

La séparation entre contenu et présentation permet de conserver des titres cohérents dans tout un rapport, de modifier globalement leur apparence et de laisser le système gérer leur numérotation. Elle facilite également la réutilisation du texte et sa répartition entre plusieurs fichiers. Elle suppose en contrepartie d'apprendre une syntaxe et d'interpréter les messages produits lorsqu'une instruction est incorrecte. La composition automatique peut elle aussi présenter des défauts que la lecture du résultat permet de repérer.

1.2 Du fichier source au PDF

La transformation du source en document composé est appelée la *compilation*. Dans ce cours, elle sera réalisée avec le programme `pdflatex`. Le schéma de la figure 1.1 résume cette chaîne. Une modification du fichier source n'affecte le PDF qu'après son enregistrement et une nouvelle compilation.

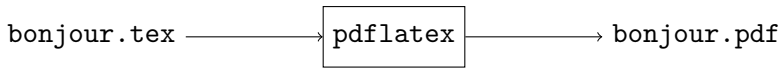


FIGURE 1.1 – Le fichier source est l'entrée de la compilation ; le PDF en est le résultat destiné à la lecture.

Trois outils ont des fonctions distinctes. L'*éditeur de texte* permet de saisir et d'enregistrer le source. La *distribution LaTeX* fournit les logiciels de compilation et un ensemble de fichiers nécessaires, notamment les classes et les extensions. Le *lecteur PDF* affiche le résultat. Le choix de l'éditeur ne change pas la syntaxe du document. La compilation des exemples est effectuée depuis le terminal et repose sur une distribution dans laquelle la commande `pdflatex` est accessible. La présence d'un éditeur de texte ne garantit pas que cette commande soit installée. La procédure d'installation dépend du système utilisé et n'est pas nécessaire à la compréhension du source présenté ici.

1.3 Créer et compiler un premier document

LaTeX est un langage de programmation destiné à la description et à la composition de documents [1]. Le fichier source constitue un programme qui suit une structure définie, contient du texte et des instructions, et peut utiliser des bibliothèques appelées *packages*. Les instructions prennent notamment la forme de *commandes* et d'*environnements* : les premières expriment une action ou un réglage, tandis que les seconds délimitent un contenu soumis à des règles communes. L'exemple suivant présente ces éléments dans un document complet.

1.3.1 Un fichier complet

Le listing 1.1 présente le contenu d'un fichier nommé `bonjour.tex`, enregistré avec l'encodage UTF-8. Ce code constitue un document complet : aucun autre fichier source ni aucune image ne sont nécessaires. Les lignes commençant par `%` sont des commentaires destinés au lecteur du source et n'apparaissent pas dans le PDF.

Listing 1.1 – Premier document complet.

```
\documentclass[a4paper,11pt]{article}
% Encodage du fichier source.
\usepackage[utf8]{inputenc}
% Configuration des fontes et du français pour pdflatex.
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\begin{document}
Bonjour \LaTeX{}!
\end{document}
```

La première ligne choisit la *classe* `article`, qui convient ici à un document court. La classe définit une organisation et des règles de présentation par défaut. Les options `a4paper` et `11pt`, placées entre crochets, demandent un papier A4 et une taille de base des caractères de onze points. Les accolades entourent le nom de la classe, argument obligatoire de la commande.

Les commandes `\usepackage` chargent des extensions, aussi appelées *packages*. Un package ajoute ou configure des fonctionnalités sans qu'il soit nécessaire de les programmer dans chaque document. L'option `utf8` du package `inputenc` déclare l'encodage UTF-8 du fichier source. L'option `T1` du package `fontenc` choisit un encodage de fontes adapté notamment au français avec `pdflatex`. Cet encodage des caractères dans les fontes est distinct de l'encodage UTF-8 du fichier source. Le package `babel`, chargé avec l'option `french`, configure notamment les règles linguistiques et typographiques françaises.

Tout ce qui précède `\begin{document}` constitue le *préambule*. Le contenu à composer se trouve dans l'environnement `document`, entre `\begin{document}` et `\end{document}`. Dans cet exemple, il se limite à une salutation. La commande `\LaTeX` produit le logo du langage ; les accolades vides qui la suivent délimitent son nom sans ajouter de texte visible. Le chapitre consacré à la syntaxe précisera le rôle des arguments, des groupes et des espaces autour des commandes.

Un LaTeX récent considère UTF-8 comme encodage d'entrée par défaut. La déclaration explicite de cet encodage dans le listing 1.1 rend néanmoins ce choix visible dans le source. Une déclaration `latin1` correspond à un autre encodage et serait incohérente avec un fichier enregistré en UTF-8.

1. LaTeX s'appuie sur le langage TeX, qui est Turing complet : il permet en principe d'exprimer tout calcul réalisable par un algorithme, sous réserve de disposer du temps et de la mémoire nécessaires.

1.3.2 La compilation et son résultat

La commande suivante compile le fichier `bonjour.tex` lorsqu'elle est exécutée dans le répertoire qui le contient :

```
$ pdflatex bonjour.tex
```

Lors de la compilation, le programme lit le source et charge la classe ainsi que les packages. Il produit le fichier `bonjour.pdf` lorsque la compilation aboutit. Le PDF obtenu contient le document mis en page. La figure 1.2 reproduit la zone contenant la salutation dans le PDF réellement compilé depuis le listing 1.1 ; les marges et le numéro de page sont laissés hors du cadrage.



FIGURE 1.2 – Extrait du PDF produit par le premier document.

La commande `\documentclass` et les lignes du préambule ne sont pas imprimées : elles règlent la composition. Le texte du corps apparaît sur la page, et la commande du logo est remplacée par sa représentation typographique. Le point d'exclamation reçoit l'espacement français défini par `babel`, bien qu'aucune espace ne le précède dans le source. Dans ce cours, cette convention s'applique au texte courant devant les signes `:`, `;`, `?` et `!` ; les contenus littéraux, comme le code à reproduire, conservent leurs caractères propres.

Le remplacement du mot `Bonjour` par `Bienvenue` dans le source se traduit par le même changement dans le PDF après enregistrement et nouvelle compilation. Sans cette nouvelle compilation, le PDF conserve le texte précédent. L'écriture du source, son enregistrement, sa compilation et la lecture du résultat constituent ainsi les étapes du cycle de production du document.

1.3.3 Les fichiers produits par la compilation

La compilation crée également des fichiers auxiliaires. Dans un répertoire contenant initialement le seul fichier `bonjour.tex`, la commande `ls` donne, après compilation, le résultat suivant :

```
$ ls
bonjour.aux  bonjour.log  bonjour.pdf  bonjour.tex
```

La sortie de `ls` énumère ici le source et les trois fichiers générés par `pdflatex`. Le fichier `bonjour.log` contient son journal détaillé ; le fichier `bonjour.aux` conserve des informations que LaTeX pourra réutiliser lors d'une future compilation. Des documents comportant une table des matières ou des références nécessiteront plusieurs compilations.

La présence d'un PDF ne suffit pas à garantir une compilation correcte : un ancien PDF peut subsister, et certaines erreurs permettent de poursuivre avec un résultat incomplet. Les messages de compilation et la correspondance entre le source enregistré et le PDF permettent d'évaluer la validité du résultat. Ce premier document fournit désormais une base compilable pour apprendre à structurer et enrichir le contenu.

Chapitre 2

Comprendre la syntaxe et structurer le document

Ce chapitre détaille d'abord la syntaxe des commandes, des environnements, des groupes et des commentaires. Il présente ensuite les principales classes de documents et leurs options, puis la hiérarchie des titres et la formation des paragraphes. La dernière section décrit la déclaration du titre, des auteurs et de la date, ainsi que la génération de la table des matières et les compilations successives qu'elle nécessite.

2.1 La syntaxe du fichier source

Un code LaTeX est constitué principalement de commandes et d'environnements.

2.1.1 Commandes et arguments

Une commande s'écrit avec une barre oblique inverse, le caractère `\`, suivie habituellement d'une suite de lettres qui constitue son nom. Par exemple, `\LaTeX` produit le logo du langage et `\textbf` permet de composer du texte en gras. LaTeX est sensible à la casse : les majuscules et les minuscules font partie du nom de la commande. Ainsi, `\LaTeX` et `\latex` sont deux noms différents, et le second n'est pas une commande définie par défaut. Il existe aussi des commandes formées d'une barre oblique inverse et d'un seul caractère non alphabétique, comme `\%` pour afficher le signe pour cent.

Une commande peut recevoir un ou plusieurs *arguments obligatoires*, qui lui fournissent les informations nécessaires à son action. Chaque argument est délimité par sa propre paire d'accolades ; dans l'écriture adoptée ici, les arguments suivent directement la commande, sans espace intercalée. Dans l'exemple suivant, la commande reçoit un argument, le texte à mettre en gras :

```
\textbf{un mot}
```

Le résultat est « **un mot** » ; les accolades délimitent l'argument sans apparaître dans le document. Lorsqu'une commande reçoit plusieurs arguments obligatoires, leur ordre détermine leur rôle. Pour une commande sans argument utilisée dans le texte, comme `\LaTeX`, l'ajout d'accolades vides permet de conserver l'espace qui suit son nom :

```
\LaTeX{} est un langage.
```

Le résultat est « LaTeX est un langage. » Sans les accolades vides, l'espace placée immédiatement après le nom de la commande serait consommée lors de sa lecture, et le mot suivant serait accolé au logo. Les accolades vides ne fournissent pas ici un argument : elles séparent la commande de l'espace à conserver.

Certaines commandes acceptent aussi des *arguments optionnels*, qui précisent leur comportement. Dans les déclarations de classe ou de package, un argument optionnel placé avant l'argument obligatoire regroupe les options entre crochets. Les options sont séparées par des virgules, par exemple :

```
\documentclass[a4paper,11pt]{article}
```

L'argument obligatoire `article` désigne la classe, tandis que les options `a4paper` et `11pt` précisent le format du papier et la taille de base des caractères. En leur absence, la classe utilise ses valeurs par défaut. La position et le rôle des arguments optionnels dépendent de la commande considérée.

2.1.2 Environnements

Un environnement délimite une portion du document par les commandes `\begin` et `\end`, qui indiquent toutes deux le nom de cet environnement :

```
\begin{nom}
Contenu de l'environnement
\end{nom}
```

Ce schéma décrit une syntaxe : `nom` représente le nom d'un environnement existant. L'environnement `document` contient le corps du document. D'autres environnements décrivent notamment les listes, les tableaux et les équations. Lorsque des environnements sont imbriqués, le dernier ouvert est le premier fermé.

Les environnements peuvent aussi recevoir des arguments obligatoires ou optionnels. Ces arguments sont placés après `\begin{nom}`, comme dans `\begin{figure}[htbp]` pour une figure. Le rôle de ces options de placement est présenté dans la section 4.3.

2.1.3 Groupes

Des accolades peuvent aussi former un *groupe*, sans être l'argument d'une commande. Un groupe limite la portée de nombreux réglages. Par exemple, dans `{\large Texte agrandi.}`, la déclaration `\large` change la taille des caractères jusqu'à la fermeture du groupe ; le texte qui suit retrouve la taille précédente. La différence est importante : `\textbf{...}` agit sur son argument, tandis que `\large` est une déclaration dont l'effet dépend de sa portée.

2.1.4 Commentaires

Le caractère `%` introduit un commentaire qui s'étend jusqu'à la fin de la ligne du source. Le texte commenté et la fin de ligne correspondante sont ignorés lors de la composition. Le commentaire peut expliquer une instruction ou conserver une information de rédaction sans apparaître dans le PDF. L'écriture `\%` conserve au contraire le signe comme contenu visible.

2.1.5 Un exemple de portée locale

Le listing 2.1 rassemble ces constructions dans un document complet. Le commentaire explique la portée du groupe, mais seule la phrase concernée apparaît en caractères agrandis dans la figure 2.1. La commande `\emph` marque une insistance, rendue ici en italique, tandis que `\textbf` demande une graisse plus forte.

Listing 2.1 – Commandes, commentaire et groupe local.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
```

```
\begin{document}
\LaTeX{} compose un \emph{document structuré}.
Un \textbf{résultat important} est mis en gras.

