le cnam

Conservatoire national des arts et métiers

ÉCOLE DOCTORALE SCIENCES DES MÉTIERS DE L'INGÉNIEUR

Laboratoire MACS, Cnam Paris

Thèse de doctorat

Présentée par : **Éric Bavu**
Soutenue le : **10 Octobre 1794**
Discipline : **60e section CNU — Mécanique, génie mécanique, génie civil**
Spécialité : **Acoustique**

GUIDE D'UTILISATION DU TEMPLATE QUARTO CNAM POUR LES THÈSES

Un document de thèse auto-documenté

Thèse dirigée par :
M. Henri Grégoire, Abbé constitutionnel, fondateur du Cnam, Paris

Jury		
M. Jean-Antoine Chaptal	Professeur des Universités, Chaire de Chimie Appliquée, Conservatoire national des arts et métiers, Paris	Président du jury
M. Donald Ervin Knuth	Professeur émérite, Department of Computer Science, Stanford University, Palo Alto, États-Unis	Rapporteur
Mme Ada Lovelace	Maîtresse de Conférences HDR, Department of Mathematics, University of London, Royaume-Uni	Rapportrice
M. Jacques de Vaucanson	Inspecteur général des manufactures, Académie royale des sciences, Paris	Évaluateur
M. Leslie Lamport	Senior Principal Researcher, Microsoft Research, Redmond, États-Unis	Évaluateur
M. Henri Grégoire	Abbé constitutionnel, fondateur du Cnam, Conservatoire national des arts et métiers, Paris	Directeur de thèse

À Donald Knuth, qui a transformé la typographie en art computationnel, et à tous les doctorants qui ont perdu une journée à cause d'une accolade LaTeX manquante.

Remerciements

Ce travail n'aurait pas été possible sans le soutien indéfectible des écrits de Donald Knuth ; le *TEXbook* [1] a accompagné des générations de doctorants pendant des nuits de débogage de leur code LaTeX. Je remercie également Leslie Lamport pour avoir rendu TeX accessible au commun des mortels avec LaTeX [2], ainsi que l'équipe Quarto — J.J. Allaire, Charles Teague, Carlos Scheidegger, Yihui Xie, Christophe Dervieux notamment — pour avoir fait de la publication scientifique reproductible une réalité [3].

Mes remerciements vont à mon directeur de thèse virtuel pour les besoins de ce document, l'Abbé Henri Grégoire, dont la vision de 1794 — mettre les arts et les métiers à la portée de tous — résonne avec l'idéal du logiciel libre. Je doute qu'il ait anticipé les PDF/A-1b, mais l'esprit est là.

Je remercie les membres du jury virtuel pour leur disponibilité, en particulier M. Vaucanson dont l'expertise en automates mécaniques s'est avérée plus pertinente qu'il n'y paraît pour comprendre le pipeline de compilation pdflatex et la magie derrière Pandoc et les filtres Lua.

Enfin, une pensée émue pour tous les doctorants du Cnam et d'ailleurs qui ont ouvert un fichier de template `.tex` mal fichu pour la première fois et se sont demandé pourquoi leur accolade gauche était en trop. Ce guide est pour vous : grâce à Quarto, écrire devient aussi simple que du Markdown, mais avec des possibilités quasi infinies, et la beauté de la génération d'un même document sous différentes formes, avec une approche moderne.

Résumé

Ce travail présente et documente le template Quarto `quarto-cnam-thesis`, une extension permettant aux doctorants du Conservatoire national des arts et métiers (Cnam) de rédiger leur thèse en Quarto Markdown et de produire simultanément un PDF conforme à la maquette institutionnelle 2024-2025 (classe `book`, moteur pdflatex, conformité PDF/A-1b) et une version HTML navigable. La thèse est organisée en cinq chapitres documentant chacun une famille de fonctionnalités : structure et mise en forme, figures et mise en page, équations et tableaux, callouts et blocs de code, références et annotations collaboratives. La plupart des fonctionnalités est présentée selon le principe *code + rendu* : un bloc de code Markdown affiche la syntaxe exacte, immédiatement suivi du rendu réel dans le document. Cette approche auto-référentielle permet au document de servir à la fois de guide d'utilisation et de démonstration vivante du template.