% Le groupe limite le changement de taille.
{\large Cette phrase est agrandie.}

Cette phrase retrouve la taille normale.
\end{document}
```

LaTeX compose un *document structuré*. Un **résultat important** est mis en gras.

Cette phrase est agrandie.

Cette phrase retrouve la taille normale.

FIGURE 2.1 – Texte composé à partir du listing 2.1 ; la taille normale est rétablie après le groupe.

2.2 La classe et les options du document

Cette section précise le rôle de la classe, de ses options et des packages chargés dans le préambule.

2.2.1 Le rôle de la classe

La commande `\documentclass` définit le cadre général du document et figure au début du préambule. Son argument obligatoire est le nom de la classe. Ce choix détermine des règles de présentation, mais aussi les commandes disponibles : une classe de lettre organise des informations différentes de celles d'un livre ou d'une présentation. Le tableau 2.1 distingue les principales classes selon leur usage.

Classe	Usage et organisation générale
`article`	Articles et documents organisés en sections, sans chapitres.
`report`	Rapports pouvant comporter des parties, des chapitres et des sections.
`book`	Livres, avec une organisation en parties et chapitres et des conventions de pagination adaptées.
`letter`	Courriers, avec une organisation propre aux lettres.
`beamer`	Présentations, dont le contenu est réparti dans des cadres de diapositives.

TABLE 2.1 – Quelques classes de documents et leurs usages.

Les noms de classe indiquent une organisation possible et ne fixent pas le sujet du texte. Un rapport court peut ainsi être rédigé avec `article` si une structure en sections suffit. Les trois classes standard `article`, `report` et `book` possèdent une commande `\part`. La commande `\chapter` existe dans `report` et `book`, mais pas dans `article`. Le choix d'une classe précède donc celui des niveaux de titres utilisés dans le corps.

2.2.2 Les options usuelles

Les options de la classe se trouvent entre crochets, avant son nom, et sont séparées par des virgules. La déclaration suivante définit, par exemple, un rapport sur papier A4, avec une taille de base de onze points et une mise en page prévue pour le recto seul :

```
\documentclass[a4paper,11pt,oneside]{report}
```

Il s'agit d'une déclaration de préambule, et non d'un document autonome. Les classes standard `article`, `report` et `book` acceptent les tailles de base `10pt`, `11pt` et `12pt`. Ces options ne fixent pas la même taille pour tous les éléments : les titres et les notes conservent des tailles relatives définies par la classe. Le format `a4paper` et le format `letterpaper` correspondent à deux dimensions de papier différentes ; l'option `landscape` permet une orientation en paysage.

Les options `onecolumn` et `twocolumn` déterminent une composition sur une ou deux colonnes. Les options `oneside` et `twoside` organisent la mise en page pour le recto seul ou le recto verso, notamment la distinction entre pages paires et impaires. Elles ne commandent pas le fonctionnement de l'imprimante. Dans les classes `report` et `book`, `openright` fait commencer les chapitres sur une page impaire, ce qui peut introduire une page blanche, tandis que `openany` permet un début sur la prochaine page disponible. Par défaut, `report` utilise `oneside` et `openany`, tandis que `book` utilise `twoside` et `openright`.

Les options disponibles dépendent de la classe : une option reconnue par `report` n'est pas nécessairement pertinente pour `letter` ou `beamer`.

2.2.3 Les packages

Les packages, chargés après la classe, complètent ou adaptent ensuite ses fonctionnalités. Leur chargement suit la syntaxe suivante :

```
\usepackage[options]{nom}
```

Les crochets sont absents lorsqu'aucune option n'est indiquée.

Avec `pdflatex`, quelques packages utiles sont :

— `inputenc`, qui déclare l'encodage du fichier source ;
— `fontenc`, qui choisit un encodage de fontes adapté à la langue et au moteur de composition ;
— `babel`, qui configure les règles linguistiques et typographiques selon la langue du document (arguments optionnels : `french`, `english`, `german`, etc.) ;
— `graphicx`, qui permet d'inclure des images ;
— `amsmath`, qui fournit des environnements et des commandes pour composer des formules mathématiques ;
— `hyperref`, qui crée des liens hypertextes dans le document PDF ;
— `geometry`, qui permet de régler les marges et la taille du papier ;
— `listings`, qui permet d'insérer du code source avec une mise en forme adaptée ;
— `listingsutf8`, qui complète `listings` pour l'inclusion de certains fichiers encodés en UTF-8 ;
— `tikz`, qui permet de créer des graphiques vectoriels.

2.3 Les titres et les paragraphes

2.3.1 Une hiérarchie de titres

Une commande de sectionnement décrit la place d'un titre dans l'organisation du document. Son argument contient le titre lui-même ; la classe détermine sa présentation et sa numérotation. Le tableau 2.2 présente les niveaux disponibles dans les trois classes standard étudiées ici.

Niveau	Commande	Classes concernées
Partie	`\part`	`article`, `report`, `book`
Chapitre	`\chapter`	`report`, `book`
Section	`\section`	Les trois classes
Sous-section	`\subsection`	Les trois classes
Sous-sous-section	`\subsubsection`	Les trois classes
Niveau suivant	`\paragraph`	Les trois classes
Dernier niveau	`\subparagraph`	Les trois classes

TABLE 2.2 – Hiérarchie des commandes de sectionnement.

Ces niveaux ne sont pas tous nécessaires dans un même document. Une partie regroupe de grands ensembles, tandis qu'une section et une sous-section organisent des développements plus courts. La commande `\paragraph` introduit un titre de niveau inférieur ; elle ne désigne pas un paragraphe de texte courant. Dans les classes standard, ce titre est généralement placé au début de la même ligne que le texte qui suit.

La numérotation est calculée automatiquement à partir des commandes et de leur niveau. Dans `article`, les sections sont numérotées indépendamment de tout chapitre ; dans `report` et `book`, leur numéro inclut habituellement celui du chapitre. La profondeur de numérotation dépend aussi de la classe : par défaut, `article` numérote jusqu'aux sous-sous-sections, tandis que `report` et `book` s'arrêtent aux sous-sections. Un titre peut donc exister sans recevoir de numéro à un niveau profond.

La forme étoilée, par exemple `\section*{Remarque}`, produit un titre sans numéro et sans entrée automatique dans la table des matières. Elle ne fait pas progresser le compteur de la section. Le choix entre les deux formes porte ainsi sur le rôle du titre dans l'organisation, et pas seulement sur son apparence.

2.3.2 Paragraphes et espaces dans le source

Dans le texte courant, plusieurs espaces ou tabulations successifs produisent généralement une seule espace. Un simple retour à la ligne dans le source est lui aussi traité comme une séparation entre mots : il ne fixe pas la fin de la ligne dans le PDF. LaTeX répartit les mots sur les lignes selon la largeur disponible. Une ligne vide termine en revanche le paragraphe en cours ; la commande `\par` produit également cette séparation. Le retrait de première ligne et l'espacement entre paragraphes dépendent des règles de composition du document.

Le listing 2.2 présente un document complet. Les deux premières phrases sont écrites sur des lignes successives et appartiennent à un même paragraphe. La ligne vide qui les sépare de la phrase suivante crée un second paragraphe. Les espaces répétés entre les premiers mots n'introduisent pas un espacement supplémentaire dans le résultat de la figure 2.2.

Listing 2.2 – Titres, paragraphes et section étoilée.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\begin{document}
\section{Introduction}
Le document     présente une mesure.
Cette phrase appartient au même paragraphe.

Ce second paragraphe précise le contexte.

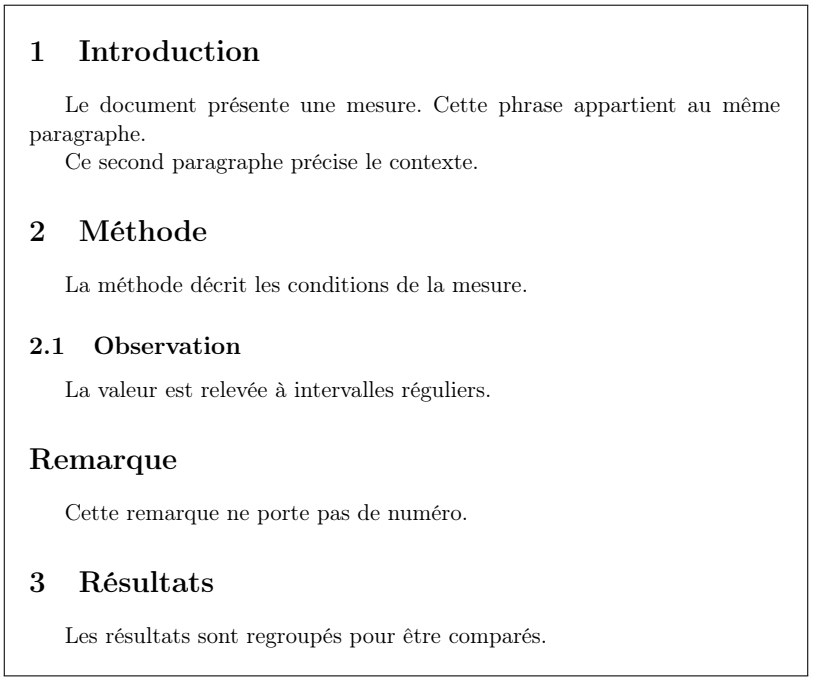
\section{Méthode}
```

```
La méthode décrit les conditions de la mesure.

\subsection{Observation}
La valeur est relevée à intervalles réguliers.

\section*{Remarque}
Cette remarque ne porte pas de numéro.

\section{Résultats}
Les résultats sont regroupés pour être comparés.
\end{document}
```



1 Introduction

Le document présente une mesure. Cette phrase appartient au même paragraphe.

Ce second paragraphe précise le contexte.

2 Méthode

La méthode décrit les conditions de la mesure.

2.1 Observation

La valeur est relevée à intervalles réguliers.

Remarque

Cette remarque ne porte pas de numéro.

3 Résultats

Les résultats sont regroupés pour être comparés.

FIGURE 2.2 – Résultat du listing 2.2 ; la remarque n'interrompt pas la numérotation des sections.

2.4 Le titre et la table des matières

2.4.1 Déclaration et composition du titre

Les commandes `\title`, `\author` et `\date` déclarent les informations associées au titre. Elles figurent ici dans le préambule et ne produisent pas immédiatement de texte sur la page. La commande `\maketitle`, placée dans le corps du document, compose ces informations selon les règles de la classe. Le listing 2.3 présente cette distinction dans un document complet. Il reprend une organisation en sections et lui ajoute un titre, deux auteurs et une table des matières.

Listing 2.3 – Document avec titre et table des matières.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\title{Compte rendu de mesure}
\author{Camille Martin \and Alex Bernard}
\date{Octobre 2026}
```

```
\begin{document}
\maketitle
\tableofcontents

\section{Introduction}
Le document présente une mesure.

\section{Méthode}
La méthode décrit les conditions de la mesure.

\subsection{Observation}
La valeur est relevée à intervalles réguliers.

\section{Résultats}
Les résultats sont regroupés pour être comparés.
\end{document}
```

Dans l'argument de `\author`, la commande `\and` sépare les deux auteurs. La déclaration `\date{Octobre 2026}` fixe le texte de la date. Une déclaration `\date{}` laisse cette information vide, tandis que l'absence de déclaration conduit les classes standard à utiliser la date courante lors de la compilation. La commande `\maketitle` ne crée pas toujours une page séparée : dans la configuration par défaut de `article`, le titre figure en tête de la première page ; dans `report` et `book`, une page de titre distincte est produite.

2.4.2 Une table des matières calculée

La commande `\tableofcontents` insère la table des matières à l'endroit où elle apparaît dans le corps du document. Les intitulés, les numéros et les pages proviennent des commandes de sectionnement. Dans le document de la figure 2.3, les sections et la sous-section apparaissent ainsi automatiquement dans la table. La profondeur des titres affichés dépend de la configuration de la classe.

La table des matières repose sur un fichier auxiliaire dont l'extension est `.toc`. Lors d'une première compilation à partir des seuls sources, les informations des titres et de leur position sont enregistrées, mais la table ne dispose pas encore de ces données lorsqu'elle est composée. La compilation suivante les relit et permet d'afficher les entrées. Pour ce document court, deux compilations suffisent normalement :

```
$ pdflatex titre.tex
$ pdflatex titre.tex
```

Une modification des titres ou de la longueur du texte peut changer la pagination. Le nombre de compilations nécessaire n'est donc pas toujours limité à deux : les compilations successives se poursuivent jusqu'à stabilisation des informations utilisées. La table des matières et les références s'appuient sur ce même principe de réutilisation des données d'une compilation précédente.

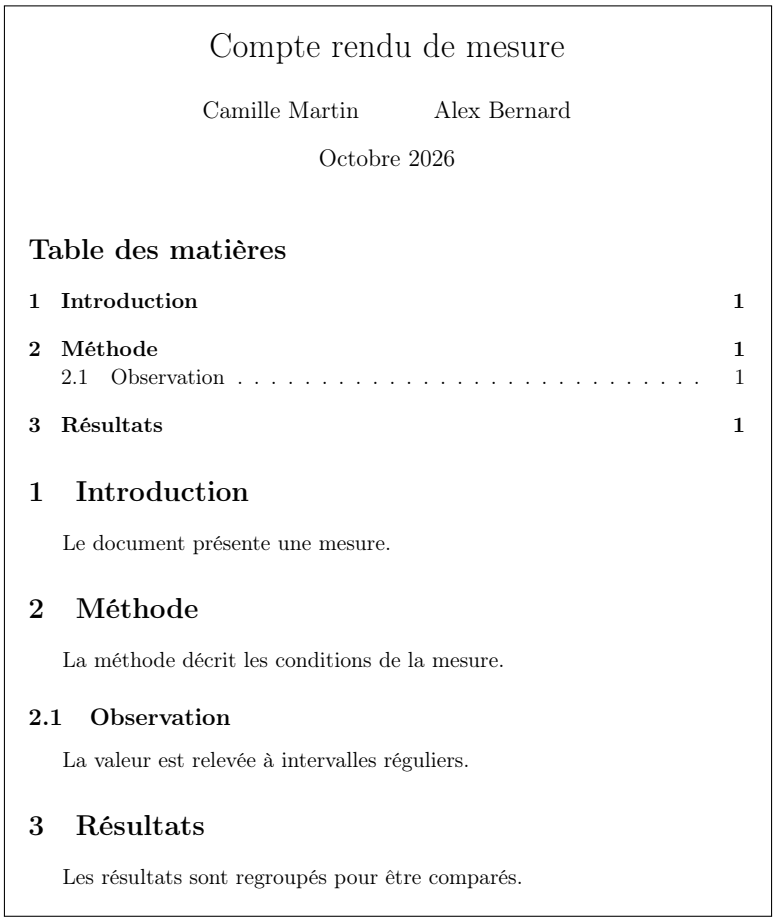


FIGURE 2.3 – Titre, auteurs, date et table des matières du listing 2.3, après stabilisation des informations.

Chapitre 3

Rédiger le texte courant

Ce chapitre présente les espaces et les caractères réservés dans le texte courant, puis les environnements de listes et leur imbrication. Il décrit ensuite les notes de bas de page, les commandes usuelles de mise en forme et les changements locaux de taille. La dernière partie distingue le texte composé par LaTeX des contenus affichés littéralement avec la commande `\verb` et l'environnement `verbatim`.

3.1 Les espaces et les caractères réservés

3.1.1 Du texte source aux paragraphes

Dans le texte courant, la disposition des lignes du source ne reproduit pas celle des lignes du PDF. Comme l'illustre l'exemple de la sous-section 2.3.2, les retours à la ligne simples et les espaces répétés servent généralement à séparer les mots, tandis qu'une ligne vide délimite les paragraphes. La commande `\\` impose un retour à la ligne, mais ne remplace pas cette séparation en paragraphes. Son utilisation doit rester parcimonieuse et justifiée par la nature du contenu, par exemple pour séparer les lignes d'une adresse. Des retours à la ligne forcés destinés à ajuster visuellement un paragraphe risquent de dégrader sa composition lorsque le texte ou la largeur disponible change. Un texte rédigé en paragraphes laisse à LaTeX le calcul des coupures de lignes et de pages.

Dans certaines expressions, une coupure entre deux mots serait indésirable. Le caractère `~` produit une espace insécable, c'est-à-dire une espace qui interdit un changement de ligne à cet endroit. Par exemple, `lot~A` maintient le nom du lot et sa lettre sur la même ligne. Ce caractère ne sert pas à régler manuellement l'espace devant les signes de ponctuation : dans la configuration française utilisée ici, cette fonction relève de `babel`.

3.1.2 Les caractères qui ont un rôle syntaxique

Certains caractères ne sont pas du texte ordinaire pour LaTeX : ils participent à la syntaxe des instructions ou des environnements. Ainsi, `%` introduit un commentaire, alors que `\%` produit le signe pour cent. Le tableau 3.1 indique les principales écritures permettant d'afficher ces caractères dans le texte courant. Les usages mathématiques et les tableaux seront détaillés dans les chapitres correspondants.

Le caractère souligné d'un nom de fichier constitue un cas fréquent. La commande `\texttt` choisit une police à chasse fixe, mais ne transforme pas son argument en texte littéral : les caractères réservés y conservent leur rôle. L'écriture du nom `mesure_1.txt` comporte donc `\_` dans l'argument de cette commande. Les caractères accentués ordinaires, comme « é » ou « à », s'écrivent directement dans un fichier UTF-8 avec la configuration retenue.

Le listing 3.1 présente un document complet. Les deux premières phrases forment un seul paragraphe malgré le retour à la ligne et les espaces répétés. La ligne vide crée un autre pa-

Caractère	Écriture dans le source	Rôle habituel du caractère réservé
\	`\textbackslash{}`	Début d'une commande.
{ et }	`\{` et `\}`	Délimitation d'arguments ou de groupes.
%	`\%`	Début d'un commentaire.
$	`\$`	Délimitation du mode mathématique.
&	`\&`	Séparation de cellules dans un tableau.
#	`\#`	Désignation des paramètres dans la définition d'une macro.
_	`\_`	Introduction d'un indice en mode mathématique.
^	`\textasciicircum{}`	Introduction d'un exposant en mode mathématique.
~	`\textasciitilde{}`	Espace insécable.

TABLE 3.1 – Caractères réservés et écriture de leur forme visible dans le texte courant.

ragraphe, et le signe pour cent ainsi que le soulignement apparaissent dans le résultat de la figure 3.1.

Listing 3.1 – Espaces, paragraphes et caractères réservés.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\begin{document}
La mesure     est stable.
La lecture est identique.

Un second paragraphe précise le contexte.
Le lot~A contient vingt pièces.
Le taux vaut 5\%.
Le fichier est nommé \texttt{mesure\_1.txt}.
\end{document}
```

La mesure est stable. La lecture est identique.

Un second paragraphe précise le contexte. Le lot A contient vingt pièces. Le taux vaut 5%. Le fichier est nommé `mesure_1.txt`.

FIGURE 3.1 – Texte composé à partir du listing 3.1.

3.2 Les listes

Une liste rassemble des éléments dont la présentation traduit une relation : une énumération sans ordre particulier, une succession ordonnée ou un ensemble de termes accompagnés de leur description. LaTeX propose respectivement les environnements `itemize`, `enumerate` et `description`. Dans chaque cas, la commande `\item` marque le début d'un élément ; le texte de cet élément s'étend jusqu'au prochain `\item` de même niveau ou à la fin de la liste.

3.2.1 Les listes non ordonnées

L'environnement `itemize` convient lorsque les éléments ont le même statut et que leur ordre n'exprime pas une progression. Dans le listing 3.2, les mesures de température et de pression constituent deux éléments de ce type. Le marqueur et les retraits sont déterminés par la configuration du document : ici, `babel` configuré pour le français produit des tirets. Les marqueurs appartiennent à la présentation de la liste et ne sont pas saisis dans le contenu des éléments.

3.2.2 Les listes ordonnées

L'environnement `enumerate` met en évidence un ordre, par exemple les étapes d'une méthode. Dans l'exemple, l'acquisition des données précède leur analyse. Chaque commande `\item` produit un nouvel élément dont le numéro est calculé automatiquement. L'insertion d'un élément supplémentaire modifie les numéros suivants, sans changement manuel dans leur texte.

3.2.3 Les listes de descriptions

L'environnement `description` associe un intitulé à son développement. Cet intitulé est l'argument optionnel de `\item`, par exemple dans l'écriture suivante :

```
\item[Mesure] Valeur fournie par un capteur.
```

Le mot placé entre crochets est mis en évidence, généralement en gras, puis suivi du texte de description. Cette organisation est adaptée à un petit vocabulaire ou à la présentation des composantes d'un système.

3.2.4 L'imbrication des listes

Une liste peut être incluse dans un élément d'une autre liste. Le listing 3.2 place ainsi une liste non ordonnée à l'intérieur du premier élément d'une liste ordonnée. La liste interne se ferme avant le début du deuxième élément de la liste externe. L'indentation du source rend cette relation lisible ; ce sont toutefois les commandes d'ouverture, de fermeture et les `\item` qui déterminent la structure. La liste de descriptions qui suit est indépendante de cette imbrication.

Listing 3.2 – Listes ordonnée, non ordonnée et de descriptions.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\begin{document}
La méthode comporte deux étapes:
\begin{enumerate}
\item Acquisition des données.
  \begin{itemize}
  \item Mesure de la température.
  \item Mesure de la pression.
  \end{itemize}
\item Analyse des résultats.
\end{enumerate}

Les termes employés sont précisés ci-dessous:
\begin{description}
\item[Mesure] Valeur fournie par un capteur.
\item[Résultat] Valeur issue de l'analyse.
\end{description}
\end{document}
```

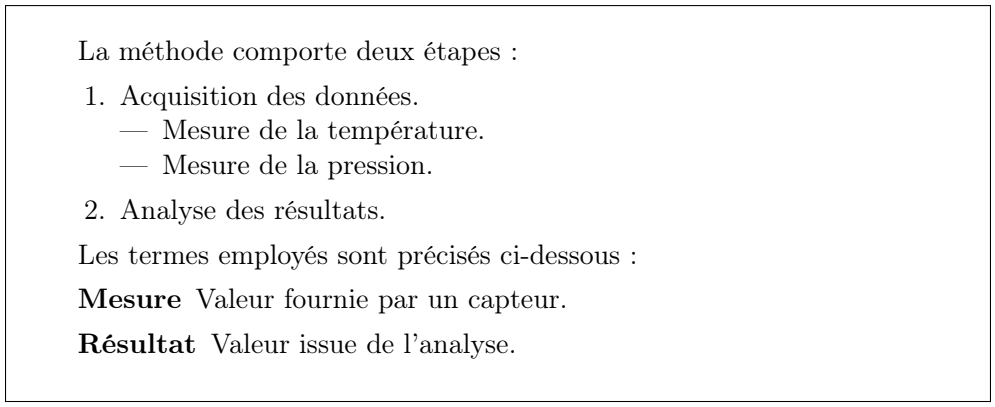


FIGURE 3.2 – Résultat du listing 3.2 ; les retraits rendent l'imbrication visible.

Les changements de niveau entraînent des retraits et peuvent modifier les marqueurs ou les formes de numérotation. La structure traduit ainsi le regroupement des idées, tandis que sa présentation reste gérée par le document.

3.3 Les notes et les commandes usuelles

3.3.1 Les notes de bas de page

Une note de bas de page apporte une précision sans interrompre le développement principal. La commande `\footnote` reçoit le texte de la note dans un argument obligatoire et s'insère à l'endroit de l'appel. LaTeX compose le repère dans le texte et place la note correspondante en bas de page. Sa numérotation est automatique et dépend de la classe : elle peut notamment reprendre à chaque chapitre dans un rapport ou un livre.

Dans le listing 3.3, la note précise les conditions de la mesure. Son texte est écrit immédiatement après le mot concerné dans le source, mais apparaît en bas de page dans le PDF. La figure 3.3 rapproche deux extraits de cette même page, le texte principal et la note, en laissant hors du cadrage l'espace qui les sépare.

3.3.2 L'emphase et la forme des caractères

Les commandes `\emph`, `\textbf`, `\textit` et `\texttt` s'appliquent au texte placé dans leur argument. La commande `\emph` exprime une insistance dont le rendu dépend du contexte ; dans du texte droit, elle produit habituellement de l'italique. La commande `\textit` demande directement l'italique, `\textbf` une graisse plus forte et `\texttt` une police à chasse fixe. Dans l'exemple, l'expression « valeur moyenne » porte une emphase et le mot « stable » est composé en gras. Ces changements ponctuels n'ont pas le rôle d'une commande de sectionnement : un mot agrandi ou en gras ne devient pas pour autant un titre de section.

3.3.3 Les changements locaux de taille

Les commandes de taille sont des déclarations, et non des commandes qui reçoivent le texte à modifier dans un argument. La succession suivante les présente de la plus petite à la plus grande dans les classes standard :

```
\tiny, \scriptsize, \footnotesize, \small,
\normalsize, \large, \Large, \LARGE, \huge, \Huge
```

Les différences de casse correspondent ici à des commandes distinctes. La taille exacte associée à chaque déclaration dépend de la classe et de la taille de base du document. La commande `\normalsize` désigne la taille normale dans cette configuration.

Un groupe limite la portée d'une déclaration de taille. Dans le listing 3.3, le texte en petits caractères se termine par `\par` avant la fermeture du groupe : le paragraphe est ainsi terminé alors que le réglage est encore actif. La taille des caractères et l'espacement entre les lignes sont alors cohérents pour l'ensemble du paragraphe. Le texte qui suit retrouve les réglages précédents.

Listing 3.3 – Note, emphase et changement local de taille.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\begin{document}
La mesure\footnote{La température est maintenue constante.}
est résumée par une \emph{valeur moyenne}.
Le résultat est \textbf{stable}.

{\small
Ce commentaire est composé en petits caractères.
\par}

Cette phrase retrouve la taille normale.
\end{document}
```

Texte principal

La mesure [1] est résumée par une *valeur moyenne*. Le résultat est **stable**.
Ce commentaire est composé en petits caractères.
Cette phrase retrouve la taille normale.

Note au bas de la même page

1. La température est maintenue constante.

FIGURE 3.3 – Deux extraits du résultat compilé du listing 3.3.

3.3.4 Le texte littéral

Certains contenus doivent être affichés tels qu'ils sont écrits, notamment une commande ou un extrait de fichier. La commande `\verb` et l'environnement `verbatim` permettent de conserver les caractères réservés sans les interpréter comme des instructions. Le texte est composé dans une police à chasse fixe.

La commande `\verb` convient à un extrait court dans une ligne de texte. Sa syntaxe diffère de celle des commandes à arguments présentées précédemment : le texte à afficher n'est pas délimité par une paire d'accolades. Le caractère placé immédiatement après le nom de la commande sert de délimiteur d'ouverture ; une seconde occurrence de ce même caractère marque la fin du texte. Avec la barre verticale `|` comme délimiteur, la syntaxe s'écrit :

```
\verb|contenu|
```

Les deux barres verticales délimitent le contenu mais n'apparaissent pas dans le résultat. Le texte situé entre elles est affiché littéralement, même s'il contient des commandes ou des caractères réservés. Par exemple :

```
\verb|\textbf{un mot}|
```

Cette écriture affiche la commande et ses accolades au lieu de produire du gras.

L'environnement `verbatim` convient à un bloc de plusieurs lignes. Il préserve les espaces et les retours à la ligne ; un signe pour cent y reste visible et ne commence pas un commentaire. Le listing 3.4 présente un document complet, dont le résultat est reproduit dans la figure 3.4. Les deux espaces devant le mot `taux` et les espaces autour du signe égal sont conservés.