Mots-clés : Quarto, LaTeX, pdflatex, PDF/A, thèse de doctorat, publication scientifique reproductible, Cnam

Abstract

This work presents and documents the `quarto-cnam-thesis` Quarto extension, which enables doctoral students at the Conservatoire national des arts et métiers (Cnam) to write their thesis in Quarto Markdown and simultaneously produce a PDF compliant with the 2024-2025 institutional template (book class, pdflatex engine, PDF/A-1b compliance) and a navigable HTML version. The thesis (a virtual one—I wrote mine a long time ago using good old LaTeX…) is organized into five chapters, each covering a family of features : document structure and formatting, figures and layout, equations and tables, callouts and code blocks, and finally references, glossary, and collaborative annotations. Most features are presented following a *code + output* principle : a Markdown code block displays the exact syntax to use, immediately followed by the live rendering in the document. This self-referential approach allows the document to serve both as a user guide and a live demonstration of the template.

Keywords : Quarto, LaTeX, pdflatex, PDF/A, doctoral thesis, reproducible scientific publishing, Cnam

Table des matières

Liste des tableaux

Liste des figures

Liste des acronymes

ABES Agence Bibliographique de l'Enseignement Supérieur

Cnam Conservatoire national des arts et métiers

CNU Conseil National des Universités

CSI Comité de Suivi Individuel

HDR Habilitation à Diriger des Recherches

LOF List of Figures

LOT List of Tables

QMD Quarto Markdown Document

TOC Table of Contents

YAML Yet Another Markup Language

Glossaire

Encadré Bloc mis en évidence destiné à attirer l'attention du lecteur ; produit par la syntaxe ::: .callout-* avec cinq types : note, tip, warning, important, caution

Liminaire Ensemble des pages placées avant le corps principal d'une thèse : remerciements, résumé, abstract, tables des matières, figures et tables, liste des acronymes, glossaire

Pandoc Convertisseur universel de documents développé par John MacFarlane ; socle technique de Quarto

pdfx Package LaTeX permettant la conformité PDF/A et PDF/X ; utilisé dans ce template pour produire un PDF/A-1b (option a-1b)

Postliminaire Ensemble des pages placées après le corps principal d'une thèse : conclusion générale, bibliographie, annexes

Quarto Système de publication scientifique open-source basé sur Pandoc, développé par Posit PBC ; produit des documents PDF, HTML, EPUB, Word, diaporamas, sites web et livres à partir de fichiers QMD

Introduction

Contexte et objectifs

Les doctorants du Conservatoire national des arts et métiers (Cnam) doivent déposer leur thèse sous forme d'un PDF conforme à la maquette institutionnelle tout en souhaitant disposer d'une version HTML navigable pour le partage et l'accessibilité. Ce double objectif — PDF/A archivable et HTML lisible — est exactement ce que propose le système de publication Quarto [3], à condition de disposer d'un template configuré pour les contraintes spécifiques du Cnam.

Le présent document est à la fois le *guide d'utilisation* du template `quarto-cnam-thesis` et sa *démonstration vivante* : chaque fonctionnalité décrite ici est simultanément expliquée, montrée en syntaxe brute, et rendue sous vos yeux dans ce même document.

Structure de ce guide

Ce guide est organisé en cinq chapitres thématiques :

1. **Structure et mise en forme** — hiérarchie des titres, numérotation, cross-références, mise en forme inline, listes.
2. **Figures et mise en page** — images statiques, figures générées par Python, mise en page multi-colonnes, figures PDF-only.
3. **Équations et tableaux** — syntaxe mathématique, tableaux Markdown simples, tableaux avancés avec l'extension `quartable`.
4. **Callouts et blocs de code** — les cinq types d'encadrés, code affiché sans exécution, code exécuté avec Python.
5. **Références, glossaire et annotations** — citations bibliographiques, système d'acronymes et de termes, commentaires collaboratifs.