Listing 3.4 – Commande et bloc affichés littéralement.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\begin{document}
La commande \verb|\textbf{un mot}| est affichée.

\begin{verbatim}
\textbf{un mot}
  taux = 5%
fichier = mesure_1.txt
\end{verbatim}
\end{document}
```

La commande `\textbf{un mot}` est affichée.

```
\textbf{un mot}
  taux = 5%
fichier = mesure_1.txt
```

FIGURE 3.4 – Résultat du listing 3.4 ; le code affiché n'est pas exécuté.

Chapitre 4

Insérer tableaux et figures, puis les référencer

Ce chapitre présente la construction des tableaux avec `tabular`, puis l'inclusion et le dimensionnement d'images avec le package `graphicx`. Il décrit ensuite les environnements flottants `table` et `figure`, leur placement, leurs légendes et les références croisées. La dernière partie explique la génération des listes de figures et de tableaux.

4.1 La construction des tableaux

Un tableau organise des informations en lignes et en colonnes. L'environnement `tabular` décrit cette organisation : il précise l'alignement des colonnes, le contenu des cellules et les éventuels traits de séparation. Il compose le tableau lui-même, indépendamment d'une légende ou d'un placement flottant. Ces dernières fonctions relèvent de l'environnement `table`, présenté dans la section 4.3.

4.1.1 Les colonnes et les cellules

L'environnement `tabular` reçoit un argument obligatoire qui décrit les colonnes. Chaque lettre `l`, `c` ou `r` définit une colonne dont le contenu est respectivement aligné à gauche, centré ou aligné à droite. Une barre verticale `|` ajoute un trait entre deux colonnes ou sur un bord du tableau, selon sa position dans cet argument. Par exemple, la déclaration suivante ouvre un tableau de quatre colonnes, la première alignée à gauche et les trois autres centrées :

```
\begin{tabular}{l|c|c|c}
```

Dans le contenu du tableau, le caractère `&` sépare deux cellules d'une même ligne et la commande `\\` termine la ligne. Une ligne de quatre cellules comporte ainsi trois séparateurs `&`, même lorsque certaines cellules sont vides. La commande `\hline` trace un trait horizontal sur toute la largeur du tableau. La commande `\cline{2-4}` limite ce trait aux colonnes désignées, ici de la deuxième à la quatrième. Leur présence est un choix de présentation, et non une condition de fonctionnement du tableau.

Le listing 4.1 présente un document complet qui compare les niveaux de structure de trois classes. La cellule vide au début de la première ligne laisse les intitulés des niveaux au-dessus des colonnes correspondantes. Les parties sont disponibles dans les trois classes, tandis que les chapitres sont absents de la classe `article`. La figure 4.1 reproduit le tableau composé.

Listing 4.1 – Un tableau de comparaison des classes.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

```
\usepackage[french]{babel}

\begin{document}
\begin{tabular}{l|c|c|c}
  \cline{2-4}
  & Parties & Chapitres & Sections\\
  \hline
  book & Oui & Oui & Oui\\
  report & Oui & Oui & Oui\\
  article & Oui & Non & Oui\\
  \hline
\end{tabular}
\end{document}
```

	Parties	Chapitres	Sections
book	Oui	Oui	Oui
report	Oui	Oui	Oui
article	Oui	Non	Oui

FIGURE 4.1 – Résultat du listing 4.1.

4.1.2 La largeur des colonnes

Avec les types `l`, `c` et `r`, la largeur d'une colonne dépend de son contenu. Une cellule longue ne se répartit pas automatiquement sur plusieurs lignes et peut rendre le tableau trop large pour la page. Le type `p{largeur}` définit au contraire une colonne dont les cellules sont composées comme des paragraphes de largeur fixée. Dans le listing 4.2, la deuxième colonne mesure cinq centimètres et le texte peut y être réparti sur plusieurs lignes.

Listing 4.2 – Une colonne de texte de largeur fixée.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\begin{document}
\begin{tabular}{l|p{5cm}}
  Élément & Description\\
  \hline
  Classe & Structure et présentation du document.\\
  Package & Ajout de fonctions pour le document.\\
\end{tabular}
\end{document}
```

La largeur totale comprend les colonnes, mais aussi les espaces entre les cellules et les traits éventuels. Une somme de largeurs de colonnes égale à la largeur du texte ne garantit donc pas que le tableau y tienne. L'environnement `tabular` compose par ailleurs un bloc qui ne se partage pas automatiquement entre plusieurs pages.

4.1.3 Les extensions pour les tableaux

Des packages complètent les possibilités de base pour des besoins plus précis. Le tableau 4.1 en présente quelques-uns. Ils ne sont pas nécessaires à la compilation des deux exemples précé-

Élément	Description
Classe	Structure et présentation du document.
Package	Ajout de fonctions pour le document.

FIGURE 4.2 – Résultat du listing 4.2 ; le texte de la seconde colonne occupe plusieurs lignes.

dents.

Package	Fonction principale
`array`	Enrichit la définition des colonnes et leur mise en forme.
`tabularx`	Propose des colonnes de type `X` dont la largeur s'ajuste à une largeur totale donnée.
`supertabular`	Permet des tableaux répartis sur plusieurs pages.
`multirow`	Permet à une cellule de s'étendre sur plusieurs lignes du tableau.

TABLE 4.1 – Quelques extensions pour la composition de tableaux.

Une cellule couvrant plusieurs colonnes relève de la commande standard `\multicolumn` ; ce besoin est distinct de celui traité par `multirow`. Le choix d'une extension dépend ainsi de la structure des données et des contraintes de présentation du tableau.

4.2 L'inclusion des images

Une illustration peut être conservée dans un fichier graphique séparé et insérée dans le document. Le package `graphicx` fournit la commande `\includegraphics` pour lire ce fichier et régler sa taille ou son orientation. Cette commande compose l'image à son point d'appel ; elle ne lui attribue pas à elle seule de numéro ou de légende.

4.2.1 Le fichier graphique et son chemin

L'argument obligatoire de `\includegraphics` est le chemin du fichier graphique. Son argument optionnel regroupe les options qui précisent les transformations appliquées à l'inclusion. Le listing 4.3 présente un document complet, accompagné d'une ressource graphique représentant la chaîne de compilation. Le chemin utilisé désigne le répertoire `images` voisin du répertoire de compilation : `..` représente le répertoire parent. Le résultat est reproduit dans la figure 4.3.

Listing 4.3 – Inclusion d'une image avec graphicx.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{graphicx}

\begin{document}
Le schéma représente la chaîne de compilation:
\begin{center}
\includegraphics[width=0.8\linewidth]
  {../images/chaine-compilation.pdf}
\end{center}
La largeur est adaptée à celle du texte.
```

```
\end{document}
```

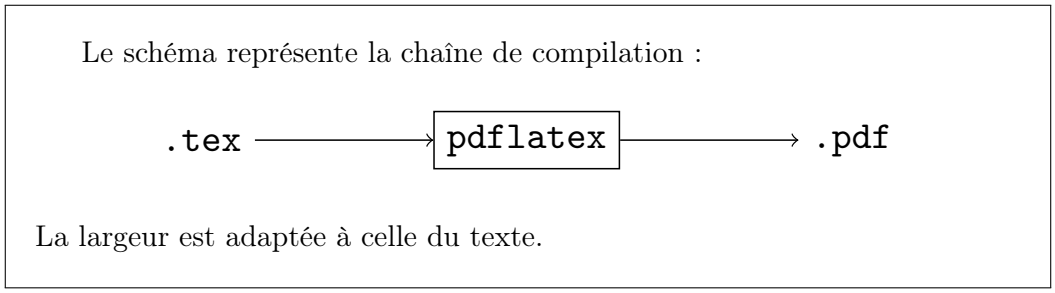


FIGURE 4.3 – Résultat du listing 4.3 ; le schéma est une image PDF incluse dans le document.

L'extension est indiquée explicitement dans cet exemple. Son omission est possible lorsque le logiciel trouve un format reconnu, mais la recherche dépend alors des extensions prises en charge et de la configuration. Le document dépend du fichier graphique : le texte du source ne suffit pas à reproduire une image externe qui serait absente.

4.2.2 Les dimensions et les transformations

L'option `width` fixe la largeur de l'image, par exemple avec une dimension comme `4cm` ou une fraction de la largeur disponible. La longueur `\linewidth` représente la largeur de ligne dans le contexte courant. Elle permet d'adapter l'image à la place disponible, y compris lorsque le contenu occupe une zone plus étroite que le texte principal.

Lorsque seule la largeur est fixée, les proportions sont conservées ; il en va de même lorsque seule la hauteur est fixée avec `height`. Fixer simultanément une largeur et une hauteur peut déformer l'image si ces dimensions ne respectent pas son rapport d'origine.

L'option `keepaspectratio` conserve les proportions à l'intérieur des deux dimensions demandées.

L'option `scale` applique un facteur de réduction ou d'agrandissement. L'option `angle` effectue une rotation, dont le point de référence peut être précisé avec `origin`. Enfin, l'option `page` sélectionne une page lorsqu'un fichier PDF en comporte plusieurs. Ces réglages concernent l'apparence de l'image incluse ; ils ne modifient pas son fichier d'origine.

4.2.3 Les formats matriciels et vectoriels

Une image matricielle est composée de pixels, comme une photographie en JPEG ou une capture d'écran en PNG. Un agrandissement important peut rendre ces pixels visibles ou diminuer la netteté perçue. Une image vectorielle décrit des formes et des tracés, qui conservent leur précision lors d'un changement d'échelle. Ce type de représentation convient particulièrement aux schémas et aux graphiques.

Avec `pdflatex`, les formats courants directement utilisables comprennent notamment PDF, PNG et JPEG. Un fichier PDF peut contenir du vectoriel, des images matricielles ou les deux : son extension ne garantit donc pas à elle seule la nature de son contenu. Le schéma de cet exemple est un dessin vectoriel exporté en PDF. Une photographie reste en revanche adaptée à un format matriciel ; sa conversion en PDF ne lui donne pas les propriétés d'un dessin vectoriel.

4.3 Les flottants, les légendes et les références

Les environnements `table` et `figure` regroupent un contenu, sa légende et son identification dans un *flottant*. Un flottant peut être déplacé par LaTeX pour s'intégrer à la composition des pages. Le premier environnement sert habituellement aux tableaux, le second aux illustrations.

Ils ne construisent pas leur contenu : un tableau reste décrit par `tabular` et une image externe reste insérée avec `\includegraphics`.

4.3.1 Le placement des flottants

L'argument optionnel de `table` ou de `figure` indique les emplacements autorisés pour le flottant. Les lettres courantes sont présentées dans le tableau 4.2.

Lettre	Emplacement autorisé
`h`	À proximité du point d'appel dans le texte.
`t`	En haut d'une page.
`b`	En bas d'une page.
`p`	Sur une page réservée aux flottants.

TABLE 4.2 – Principales options de placement des flottants.

L'écriture `\begin{figure}[htbp]` autorise ces différents emplacements. Elle ne signifie pas que la figure sera nécessairement composée à l'endroit exact de son appel. La place disponible et les règles de répartition du texte et des flottants interviennent aussi dans la décision. L'ordre des lettres ne définit pas un ordre de préférence personnalisé. Un flottant qui ne trouve pas de place autorisée peut rester en attente et apparaître plus loin dans le document.

Les références numérotées restent valides malgré ces déplacements. Elles permettent de désigner une figure ou un tableau sans dépendre d'une indication telle que « ci-dessous », qui peut devenir inexacte après une modification du texte.

4.3.2 La légende et l'étiquette

La commande `\caption` compose une légende et fait avancer le compteur correspondant au type de flottant. Elle figure généralement au-dessus d'un tableau et au-dessous d'une illustration. Cette position est une convention de présentation ; l'élément déterminant pour la référence est l'ordre entre la légende et l'étiquette. La commande `\label` est placée après `\caption` afin d'enregistrer le numéro qui vient d'être attribué.

Le préfixe de l'étiquette, par exemple `tab:` ou `fig:`, facilite sa lecture dans le source. Il ne détermine pas le type de l'élément ni sa numérotation. Les identifiants restent uniques dans l'ensemble du document, y compris lorsque les contenus sont répartis dans plusieurs fichiers.

Le listing 4.4 rassemble un tableau, une image et des renvois dans un même document. La déclaration `\centering` centre le contenu à l'intérieur de chaque flottant. Elle reste limitée à l'environnement concerné. Le tableau des classes et le schéma conservent des compteurs distincts.

Listing 4.4 – Flottants avec légendes, étiquettes et références.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{graphicx}

\begin{document}
\listoffigures
\listoftables
Le tableau~\ref{tab:classes} compare les classes.
La figure~\ref{fig:chaine}, page~\pageref{fig:chaine},
présente la compilation.

\begin{table}[htbp]
  \centering
```

```
  \caption{Classes et niveaux de structure.}
  \label{tab:classes}
  \begin{tabular}{l|c|c}
    Classe & Parties & Chapitres\\
    \hline
    book & Oui & Oui\\
    report & Oui & Oui\\
    article & Oui & Non\\
  \end{tabular}
\end{table}

\begin{figure}[htbp]
  \centering
  \includegraphics[width=0.6\linewidth]
    {../images/chaine-compilation.pdf}
  \caption{Chaîne de compilation.}
  \label{fig:chaine}
\end{figure}
\end{document}
```

Table des figures

1 Chaîne de compilation. 1

Liste des tableaux

1 Classes et niveaux de structure. 1

Le tableau 1 compare les classes. La figure 1, page 1, présente la compilation.