Le principe directeur est constant : pour chaque fonctionnalité, un bloc de code affiche la syntaxe Markdown exacte (non exécutée), suivi immédiatement du rendu réel.

Prérequis

Les fichiers sources de la thèse sont des fichiers texte ordinaires avec l'extension `.qmd` (Quarto Markdown). Ils s'ouvrent et s'éditent avec n'importe quel éditeur de texte — VS Code, RStudio, Neovim, Notepad++ ou même le bloc-notes du système.

Quarto et LaTeX

Ce template requiert **Quarto 1.4** ou supérieur. Le moteur de compilation est pdflatex — XeLaTeX et LuaLaTeX ne sont pas supportés en raison des contraintes du package `pdfx` utilisé pour la conformité PDF/A-1b.

> **Étape 1 — Installer Quarto**
>
> `https://quarto.org/docs/get-started/` — installeur `.pkg`, `.msi` ou `.deb` selon la plateforme ; guide pas à pas pour VS Code, RStudio et autres éditeurs inclus.

Étape 2 — Installer LaTeX. Si vous n'avez pas encore LaTeX, utilisez TinyTeX, la distribution minimale intégrée à Quarto. Elle s'installe en quelques secondes et télécharge automatiquement les packages manquants au premier rendu :

Code (bash)

```bash
quarto install tinytex
```

> **Le premier rendu nécessite une connexion internet**
>
> Au premier rendu, TinyTeX télécharge les packages requis par le template (`pdfx`, `minitoc`, `tcolorbox`, `glossaries`…). Cela peut prendre quelques minutes. Les rendus suivants fonctionnent entièrement hors ligne.

Vous avez déjà LaTeX ? Toute distribution **TeX Live 2023+** ou **MiKTeX** à jour fonctionne sans configuration supplémentaire. Si votre distribution est ancienne, mettez-la à jour ou lancez `quarto install tinytex` pour passer à TinyTeX.

Système	Quarto	LaTeX
macOS	`.pkg` ou `brew install quarto`	`quarto install tinytex` *(recommandé)* ou TeX Live / MacTeX
Windows	`.msi` (64 bits)	`quarto install tinytex` *(recommandé)* ou MiKTeX / TeX Live

Système	Quarto	LaTeX
Linux (Debian/Ubuntu)	`sudo dpkg -i quarto-*.deb`	`quarto install tinytex` *(recommandé)* ou `sudo apt install texlive-full`

Code exécutable (Python, R…)

Un langage de script n'est nécessaire **que si la thèse contient des cellules de code exécutable** — figures calculées ou tableaux générés directement dans le document. Une thèse rédigée en texte pur avec des images statiques **n'a besoin d'aucun langage de script** : le script post-render (`postrender.ts`) tourne sur le runtime Deno intégré à Quarto.

Quarto supporte plusieurs moteurs de calcul :

— **Python** (via Jupyter) — utilisé dans les exemples de ce guide. → Documentation Quarto : Python

— **R** (via knitr) — bien adapté aux statistiques, à l'économétrie, à l'écologie et aux disciplines où R est le langage dominant. → Documentation Quarto : R

La suite de cette section détaille la configuration de **Python**, qui alimente les exemples de ce document. Pour R, se référer à la documentation Quarto ci-dessus.

Les packages requis par les exemples de ce document sont listés dans `requirements.txt` à la racine du projet : `numpy`, `matplotlib`, `pandas`, `tabulate`.