TABLE 1 – Classes et niveaux de structure.

Classe	Parties	Chapitres
book	Oui	Oui
report	Oui	Oui
article	Oui	Non

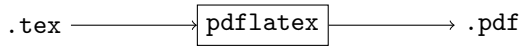


FIGURE 1 – Chaîne de compilation.

FIGURE 4.4 – Résultat du listing 4.4, après résolution des références et des listes.

4.3.3 Les références et les listes

La commande `\ref` produit le numéro associé à une étiquette, sans ajouter le mot « tableau » ou « figure ». La commande `\pageref` donne la page de l'élément référencé. Habituellement, une espace insécable `~` précède `\ref` et `\pageref` pour maintenir le numéro sur la même ligne que le mot qui l'introduit. Par exemple :

```
figure~\ref{fig:chaine}, page~\pageref{fig:chaine}
```

Le mot « figure » reste ainsi lié au numéro de la figure et le mot « page » au numéro de page.

La commande `\listoffigures` produit la liste des figures et `\listoftables` celle des tableaux, à l'endroit où elles apparaissent dans le corps du document. Les entrées proviennent des légendes `\caption` et indiquent les pages des éléments correspondants. Ces listes sont distinctes de la table des matières, qui est construite à partir des titres de sectionnement.

Les informations nécessaires sont conservées dans les fichiers auxiliaires, notamment ceux d'extensions `.lof` pour les figures et `.lot` pour les tableaux. Lors d'une première compilation, les légendes, les étiquettes et les pages sont enregistrées. Les compilations suivantes permettent de les réutiliser et de stabiliser les listes et les références visibles dans la figure 4.4.

Chapitre 5

Composer des mathématiques

Ce chapitre présente les modes mathématiques dans le texte et hors texte, puis les écritures des indices, exposants, fractions et sommes. Il décrit ensuite les équations numérotées ou non numérotées et les commandes permettant de leur associer une étiquette et d'y faire référence. Les exemples rapprochent le code des formules de leur résultat compilé.

5.1 Les modes mathématiques

LaTeX distingue le texte courant du contenu mathématique. Le passage en mode mathématique change l'interprétation de certains caractères, la forme des lettres et les règles d'espacement. Une expression peut être insérée dans une phrase ou composée séparément, selon la place qu'elle occupe dans le raisonnement.

5.1.1 Les formules dans le texte

Deux signes `$` délimitent une formule dans le texte : le premier ouvre le mode mathématique et le second le ferme. L'extrait suivant contient ainsi une expression mathématique au sein d'une phrase :

```
La valeur $x = 2$ est positive.
```

Le résultat est « La valeur $x = 2$ est positive. » La variable est composée en italique mathématique et les espaces autour du signe égal sont calculées par LaTeX. Le point final reste ici dans le texte courant, après la fermeture du mode mathématique. Une même lettre peut donc avoir un rôle et une apparence différents selon qu'elle appartient à une phrase ou à une formule.

5.1.2 Les formules hors texte

Une formule hors texte est détachée du paragraphe et dispose d'un espace vertical propre. Les commandes `\[` et `\]` délimitent une telle formule sans numéro. Elles ouvrent et ferment elles-mêmes le mode mathématique : le contenu n'est donc pas entouré en plus de signes `$`.

Le listing 5.1 présente un document complet dans lequel la même somme apparaît dans une phrase, puis hors texte. Le symbole n désigne un entier strictement positif et i l'indice de sommation, qui parcourt les entiers de 1 à n. La formule exprime la somme de ces entiers sous la forme d'une fraction. Dans la figure 5.1, la somme et la fraction sont plus développées hors texte, tandis que leur écriture dans la phrase reste plus compacte. Le sens de l'égalité est identique dans les deux cas.

Listing 5.1 – Une même formule dans le texte et hors texte.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
```

```
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\begin{document}
Dans une phrase, la somme s'écrit
$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

La même égalité hors texte:
\[
  \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.
\]
Les limites de la somme et la fraction changent de présentation.
\end{document}
```

Dans une phrase, la somme s'écrit $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.
La même égalité hors texte :

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Les limites de la somme et la fraction changent de présentation.

FIGURE 5.1 – Résultat du listing 5.1 ; le contexte détermine la présentation de la formule.

5.1.3 Les écritures usuelles

Dans une formule, le caractère `_` introduit un indice et le caractère `^` un exposant. Une paire d'accolades regroupe les éléments qui appartiennent à un même indice ou exposant. Ainsi, `x_{i+1}` place tout le contenu `i+1` en indice ; sans les accolades, seul le caractère immédiatement suivant `_` serait concerné. Les accolades organisent le source et ne sont pas imprimées.

La commande `\frac` reçoit deux arguments obligatoires, le numérateur et le dénominateur. La commande `\sum` produit le symbole de sommation, dont les bornes sont indiquées à l'aide d'un indice et d'un exposant. Le tableau 5.1 rapproche ces écritures de leur résultat. Les extraits de la première colonne constituent le contenu d'un mode mathématique et ne sont pas des documents autonomes.

Contenu du source mathématique	Résultat dans une ligne
`x_i`	x_i
`x_{i+1}`	x_{i+1}
`x^{n+1}`	x^{n+1}
`\frac{a}{b}`	$\frac{a}{b}$
`\sum_{i=1}^{n} i`	$\sum_{i=1}^{n} i$

TABLE 5.1 – Quelques constructions usuelles en mode mathématique.

Dans la somme du listing 5.1, l'indice `{i=1}` précise la borne inférieure et l'exposant `{n}` la borne supérieure. Le terme `i` qui suit ces bornes désigne la quantité additionnée. La commande

du symbole, ses bornes et ce terme ont donc des rôles distincts. LaTeX ne vérifie pas la validité mathématique de l'égalité : une formule peut être correctement composée tout en étant fausse ou incomplète.

Les espaces ordinaires du source mathématique ne servent pas à régler l'écartement des symboles. LaTeX distingue notamment les variables, les opérateurs et les relations pour calculer cet espacement. Ces règles sont différentes de celles du texte courant ; le mode mathématique sert à décrire une expression, pas à obtenir artificiellement du texte en italique.

5.2 Les équations et leurs références

5.2.1 Les équations numérotées

L'environnement `equation` compose une formule hors texte et lui attribue automatiquement un numéro. Il fournit déjà le mode mathématique : la formule s'écrit directement entre les commandes d'ouverture et de fermeture de l'environnement. Un numéro permet de désigner une équation dans le raisonnement sans recopier son contenu. Sa forme dépend de la classe : dans un article, les équations sont généralement numérotées à la suite, tandis qu'un rapport ou un livre inclut habituellement le numéro du chapitre.

Le listing 5.2 présente un document complet comportant une équation numérotée et une équation non numérotée. Le package `amsmath`, chargé dans le préambule, fournit notamment la forme étoilée `equation*` et la commande de référence `\eqref` utilisées dans cet exemple. L'environnement `equation` lui-même appartient à LaTeX.

Listing 5.2 – Équation numérotée, référence et équation sans numéro.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{amsmath}

\begin{document}
La somme des entiers de 1 à $n$ vérifie:
\begin{equation}
  \label{eq:somme}
  \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.
\end{equation}
L'équation~\eqref{eq:somme} est donnée page~\pageref{eq:somme}.

Un cas particulier sans numéro:
\begin{equation*}
  \sum_{i=1}^{3} i = 1+2+3 = 6.
\end{equation*}
\end{document}
```

5.2.2 Les équations non numérotées

L'environnement `equation*` compose une formule hors texte sans lui attribuer de numéro. Il convient à un calcul ponctuel ou à une égalité qui ne nécessite pas de référence numérotée. Dans l'exemple, la seconde formule applique la somme au cas de trois entiers et explicite son résultat. L'absence de numéro ne modifie ni son sens mathématique ni les règles de composition de ses symboles.

Les délimiteurs `\[` et `\]` rencontrés précédemment permettent également une formule hors texte sans numéro. Le tableau 5.2 récapitule les écritures utilisées dans ce chapitre.

La somme des entiers de 1 à n vérifie :

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \tag{1}$$

L'équation (1) est donnée page 1.

Un cas particulier sans numéro :

$$\sum_{i=1}^{3} i = 1 + 2 + 3 = 6.$$

FIGURE 5.2 – Résultat du listing 5.2 après résolution des références.

Écriture	Position	Numérotation
`$...$`	Dans le texte	Aucune
`\[...\]`	Hors texte	Aucune
`equation`	Hors texte	Automatique
`equation*` avec `amsmath`	Hors texte	Aucune

TABLE 5.2 – Modes mathématiques et numérotation.

5.2.3 Les étiquettes et les références croisées

La commande `\label` associe un identifiant à l'équation numérotée dans laquelle elle apparaît. Son argument est un nom choisi pour la désigner dans le source : le préfixe `eq:` indique ici qu'il s'agit d'une équation, mais ne participe pas au numéro imprimé. Chaque identifiant désigne un seul élément dans l'ensemble du document. Dans le listing 5.2, cette étiquette est placée à l'intérieur de l'environnement `equation`, immédiatement après son ouverture.

La commande `\eqref` reçoit cet identifiant et produit le numéro de l'équation entre parenthèses. La commande `\ref` donne le numéro sans ces parenthèses et `\pageref` donne la page de l'élément. Le texte du listing réutilise ainsi le numéro et la page calculés lors de la compilation. L'insertion d'une équation précédente ou une modification de pagination peuvent changer ces valeurs, sans modification de la phrase qui contient les références.

Le numéro est fourni par l'environnement, pas par l'étiquette. L'ajout d'un `\label` dans `equation*` ne transforme donc pas une formule non numérotée en équation référencée.

Comme pour la table des matières, les références reposent sur des informations conservées dans les fichiers auxiliaires. Une première compilation enregistre les étiquettes et leurs valeurs ; une compilation suivante les réutilise dans le texte. Après modification, les compilations successives permettent de stabiliser les numéros et les pages affichés dans la figure 5.2.

Chapitre 6

Organiser un document en plusieurs fichiers

Ce chapitre présente le passage d'un document contenu dans un seul fichier à un projet organisé autour d'un fichier maître et de fichiers inclus avec `\input`. Il développe ensuite un projet de rédaction à deux auteurs, de son arborescence et de sa compilation à la répartition des contributions et aux conventions communes.

6.1 Du fichier unique au fichier maître

6.1.1 Les limites d'un document monolithique

Un document est dit *monolithique* lorsque son préambule et tout son contenu sont regroupés dans un seul fichier source. Cette organisation convient à un texte court : les réglages, les titres et les paragraphes sont visibles au même endroit. Le listing 6.1 présente ainsi un compte rendu composé d'une introduction, d'une méthode et de résultats.

Lorsque le document s'allonge, le fichier unique rend les déplacements dans le source moins commodes et mêle des développements qui peuvent être rédigés séparément. La coécriture ajoute une difficulté : plusieurs auteurs peuvent modifier simultanément le même fichier, même lorsqu'ils travaillent sur des sections différentes. Le découpage en fichiers permet de donner une unité de rédaction propre à chaque partie du contenu. Il facilite aussi le repérage d'un passage et la réutilisation d'un ensemble cohérent de texte ou d'un schéma.

Ce découpage porte sur l'organisation des sources, pas sur la structure logique du document. Une section reste une section, qu'elle soit écrite dans le fichier principal ou dans un fichier séparé. La numérotation, la pagination et les références restent celles d'un document commun.

Listing 6.1 – Un rapport entièrement contenu dans un seul fichier.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\title{Compte rendu de mesure}
\author{Camille Martin \and Alex Bernard}
\date{}

\begin{document}
\maketitle
\section{Introduction}
Ce rapport présente une mesure et son analyse.

\section{Méthode}
```

```
\label{sec:rapport-methode}
La mesure est relevée avant son analyse.
\begin{center}
\fbox{Acquisition}\quad$\longrightarrow$\quad\fbox{Analyse}
\end{center}

\section{Résultats}
Les résultats sont interprétés selon la méthode
de la section~\ref{sec:rapport-methode}.
La mesure est stable pendant l'observation.
\end{document}
```

6.1.2 L'insertion du contenu d'un fichier

La commande `\input` reçoit le nom d'un fichier et fait lire son contenu à l'endroit de l'appel. Par exemple, l'instruction suivante insère le contenu d'un fichier de section :

```
\input{sections/introduction}
```

L'extension `.tex` peut être omise dans cet appel. Une fois le fichier inclus lu, la compilation reprend à la suite de la commande `\input` dans le fichier appelant. Cette insertion ne crée ni document indépendant ni saut de page automatique. L'ordre des appels détermine l'ordre de lecture des contenus ; il ne dépend pas de l'ordre alphabétique des noms de fichiers.

Le *fichier maître* est le point d'entrée de la compilation. Dans l'organisation retenue ici, il contient la classe, les packages, les informations du titre et l'environnement `document`. Les fichiers inclus contiennent les sections et leur développement, sans nouveau préambule ni environnement `document`. Les commandes et les packages chargés dans le préambule commun sont donc disponibles dans l'ensemble du texte assemblé.

Un fichier inclus ne constitue pas non plus un groupe. Une déclaration de mise en forme peut continuer à agir après la fin de ce fichier si sa portée n'a pas été limitée par un groupe ou un environnement approprié. De même, l'ouverture et la fermeture d'un environnement appartiennent à la structure du texte assemblé, et non aux limites physiques des fichiers. Des fichiers dont les groupes et les environnements sont équilibrés restent plus faciles à comprendre et à maintenir.

6.2 Un projet de document à deux auteurs

Cette section propose une structure de fichiers pour la rédaction d'un document à deux auteurs. L'exemple distingue le fichier maître, les sections rédigées séparément et les ressources associées, puis décrit leur assemblage et leur compilation dans un document commun. Il présente ensuite la répartition des contributions et les conventions qui assurent leur cohérence.

6.2.1 Un projet complet

Le projet est organisé dans le répertoire `RapportDeuxAuteurs`. Ce répertoire constitue la racine du projet ; il contient le fichier maître, un répertoire de sections et un répertoire de ressources graphiques, selon l'arborescence suivante. Les noms de fichiers sont explicités ici parce qu'ils déterminent les relations entre les éléments du projet.