Environnement virtuel — pourquoi c'est important. Un environnement virtuel isole les packages de ce projet du reste de l'installation Python du système, évite les conflits de versions et facilite la reproductibilité. Il est fortement déconseillé d'installer les packages directement dans Python système.

uv — recommandé

uv installe Python, crée l'environnement virtuel et gère les packages en une seule commande, sans Python préalable sur le système. Quarto détecte automatiquement le dossier `.venv/` créé — aucune activation manuelle n'est nécessaire avant `quarto render`.

Code (bash)

```bash
# Étape 1 - installer uv (une seule fois, sur le système)
# macOS / Linux :
curl -LsSf https://astral.sh/uv/install.sh | sh
# Windows (PowerShell) :
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

```

# Étape 2 - dans le répertoire de la thèse :
uv python install 3.12          # télécharge et installe Python 3.12
uv venv                         # crée .venv/ dans le répertoire courant
uv pip install -r requirements.txt  # installe tous les packages + Jupyter
```

venv + pip — alternative

Si Python 3.10+ est déjà installé sur le système :

Code (bash)

```
python -m venv .venv
source .venv/bin/activate        # macOS / Linux
# .venv\Scripts\activate         # Windows (cmd)
# .venv\Scripts\Activate.ps1     # Windows (PowerShell)
pip install jupyter -r requirements.txt
```

Conda, pyenv...

Les utilisateurs conda peuvent créer un environnement dédié : `conda create -n mathese python=3.12 && conda activate mathese && pip install jupyter -r requirements.txt`. Les utilisateurs pyenv installent la version cible avec pyenv, puis utilisent venv comme ci-dessus.

Installer le template

Installer le template

Code (bash)

```
quarto use template zinc75/quarto-cnam-thesis
```

Cette commande copie tous les fichiers sources dans le répertoire courant, y compris les fichiers `.qmd` de cette documentation, qui servent de point de départ. Les fichiers de contenu à modifier se trouvent dans `content_fr/` (profil français) et `content_en/` (profil anglais), ainsi que les fichiers de configuration `yaml` (voir Annexe A).

Adapter le template à votre thèse

Deux fichiers sont à modifier avant le premier rendu.

`_quarto.yml` — les métadonnées de la thèse : titre, auteur, jury, date de soutenance, discipline... Ce fichier est commun aux deux sorties PDF et HTML. Une référence complète est donnée en

Annexe A.

`_quarto-fr.yml` (ou `_quarto-en.yml`) — la liste des chapitres qui constituent le livre. C'est le premier endroit à modifier : remplacer les chapitres d'exemple par vos propres fichiers `.qmd` :

Code (yaml)

```yaml
book:
  chapters:
    - index.qmd                                  # ne pas supprimer
    - content_fr/liminaire/remerciements.qmd
    - content_fr/liminaire/resume.qmd
    - content_fr/liminaire/abstract.qmd
    - content_fr/liminaire/tables.qmd            # ne pas supprimer
    - content_fr/chapitres/00-introduction.qmd
    - content_fr/chapitres/01-mon-chapitre.qmd   # ← vos chapitres ici
    - content_fr/chapitres/02-mon-chapitre.qmd
    - content_fr/postliminaire/conclusion.qmd
    - content_fr/postliminaire/bibliographie.qmd
  appendices:
    - content_fr/postliminaire/annexes.qmd
    - content_fr/postliminaire/annexes-pdfa.qmd
```

Pour créer un chapitre : ajouter un fichier `.qmd` dans `content_fr/chapitres/` et l'insérer dans cette liste.

Générer le document

En Quarto, on appelle ça le **rendu** : cela lance la génération du fichier `.tex`, la compilation, et la création des fichiers `html`, puisque ce template fournit une double sortie `PDF` et `HTML`. Pour cela, lancer l'une des commandes qui suivent, en fonction de votre langue de rédaction de la thèse (Français ou Anglais suivant les règles des écoles doctorales et la composition de votre jury) :

! Toujours utiliser un profil de langue

Code (bash)