```
RapportDeuxAuteurs/
+-- rapport.tex
+-- sections/
|   +-- introduction.tex
|   +-- methode.tex
|   +-- resultats.tex
+-- images/
```

```
    +-- schema.tex
```

Le listing 6.2 présente le fichier maître `rapport.tex`. Il conserve le préambule et le titre du document monolithique, puis remplace le corps des sections par trois appels à `\input`. Ces appels forment la table d'assemblage du document.

Listing 6.2 – Fichier maître rapport.tex.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\title{Compte rendu de mesure}
\author{Camille Martin \and Alex Bernard}
\date{}

\begin{document}
\maketitle
\input{sections/introduction}
\input{sections/methode}
\input{sections/resultats}
\end{document}
```

Le contenu de chaque section est donné dans les listings 6.3, 6.4 et 6.5. Ces fichiers sont des fragments du document : ils ne sont pas destinés à être compilés séparément. Le fichier de méthode contient à son tour un appel à `\input` pour insérer le schéma. Le découpage peut donc comporter plusieurs niveaux d'inclusion tout en conservant un seul point d'entrée.

Listing 6.3 – Fichier sections/introduction.tex.

```
\section{Introduction}
Ce rapport présente une mesure et son analyse.
```

Listing 6.4 – Fichier sections/methode.tex.

```
\section{Méthode}
\label{sec:rapport-methode}
La mesure est relevée avant son analyse.
\input{images/schema}
```

Listing 6.5 – Fichier sections/resultats.tex.

```
\section{Résultats}
Les résultats sont interprétés selon la méthode
de la section~\ref{sec:rapport-methode}.
La mesure est stable pendant l'observation.
```

Le schéma du listing 6.6 est lui aussi décrit en LaTeX. La commande `\fbox` encadre son argument, tandis que la flèche appartient au mode mathématique ; `\quad` ajoute une séparation entre ces éléments. Cette ressource ne demande aucun package supplémentaire. Son inclusion produit le dessin au milieu de la section de méthode.

Listing 6.6 – Ressource images/schema.tex.

```
\begin{center}
\fbox{Acquisition}\quad$\longrightarrow$\quad\fbox{Analyse}
\end{center}
```

Les listings donnent ainsi l'ensemble des sources nécessaires à l'exemple. Le résultat commun aux versions monolithique et modulaire est reproduit dans la figure 6.1. L'insertion des sections et du schéma ne change ni l'apparence du texte ni sa numérotation.

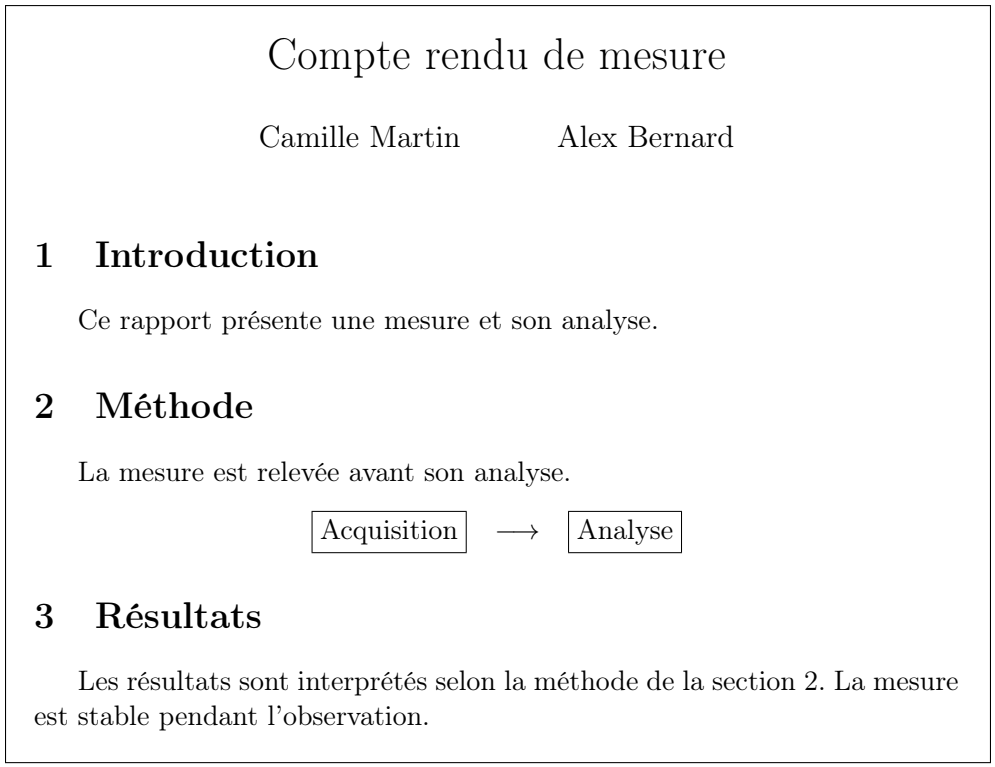


FIGURE 6.1 – Rapport assemblé à partir du fichier maître et de ses fichiers inclus.

6.2.2 Les chemins relatifs

Un chemin relatif décrit la position d'un fichier à partir d'un répertoire de référence. Dans cet exemple, la compilation est lancée depuis la racine `RapportDeuxAuteurs`. Les chemins `sections/introduction` et `images/schema` sont alors interprétés depuis ce répertoire de travail, y compris lorsqu'un appel figure dans un fichier inclus. L'ouverture de `sections/methode.tex` ne déplace pas le répertoire de travail dans `sections`.

L'appel au schéma reste donc `\input{images/schema}` dans le fichier de méthode. Il ne s'écrit pas avec un chemin remontant d'abord au répertoire parent, comme si le fichier de méthode était lui-même le point de départ de la compilation. Cette convention commune rend les chemins prévisibles dans tout le projet. Elle s'applique également aux ressources externes : avec `graphicx`, une image placée à la racine dans `images` serait désignée par un chemin tel que `images/schema.pdf`.

Les chemins relatifs permettent de déplacer le répertoire complet sans y inscrire un chemin propre à l'ordinateur d'un auteur. Les noms, leur casse et l'organisation des sous-répertoires font en revanche partie du projet et doivent correspondre aux appels du source.

6.2.3 Une compilation globale

La compilation porte sur le fichier maître. Depuis la racine du projet, elle s'écrit :

```
$ pdflatex rapport.tex
$ pdflatex rapport.tex
```

La première compilation lit les trois sections dans l'ordre fixé par le maître et enregistre les informations de référence. La suivante permet de résoudre le renvoi de la section de résultats vers celle de méthode. Un seul PDF est produit pour le rapport ; les sections ne possèdent pas chacune une pagination indépendante. Les fichiers auxiliaires associés à la compilation portent ici le nom du fichier maître.

Une modification de la méthode ou du schéma est prise en compte lors de la compilation suivante de ce même maître. Le document reste cependant dépendant de tous les fichiers inclus : le fichier maître seul ne contient plus l'ensemble du texte. La conservation et le partage du projet portent donc sur l'arborescence complète.

6.2.4 La répartition des contributions

Camille Martin et Alex Bernard rédigent ensemble le rapport de l'exemple. Camille prend en charge l'introduction, la méthode et le schéma associé ; Alex rédige les résultats. Le tableau 6.1 relie ces responsabilités aux fichiers concernés. Chacun travaille ainsi sur une unité identifiée, tandis que le fichier maître fixe la présentation et l'ordre du document commun.

Contribution	Fichiers	Responsabilité
Contexte	`sections/introduction.tex` `sections/methode.tex`	Camille
Méthode	`images/schema.tex`	Camille
Résultats	`sections/resultats.tex`	Alex
Assemblage	`rapport.tex`	Camille

TABLE 6.1 – Répartition de la rédaction dans le projet commun.

Au début de la rédaction, les deux auteurs disposent de la même arborescence et du même préambule. Camille décrit la méthode et lui attribue une étiquette ; Alex peut utiliser cette étiquette dans son fichier de résultats sans recopier la méthode ni saisir son numéro. Lors de l'intégration, les fichiers relus occupent les emplacements prévus dans le projet commun. La compilation du maître rassemble les contributions et calcule leurs numéros ainsi que leurs références.

Si Camille complète ensuite la méthode, la contribution d'Alex reste dans son fichier propre. Le résultat commun peut néanmoins changer de pagination, ce que la compilation globale prend en compte. La séparation des fichiers facilite donc la répartition du travail sans rendre les parties indépendantes.

6.2.5 Les conventions communes

Le préambule unique évite que les auteurs chargent séparément des configurations divergentes. Un package nécessaire à une contribution est ajouté au préambule commun, afin que la compilation du maître fournisse les mêmes possibilités à tous les fichiers. La classe, les règles linguistiques et les commandes partagées restent ainsi définies à un seul endroit.

Les étiquettes appartiennent au document entier. Deux fichiers différents ne peuvent donc pas attribuer le même identifiant à deux éléments distincts sans rendre les références ambiguës. Dans l'exemple, l'étiquette de la méthode est réutilisée par la commande `\ref` du fichier de résultats : la frontière entre les fichiers ne limite pas la portée de cette référence.

Les auteurs partagent également une convention de noms, un encodage UTF-8 et la même racine de compilation. Un changement de nom de fichier ou de répertoire implique la mise à jour des chemins concernés. La cohérence de la notation, du vocabulaire et des transitions entre sections relève enfin d'une relecture commune du PDF assemblé. L'organisation modulaire facilite la maintenance et la corédaction, tandis que cette relecture assure la continuité du document final.

Chapitre 7

Découvrir les usages spécialisés

Les traitements de texte WYSIWYG ont une vocation généraliste : leurs fonctions répondent à des besoins communs de rédaction et de mise en forme. LaTeX est un langage de programmation extensible, dont les packages peuvent apporter des fonctionnalités propres à un métier. Ces packages métiers facilitent la description des objets d'un domaine et améliorent leur présentation en prenant en charge des conventions spécialisées. Ils permettent, par exemple, la composition de partitions musicales, de structures chimiques ou de revues de jeux présentant des parties d'échecs, des grilles de sudoku ou des parties de go.

Ce chapitre présente des packages utiles aux informaticiens. Il décrit l'inclusion de code source et la composition de pseudocode, puis les dessins avec TikZ. La dernière section expose l'interaction avec Python et la chaîne de compilation nécessaire à l'insertion de ses résultats dans un document.

7.1 Le code source et le pseudocode

La présentation d'un programme ou d'un algorithme associe un contenu à des conventions de lecture : mots-clés, indentation et organisation des instructions. Des packages prennent en charge cette présentation tout en conservant une description textuelle dans le source du document. Leur rôle dépend du contenu : un listing reproduit du code, tandis que le pseudocode décrit les étapes d'un algorithme avec une notation propre.

7.1.1 L'inclusion d'un programme avec listings

Le package `listings` fournit l'environnement `lstlisting` pour du code écrit directement dans le source et la commande `\lstinputlisting` pour lire un fichier séparé. L'argument obligatoire de cette commande est le chemin du fichier. L'option `language=C` sélectionne les règles de présentation du langage C, notamment la reconnaissance des mots-clés. Dans le listing 7.1, `basicstyle` fixe une police à chasse fixe, `breaklines=true` permet de couper visuellement les lignes longues et `showstringspaces=false` évite de matérialiser les espaces des chaînes de caractères. Ces réglages modifient le rendu dans le PDF, sans modifier le fichier de code.

Listing 7.1 – Document incluant un fichier de code C.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{listings}

\begin{document}
\lstinputlisting[language=C, basicstyle=\small\ttfamily,
  breaklines=true, showstringspaces=false]{triParTas.c}
```

```
\end{document}
```

```
#include"triParTas.h"
#include"tas.h"
#include"echanger.h"

void trierEntiersParTas(long int tab[], long int
    nbElements) {
  long int i;
  tamiser(tab,nbElements);
  for(i=nbElements-1;i>0;i--) {
    echangerEntiers(&tab[0],&tab[i]);
    faireDescendre(tab,i,0);
  }
}
```

FIGURE 7.1 – Résultat du listing 7.1 : le contenu du fichier C est mis en forme dans le PDF.

Le fragment C visible dans la figure 7.1 dépend de fonctions et de fichiers d'en-tête qui ne figurent pas dans cet exemple ; il ne constitue pas un programme C autonome. La compilation du document LaTeX lit ce fragment comme du texte à afficher et ne l'exécute pas. L'encodage des fichiers inclus demande également une configuration adaptée lorsqu'ils contiennent des caractères accentués. L'extrait C utilisé ici ne contient que des caractères ASCII.

Pour les fichiers UTF-8, le package `listingsutf8` complète la commande d'inclusion. Il convertit le contenu vers un encodage à un octet avant sa lecture. L'option `inputencoding`, avec la valeur `utf8/latin1`, convient ainsi lorsque tous les caractères du fichier existent dans l'encodage Latin-1. Ce mécanisme concerne uniquement les fichiers lus par `\lstinputlisting`. Il ne s'applique pas au code écrit directement dans un environnement `lstlisting`.

7.1.2 La description d'une procédure en pseudocode

Le package `pseudocode` développé à l'INSA Rouen Normandie [1] fournit des commandes françaises pour décrire des algorithmes. Le listing 7.2 utilise son environnement `algorithme` et sa commande `\procedure` pour représenter un échange de valeurs. Le package doit être disponible lors de la compilation ; les commandes présentées sont celles de cette implémentation.

La commande `\procedure` reçoit cinq arguments : le nom, les paramètres formels, les préconditions, les variables locales et le corps de la procédure. Dans cet exemple, le troisième argument est vide, car aucune précondition n'est indiquée. La commande `\paramEntreeSortie` signale que les deux paramètres peuvent être modifiés, et `\affecter` décrit une affectation avec sa destination et sa valeur.

Listing 7.2 – Une procédure d'échange avec le package pseudocode du cours.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{pseudocode}

\begin{document}
\begin{algorithme}
\procedure{echanger}
```

1. https://gitlab.insa-rouen.fr/lbrebant/pseudocode

```
  {\paramEntreeSortie{a,b: \entier}}
  {}
  {tmp: \entier}
  {\affecter{tmp}{a}
   \affecter{a}{b}
   \affecter{b}{tmp}}
\end{algorithme}
\end{document}
```

procédure echanger **(E/S** a,b : **Entier)**
  Déclaration tmp : **Entier**
debut
  tmp ← a
  a ← b
  b ← tmp
fin

FIGURE 7.2 – Résultat du listing 7.2 : la présentation distingue les paramètres, la variable locale et les instructions.

La variable temporaire conserve la valeur initiale du premier paramètre avant son remplacement. Les commandes produisent les mots-clés, les flèches et les retraits visibles dans la figure 7.2.

7.2 Les dessins avec TikZ

Le package `tikz` permet de décrire un dessin par des nœuds, des coordonnées et des tracés. La compilation construit le dessin à partir de ces instructions. Le résultat peut ainsi rester vectoriel, avec du texte et des formes dont la présentation est définie dans le source.

7.2.1 Les nœuds et les flèches

L'environnement `tikzpicture` délimite un dessin. Dans le listing 7.3, chaque commande `\node` crée un nœud, lui associe un nom entre parenthèses et place son contenu entre accolades. L'instruction `at (3,0)` fixe une position dans le repère du dessin, dont l'unité usuelle est le centimètre. Les options `draw,rectangle` ajoutent un cadre autour du nœud central.

La commande `\draw` relie ensuite les nœuds nommés. Le symbole `-` indique un segment et l'option `->` lui donne une pointe de flèche. Chaque instruction TikZ se termine par un point-virgule. La figure 7.3 montre le schéma produit par ces instructions.

Listing 7.3 – Description d'un schéma de compilation en TikZ.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{tikz}

\begin{document}
\begin{tikzpicture}
  \node (source) {Code \LaTeX{}};
```

```
  \node[draw,rectangle] (moteur) at (3,0) {\texttt{pdflatex}};
  \node (resultat) at (6,0) {PDF};
  \draw[->] (source) -- (moteur);
  \draw[->] (moteur) -- (resultat);
\end{tikzpicture}
\end{document}
```

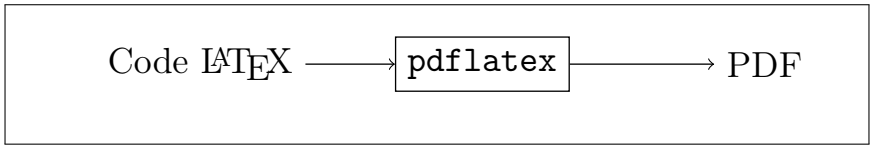


FIGURE 7.3 – Résultat du listing 7.3 ; les noms des nœuds servent à définir les connexions.

7.2.2 Les formes et les couleurs

TikZ peut aussi construire des formes à partir de coordonnées et de règles de remplissage. Le listing 7.4 définit cinq centres avec `\coordinate`, puis utilise `\fill` pour dessiner des anneaux colorés. Pour chaque centre, les deux cercles et la règle `even odd rule` laissent vide la région intérieure et remplissent la couronne. Le tracé des formes et leur couleur sont ainsi décrits séparément de leur position.

Listing 7.4 – Coordonnées et remplissage de cinq anneaux.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{tikz}

\begin{document}
\begin{tikzpicture}[scale=0.3]
  \coordinate (A) at (-6.4,0);
  \coordinate (B) at (-3.2,-3.2);
  \coordinate (C) at (0,0);
  \coordinate (D) at (3.2,-3.2);
  \coordinate (E) at (6.4,0);
  \fill[blue,even odd rule] (A) circle(2.5) circle(3);
  \fill[yellow,even odd rule] (B) circle(2.5) circle(3);
  \fill[black,even odd rule] (C) circle(2.5) circle(3);
  \fill[green,even odd rule] (D) circle(2.5) circle(3);
  \fill[red,even odd rule] (E) circle(2.5) circle(3);
\end{tikzpicture}
\end{document}
```

L'option `scale=0.3` réduit les dimensions géométriques du dessin. Dans ces deux exemples, la description TikZ appartient directement au document LaTeX ; elle peut également être conservée dans un fichier séparé pour faciliter sa réutilisation.

7.3 L'interaction avec Python

Le package `pythontex` permet d'exécuter du code Python associé à un document et d'intégrer ses résultats à la composition. Le document peut ainsi utiliser une valeur calculée tout en conservant le calcul dans son source. Cette possibilité ajoute une étape d'exécution à la chaîne de compilation.

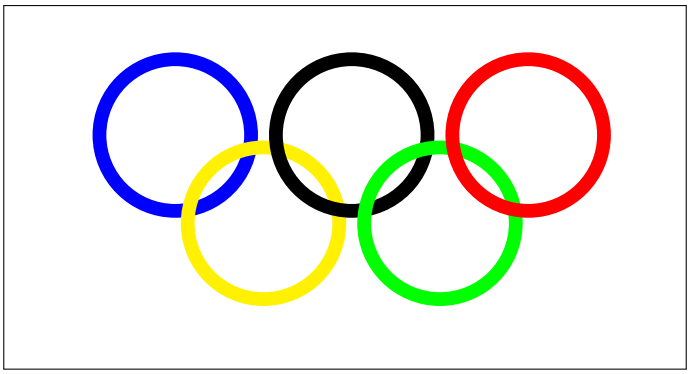


FIGURE 7.4 – Résultat du listing 7.4 : les anneaux sont obtenus par remplissage de couronnes.

7.3.1 Le calcul et son résultat dans le texte

Dans le listing 7.5, l'environnement `pycode` contient une fonction qui ajoute une unité à son argument. L'instruction `x = foo(0)` affecte donc la valeur calculée, ici `1`, à la variable Python `x`. La fonction `print` produit une ligne de texte intégrée au document ; le bloc de code lui-même n'est pas affiché dans le résultat. La commande `\py{x}` évalue ensuite une expression Python et insère sa valeur dans une phrase LaTeX.

Listing 7.5 – Calcul Python et réutilisation de sa valeur.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{pythontex}

\begin{document}
\begin{pycode}
def foo(x):
    return x + 1
x = foo(0)
print(f'x vaut {x} (depuis le code Python).')
\end{pycode}

Depuis le code \LaTeX{}, x vaut \py{x}.
\end{document}
```

x vaut 1 (depuis le code Python).
Depuis le code LaTeX, x vaut 1.

FIGURE 7.5 – Résultat du listing 7.5, après exécution du code Python.

La figure 7.5 montre les deux utilisations de la même valeur : sa production par `print` et son insertion par `\py`. La variable appartient à la session Python de l'exemple ; elle n'est pas une commande LaTeX.

7.3.2 La chaîne de compilation et ses dépendances

Cet exemple nécessite le package `pythontex`, son programme de traitement et un interpréteur Python 3 accessible depuis le terminal. Pour un document nommé `calcul.tex`, la chaîne s'écrit :

```
$ pdflatex calcul.tex
$ pythontex --interpreter python:python3 calcul.tex
$ pdflatex calcul.tex
```

La première compilation extrait les informations nécessaires au traitement dans un fichier auxiliaire. Le programme `pythontex` exécute le code et enregistre les résultats. La compilation suivante réutilise ces résultats pour produire le PDF final.

L'option `--interpreter` indique ici la commande choisie pour lancer Python. Une modification du calcul est prise en compte lorsque cette chaîne est de nouveau exécutée.

L'étape Python exécute réellement les instructions avec les droits du compte qui lance le traitement. Le contenu du programme et les bibliothèques qu'il utilise font donc partie des dépendances à connaître pour compiler le document. L'exemple présenté utilise uniquement les fonctions de base de Python ; un calcul faisant appel à d'autres bibliothèques demanderait leur installation dans l'environnement Python sélectionné.

Chapitre 8

Personnaliser et poursuivre son apprentissage

Ce chapitre présente la définition de commandes et d'environnements personnalisés, puis leur redéfinition. Il décrit ensuite quelques packages utiles à la présentation des pages, aux notations mathématiques et aux liens hypertextes. La dernière section présente la consultation de la documentation depuis la ligne de commande et les ressources disponibles en ligne.

8.1 Définir ses commandes et ses environnements

Une commande personnalisée associe un nom à un contenu ou à un ensemble d'instructions. Elle peut éviter la répétition d'une expression, mais aussi donner un nom à un rôle du texte, indépendamment de son apparence. Par exemple, une commande désignant un nom de fichier permet de choisir une présentation commune à toutes ses occurrences. Cette définition rassemble le choix de présentation en un seul endroit, au lieu de le répéter dans chaque phrase.

8.1.1 Une commande sans argument

La commande `\newcommand` reçoit le nom de la commande à créer, puis sa définition. Ces deux arguments sont entourés d'accolades. Dans le listing 8.1, la définition suivante associe un nom à l'expression « Analyse de mesures » :

```
\newcommand{\nomProjet}{Analyse de mesures}
```

La définition est placée dans le préambule, ce qui la rend disponible dans l'ensemble du corps du document. Lors de la compilation, chaque appel à `\nomProjet` produit le contenu défini. Les accolades vides de `\nomProjet{}` délimitent le nom de la commande et préservent l'espace qui suit dans une phrase. Un changement du nom du projet dans sa définition se répercute ainsi sur tous les appels, sans modification des phrases qui les contiennent.

8.1.2 Une commande avec un argument

Une définition peut également recevoir des arguments. Le nombre d'arguments est indiqué entre crochets après le nom de la nouvelle commande. Dans sa définition, `#1` désigne le premier argument, `#2` le deuxième, et ainsi de suite, jusqu'à neuf arguments avec cette syntaxe. Dans le listing 8.1, la commande `\fichier` reçoit un argument :

```
\newcommand{\fichier}[1]{\texttt{#1}}
```

L'appel `\fichier{mesures.csv}` transmet le texte `mesures.csv` à la définition. Ce texte prend la place de `#1` et devient donc l'argument de `\texttt`. Le résultat est le nom du fichier

composé dans une police à chasse fixe, comme le montre la figure 8.1. Le nom `\fichier` exprime le rôle du contenu, tandis que sa définition détermine son apparence. Cette commande ne lit pas le fichier mentionné : elle compose seulement son nom.

Listing 8.1 – Définition et utilisation de deux commandes personnelles dans un document complet.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\newcommand{\nomProjet}{Analyse de mesures}
\newcommand{\fichier}[1]{\texttt{#1}}

\begin{document}
Le projet \nomProjet{} utilise \fichier{mesures.csv}.

\renewcommand{\fichier}[1]{\textbf{#1}}
Le projet \nomProjet{} produit \fichier{resultats.csv}.
\end{document}
```

Le projet Analyse de mesures utilise `mesures.csv`.
Le projet Analyse de mesures produit **resultats.csv**.

FIGURE 8.1 – Résultat du listing 8.1.

8.1.3 Un environnement personnalisé

Un environnement personnalisé rassemble les instructions qui entourent un contenu. La commande `\newenvironment` reçoit trois arguments obligatoires : le nom de l'environnement, les instructions exécutées à son ouverture et les instructions exécutées à sa fermeture. Le nom s'écrit sans barre oblique inverse, comme dans les appels à `\begin` et à `\end`.

Le listing 8.2 définit un environnement nommé `remarqueCours`. À son ouverture, il commence un environnement `quote`, qui décale le texte par rapport aux marges du texte courant, puis réduit la taille des caractères avec `\small`. Le préfixe « Remarque. » est composé en gras avant le contenu de la remarque. À la fermeture, `\end{quote}` termine le bloc.

L'environnement constitue un groupe : la déclaration `\small` reste locale à la remarque. La phrase placée après le bloc retrouve donc la taille normale, comme le montre la figure 8.2. Une telle définition assure la même présentation à toutes les remarques, sans répéter leurs réglages dans le corps du document. Elle ne crée pas de compteur ni de numérotation automatique.