```bash
quarto render --profile fr   # → _these_fr/these_fr_<auteur>.pdf + HTML
quarto render --profile en   # → _thesis-en/these_en_<auteur>.pdf + HTML
```

`quarto render` sans profil échoue : la liste des chapitres est définie dans les profils `_quarto-fr.yml` et `_quarto-en.yml`, pas dans `_quarto.yml`.

Prévisualisation en temps réel

Pendant la rédaction, il n'est pas nécessaire de relancer `quarto render` à chaque modification. La commande `quarto preview` surveille les fichiers sources et reconstruit automatiquement le document dès qu'un fichier `.qmd` est sauvegardé :

Code (bash)

```bash
quarto preview --profile fr
```

Quarto ouvre alors la version HTML dans le navigateur et la rafraîchit en quelques secondes à chaque sauvegarde. Seul le chapitre modifié est recompilé (reconstruction incrémentale), ce qui rend le cycle édition–aperçu très rapide — même avec des blocs de code Python, grâce au cache `freeze: auto`.

Workflow recommandé pendant la rédaction

Utiliser `quarto preview` au quotidien pour vérifier la mise en forme, les cross-références et les figures. Relancer `quarto render` ponctuellement (avant une réunion de suivi, avant un dépôt) pour obtenir le PDF final. Les deux sorties — HTML et PDF — sont cohérentes : ce qui s'affiche correctement en HTML se compile correctement en PDF.
Pour les utilisateurs de **VS Code** ou **RStudio**, l'extension Quarto intègre un bouton *Render* et un panneau de prévisualisation directement dans l'éditeur — sans passer par le terminal.

CHAPITRE

1

Structure et mise en forme du document

Contenu

Ce chapitre présente les blocs de base de tout document Quarto : la hiérarchie des titres, la numérotation, les cross-références de sections, et la mise en forme inline.

1.1 Hiérarchie des titres

Quarto supporte quatre niveaux de titres numérotés dans ce template (paramètre `number-depth: 4` dans `_extension.yml`). La syntaxe utilise les dièses Markdown :

Code (`markdown`)

```markdown
# Titre de chapitre (niveau 1)

## Titre de section (niveau 2)

### Titre de sous-section (niveau 3)

```

```
#### Titre de sous-sous-section (niveau 4)
```

En PDF, ces niveaux correspondent respectivement aux commandes LaTeX `\chapter`, `\section`, `\subsection` et `\subsubsection`. En HTML, ils génèrent les balises `<h2>` à `<h5>` (le `<h1>` étant réservé au titre du livre).

1.1.1 Sous-section de démonstration

Ceci est une `###` en situation. La numérotation automatique produit ici « 1.1.1 ».

1.1.1.1 Sous-sous-section de démonstration

Et ceci est une `####`, numérotée « 1.1.1.1 ». Au-delà de ce niveau, Quarto ne numérote plus — utilisez des listes ou des paragraphes en gras si vous avez besoin d'un cinquième niveau.

> **Profondeur maximale recommandée**
>
> Un plan à quatre niveaux numérotés est généralement le maximum acceptable dans une thèse. Un plan plus profond est souvent le signe que la structure doit être repensée.

1.2 Sections numérotées et non numérotées

Par défaut, tous les titres de niveau 1 à 4 sont numérotés. Pour retirer la numérotation, on ajoute la classe `{.unnumbered}` :

Code (markdown)

```
## Section sans numéro {.unnumbered}
```

Pour les pages de la partie liminaire et postliminaire (remerciements, résumé, conclusion…), on combine `{.unnumbered}` avec `{.toc-black}` — cette dernière classe demande au filtre Lua d'afficher le titre en noir dans la table des matières, et déclenche la numérotation romaine en front matter quand `pagestyle-sections: true` :

Code (markdown)