Listing 8.2 – Définition puis redéfinition d'un environnement dans un document complet.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}

\newenvironment{remarqueCours}
  {\begin{quote}\small\textbf{Remarque.} }
  {\end{quote}}

\begin{document}
```

```
\begin{remarqueCours}
Les mesures sont exprimées en secondes.
\end{remarqueCours}
Le texte courant conserve sa taille normale.

\renewenvironment{remarqueCours}
  {\begin{quote}\small\textbf{Précision.} \itshape}
  {\end{quote}}
\begin{remarqueCours}
La précision de la mesure dépend du capteur.
\end{remarqueCours}
Le texte courant retrouve sa forme droite.
\end{document}
```

Remarque. Les mesures sont exprimées en secondes.

Le texte courant conserve sa taille normale.

Précision. *La précision de la mesure dépend du capteur.*

Le texte courant retrouve sa forme droite.

FIGURE 8.2 – Résultat du listing 8.2 : deux présentations successives d'un même environnement.

8.1.4 La redéfinition et sa portée

La commande `\newcommand` signale une erreur si le nom demandé est déjà défini. Ce contrôle évite de remplacer involontairement une commande fournie par LaTeX, par la classe ou par un package. À l'inverse, `\renewcommand` remplace la définition d'une commande qui existe déjà et signale une erreur si elle n'existe pas. Sa syntaxe est celle de `\newcommand`.

Dans le listing 8.1, la redéfinition de `\fichier` conserve son argument, mais le compose en gras. Les appels suivants utilisent cette nouvelle définition ; le premier appel, déjà composé, reste en chasse fixe. Pour fixer une présentation commune à tout un document, la définition retenue figure habituellement dans le préambule. Son remplacement au milieu du corps sert ici à rendre son effet visible dans un exemple court.

La commande `\renewenvironment` joue le même rôle pour un environnement existant. Dans le listing 8.2, elle remplace le préfixe « Remarque. » par « Précision. ». La déclaration `\itshape` applique l'italique au contenu du second bloc, avec une portée limitée à l'environnement. Les instructions d'ouverture et de fermeture sont toutes deux données dans cette nouvelle définition. La commande `\newenvironment` refuse un nom d'environnement déjà utilisé, tandis que `\renewenvironment` suppose que l'environnement existe.

La redéfinition d'une commande ou d'un environnement fourni par une classe ou un package peut modifier d'autres éléments qui l'utilisent. Les options et les commandes de configuration prévues par leur documentation permettent souvent d'obtenir le résultat souhaité sans remplacer leur définition. Les exemples de cette section se limitent à la redéfinition des commandes et environnements propres au document.

8.2 Quelques packages complémentaires

Les packages complètent les fonctions de la classe sans changer la nature du document. Certains apportent de nouveaux objets, comme les dessins du chapitre 7, tandis que d'autres règlent

des aspects transversaux de la présentation. Le listing 8.3 associe plusieurs de ces fonctions dans un document complet.

8.2.1 Les marges, les en-têtes et les pieds de page

Le package `geometry` décrit les dimensions de la page et de sa zone de composition à l'aide d'options. Dans l'exemple, `margin=2.5cm` fixe les marges à `2.5cm`. L'option `includeheadfoot` fait entrer l'en-tête et le pied de page dans la zone délimitée par ces marges. La hauteur de l'en-tête est fixée à `14pt` par l'option `headheight=14pt`. Ces réglages expriment les dimensions souhaitées sans modifier séparément les différentes longueurs qui déterminent la page.

Le package `fancyhdr` configure le contenu des en-têtes et des pieds de page. La commande `\pagestyle{fancy}` sélectionne le style qu'il fournit, et `\fancyhf{}` vide ses champs avant leur configuration. Dans l'exemple, `\fancyhead[L]` place le nom du projet à gauche de l'en-tête ; `\fancyfoot[C]` centre le numéro dans le pied de page. La commande `\thepage` produit la représentation du numéro de la page courante. Les lettres `L`, `C` et `R` désignent respectivement les positions gauche, centrale et droite.

La figure 8.3 reproduit séparément le haut et le bas de la page produite. Dans un rapport ou un livre, certaines pages, notamment les ouvertures de chapitre, utilisent un style particulier, généralement `plain` avec les classes standard. La configuration du style `fancy` ne modifie pas automatiquement ces autres styles.

Listing 8.3 – Configuration des pages, symboles mathématiques et liens dans un document complet.

```
\documentclass[a4paper,11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage[margin=2.5cm,includeheadfoot,
            headheight=14pt]{geometry}
\usepackage{fancyhdr}
\usepackage{amssymb}
\usepackage[hidelinks]{hyperref}

\pagestyle{fancy}
\fancyhf{}
\fancyhead[L]{Analyse de mesures}
\fancyfoot[C]{\thepage}

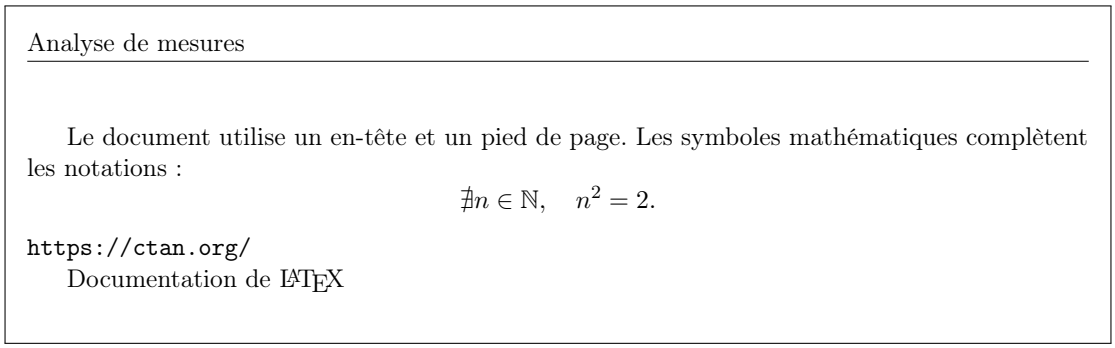
\begin{document}
Le document utilise un en-tête et un pied de page.
Les symboles mathématiques complètent les notations:
\[
  \nexists n \in \mathbb{N}, \quad n^2 = 2.
\]
\url{https://ctan.org/}

\href{https://www.latex-project.org/help/documentation/}
     {Documentation de \LaTeX{}}
\end{document}
```

8.2.2 Les polices et les symboles mathématiques

Les packages `amsfonts` et `amssymb` complètent les possibilités mathématiques décrites au chapitre 5. Le premier donne notamment accès à l'alphabet ajouré avec `\mathbb`, souvent utilisé pour les ensembles de nombres. Le second charge `amsfonts` et définit des commandes de symboles

Haut de la page



Pied de la même page



FIGURE 8.3 – Deux extraits du résultat compilé du listing 8.3.

supplémentaires : son chargement suffit donc dans l'exemple. Dans la formule du listing 8.3, `\mathbb{N}` compose le symbole de l'ensemble des entiers naturels et `\nexists` le symbole signifiant « il n'existe pas ». La formule indique qu'aucun entier naturel n'a pour carré la valeur deux. Ces commandes sont utilisées en mode mathématique, et non directement dans le texte courant.

8.2.3 Les adresses et les liens hypertextes

Le package `url` fournit la commande `\url` pour composer une adresse en conservant ses caractères particuliers et en autorisant des coupures à des emplacements adaptés. Il ne rend pas, à lui seul, les adresses cliquables. Le package `hyperref` ajoute des liens hypertextes aux références du document et aux adresses composées avec `\url`. Il charge lui-même `url` : leur chargement simultané n'est donc pas nécessaire dans le listing 8.3.

La commande `\href`, fournie par `hyperref`, reçoit deux arguments : l'adresse de destination, puis le texte visible du lien. L'exemple affiche ainsi le texte « Documentation de LaTeX » tout en lui associant l'adresse du site officiel. L'option `hidelinks` conserve les liens actifs sans leur ajouter de cadre ni de couleur particulière. Dans cet exemple, `hyperref` est chargé après les autres packages ; sa documentation précise les éventuelles exceptions à cet ordre selon les packages utilisés.

8.3 La documentation et les ressources en ligne

La documentation d'un package décrit les commandes qu'il fournit, leurs arguments et ses options de configuration. Elle précise également ses dépendances et ses interactions avec les autres packages. Ces informations permettent de distinguer une fonctionnalité standard de LaTeX d'une extension qui nécessite un chargement dans le préambule.

8.3.1 La consultation depuis la ligne de commande

L'utilitaire `texdoc`, fourni notamment avec TeX Live, recherche la documentation présente dans la distribution installée. La commande suivante recherche celle du package `geometry` et ouvre le document retenu dans un logiciel de lecture :

```
$ texdoc geometry
```

Le nom du package constitue ici le terme de recherche. La même commande accepte par exemple `fancyhdr`, `listings` ou `hyperref`. La présence de leur documentation dépend des composants installés avec la distribution. L'option `-l` affiche la liste des documents trouvés au lieu d'ouvrir directement le résultat :

```
$ texdoc -l geometry
```

La documentation locale accompagne la version installée du package. Cette correspondance est utile lorsqu'un exemple trouvé en ligne emploie une option ajoutée dans une version plus récente.

8.3.2 CTAN, la FAQ francophone et les guides de référence

CTAN, pour *Comprehensive TeX Archive Network*, rassemble des packages, des classes, leurs sources et leurs documentations. Le catalogue est accessible à l'adresse `https://ctan.org/`. Une notice, comme celle de `geometry` à l'adresse `https://ctan.org/pkg/geometry`, indique le rôle du package et donne accès à son manuel. La version proposée par CTAN peut différer de celle de la distribution installée.

La FAQ LaTeX francophone de l'association GUTenberg est accessible à l'adresse `https://faq.gutenberg-asso.fr/`. Elle rassemble des réponses organisées par thèmes et complète les manuels par des explications en français sur les usages courants. Le site du projet LaTeX réunit quant à lui les guides de référence et les liens vers des ressources d'apprentissage à l'adresse `https://www.latex-project.org/help/documentation/`.

L'interprétation d'un exemple dépend de sa classe, de ses packages et de leurs options. Un extrait limité au corps du document peut donc omettre une dépendance indispensable à sa compilation. Un document minimal complet, comprenant le préambule et le passage concerné, permet d'isoler le rôle de ces dépendances et de rapprocher le résultat observé de celui décrit dans la documentation.

Conclusion générale

LaTeX est un système de traitement de texte fondé sur l'approche WYMIWYG : l'auteur décrit le sens et l'organisation du document pour se concentrer sur le fond. La classe définit sa structure et les règles générales de sa présentation, tandis que le package `babel`, configuré pour la langue du texte, prend en charge des conventions typographiques propres à cette langue. Cette séparation permet de conserver une présentation cohérente sans multiplier les réglages locaux au fil de la rédaction.

LaTeX est aussi un langage de programmation : le document source respecte une syntaxe et met en œuvre des commandes et des environnements. Les définitions personnelles permettent de regrouper des instructions récurrentes, et les packages étendent le langage pour répondre à des besoins généraux ou propres à un métier. La rédaction d'un document repose ainsi sur un ensemble de règles explicites et réutilisables.

L'apprentissage de cette approche demande un investissement initial qui peut sembler être une perte de temps. La compréhension de la syntaxe, du rôle du préambule et de la compilation précède en effet la maîtrise de l'outil. Cet investissement permet ensuite de gagner du temps : la mise en forme commune, la numérotation, les références croisées et la réutilisation des définitions limitent les reprises manuelles lors des modifications. Le bénéfice devient particulièrement sensible lorsque le document s'enrichit ou que plusieurs auteurs contribuent à sa rédaction.

Ce polycopié constitue une introduction au langage LaTeX. Les notions présentées donnent néanmoins déjà aux étudiants du département ISIA les bases nécessaires à la rédaction des rapports de projets intégratifs, de projets INSA certifiés et des différents stages. La structuration du texte, la présentation des tableaux, des figures et des mathématiques, ainsi que l'organisation en plusieurs fichiers répondent à des besoins courants de ces documents. La qualité d'un rapport repose également sur la clarté de ses objectifs, la justification des choix et l'analyse des résultats. Les figures et les tableaux contribuent à cette démonstration lorsqu'ils sont introduits et commentés dans le texte ; la relecture du PDF permet d'en vérifier la lisibilité.

Deux ouvrages de référence constituent des lectures essentielles pour approfondir cet apprentissage. *La maîtrise de TeX et LaTeX*, de Thomas Lachand-Robert, publié chez Masson en 1995, permet notamment d'approfondir le fonctionnement de TeX et la programmation des macros. *The LaTeX Companion*, de Frank Mittelbach avec Ulrike Fischer, développe l'utilisation de LaTeX et de ses extensions dans sa troisième édition, publiée en deux volumes chez Addison-Wesley Professional en 2023 et proposée en version brochée depuis 2025.

Au-delà d'un plus grand nombre de commandes et d'explications plus détaillées de celles abordées ici, ces lectures ouvrent l'accès à d'autres dimensions de la production documentaire : la gestion d'une bibliographie, des abréviations et, plus largement, la personnalisation du système jusqu'au développement de classes ou de packages. Elles prolongent ainsi cette introduction vers une utilisation plus autonome du langage et une meilleure adaptation aux exigences des documents à rédiger. Cette autonomie se construit aussi par la pratique et par la consultation de la documentation présentée à la section 8.3. La compréhension des commandes et des dépendances d'un exemple facilite son adaptation et permet de constituer progressivement des modèles de documents réutilisables.