```
# Conclusion {.unnumbered}

# Remerciements {.unnumbered .toc-black}
```

! `.unnumbered` vs `.unnumbered .toc-black`

— `{.unnumbered}` seul : titre non numéroté, affiché en couleur dans la TOC (chapitres du corps de thèse : introduction, conclusion).
— `{.unnumbered .toc-black}` : titre non numéroté, affiché en **noir** dans la TOC (parties liminaires et postliminaires).

Ne mettez jamais `{.unnumbered}` sur un chapitre du corps de thèse numéroté (chapitres 1, 2, 3...) — ce serait incohérent avec le plan général.

1.3 Cross-références de sections (@sec-)

Pour créer un renvoi vers une section, il faut : (1) donner un identifiant au titre cible avec `{#sec-label}`, puis (2) y faire référence avec `@sec-label` :

Code (markdown)

```
## Hiérarchie des titres {#sec-titres}

Comme expliqué en @sec-titres, les niveaux vont de `#` à `####`.
```

Rendu :

Comme expliqué en Section 1.1, les niveaux vont de # à ####.

Quarto résout automatiquement le numéro et le lien hypertexte dans les deux formats.

Convention de nommage des labels

Utilisez le préfixe `sec-` pour les sections, `fig-` pour les figures, `tbl-` pour les tableaux, `eq-` pour les équations. Ces préfixes sont imposés par Quarto pour que les cross-références fonctionnent correctement dans les deux formats.

1.4 Titre court (`short-title`)

Quand un titre de chapitre ou de section est trop long pour tenir dans le *running header* du PDF ou dans la table des matières, on utilise l'attribut `short-title` :

Code (markdown)

```
# Un titre vraiment très long qui déborde dans le header {short-title="Titre court"}

## Une sous-section au titre interminable {short-title="Titre court"}
```

Le titre court est utilisé dans le running header (en-tête de page recto/verso) et dans la table des matières. Le titre long reste tel quel dans le corps du texte.

Astuce

Visez 40 à 50 caractères maximum pour un `short-title`. Au-delà, il risque de déborder lui aussi selon la mise en page.

1.5 Mise en forme inline

1.5.1 Emphase et code

Code (markdown)

```
**gras**, *italique*, ***gras italique***, `code inline`
~~barré~~, ^exposant^, ~indice~
:::
```

Rendu :

gras, *italique*, ***gras italique***, `code inline`, ~~barré~~, $^{\text{exposant}}$, $_{\text{indice}}$.

Usage académique du gras et de l'italique

Dans une thèse, le gras et l'italique servent à marquer la **terminologie** lors de sa première introduction, ou à citer un *titre d'ouvrage*. Les utiliser pour décorer le texte ou insister sur des phrases entières est une pratique à éviter.

1.5.2 Soulignement

Markdown n'a pas de syntaxe native pour le soulignement. Quarto utilise la notation span avec la classe `.underline` :

Code (markdown)

```
[texte souligné]{.underline}
```

Rendu :

texte souligné.

En PDF, Pandoc génère `\underline{... }`; en HTML, un `<span>` avec `text-decoration: underline`. La syntaxe `<u>texte</u>` fonctionne en HTML mais est ignorée en PDF.

Le soulignement est à éviter dans une thèse

Le soulignement est un héritage de la machine à écrire, époque où l'italique n'était pas disponible. Dans un document typographié, il est réservé aux liens hypertexte. Pour mettre en valeur un terme, utilisez `*italique*` ou `**gras**`.

1.5.3 Listes

Code (markdown)

```
- Premier élément non ordonné
- Deuxième élément
  - Sous-élément (indentation 2 espaces)

1. Premier élément ordonné
2. Deuxième élément
   a. Sous-élément (indentation 3 espaces)
```

Rendu :

- Premier élément non ordonné
- Deuxième élément
  - Sous-élément (indentation 2 espaces)

1. Premier élément ordonné
2. Deuxième élément

   a. Sous-élément (indentation 3 espaces)

Le numéro écrit n'a pas d'importance

Pandoc ne lit que le numéro du **premier** élément (pour déterminer le départ du compteur) ; tous les suivants sont ignorés et auto-incrémentés. Ces deux syntaxes produisent le même rendu :

Code (markdown)

```
1. Premier
2. Deuxième
3. Troisième
```

Code (markdown)

```
1. Premier
1. Deuxième
1. Troisième
```

La convention `1. 1. 1.` est même recommandée : plus besoin de renuméroter si l'on insère ou déplace un élément. La même logique s'applique aux sous-listes : `a. a. a.` suffit pour obtenir `a.`, `b.`, `c.`...

1.5.4 Listes de tâches

La syntaxe `- [ ]` / `- [x]` produit une liste de tâches. La casse de `x` n'a pas d'importance (`X` fonctionne aussi) :

Code (markdown)

```
- [ ] Rédiger l'introduction
- [x] Choisir le template Quarto
- [ ] Soumettre au directeur de thèse
```

Rendu :

- ☐ Rédiger l'introduction
- ☒ Choisir le template Quarto
- ☐ Soumettre au directeur de thèse

Rendu différent selon le format de sortie

En **HTML**, Quarto génère de vraies cases à cocher `<input type="checkbox">` (désactivées, non interactives).
En **PDF** (pdflatex), Pandoc utilise les symboles mathématiques `$\square$` (case vide) et `$\boxtimes$` (case cochée) du package `amssymb`. L'apparence est correcte mais typographiquement différente des cases HTML.

Usage dans une thèse

Les listes de tâches sont adaptées aux notes de travail, aux annexes méthodologiques, ou à un README. Dans le corps de la thèse, préférez une liste non ordonnée classique accompagnée d'un verbe d'état (« complété », « en cours »…) : c'est plus neutre et plus conforme au registre académique.

1.5.5 Blockquotes

Code (markdown)

```
> « La typographie est le métier de donner une forme visuelle au langage. »
>
> - Robert Bringhurst [@Bringhurst2004]
```

Rendu :

> « La typographie est le métier de donner une forme visuelle au langage. »
>
> — Robert Bringhurst [4]

Listes de définitions en pdflatex

La syntaxe Pandoc de liste de définitions (`terme\n:  définition`) est supportée par Quarto mais produit un rendu sans style visuel particulier en pdflatex. Si vous avez besoin de listes de définitions mises en forme, utilisez un tableau Markdown (voir Section 3.4) ou un environnement LaTeX brut.

1.5.6 Notes de bas de page

Quarto propose deux syntaxes pour les notes de bas de page.

Syntaxe inline — la note est écrite directement dans le texte, entre `^[` et `]` :

Code (markdown)

```
Le Cnam a été fondé en 1794^[Décret de la Convention nationale du 19 vendémiaire
an III (10 octobre 1794).] à l'initiative de l'abbé Grégoire.
```

Rendu :

Le Cnam a été fondé en 1794 [1] à l'initiative de l'abbé Grégoire.

Syntaxe par référence — l'appel de note et son contenu sont séparés. Utile pour les notes longues ou pour garder le texte principal lisible :

Code (markdown)

```
La thèse doit être déposée sur theses.fr[^depot] avant la soutenance.

[^depot]: Portail national des thèses géré par l',
    accessible sur <https://theses.fr>.
```

Rendu :

La thèse doit être déposée sur theses.fr [2] avant la soutenance.

En PDF, les deux syntaxes produisent une note numérotée en bas de page. En HTML, elles produisent une note de bas de page interactive (numéro cliquable avec renvoi vers l'appel de note).

Quelle syntaxe choisir ?

Préférez la **syntaxe inline** `^[...]` pour les notes courtes (une phrase) — le contenu reste visible au fil de la rédaction. Utilisez la **syntaxe par référence** `[^label]` pour les notes longues ou multi-paragraphes, et pour éviter de surcharger visuellement la source Markdown quand la note dépasse deux lignes.

1. Décret de la Convention nationale du 19 vendémiaire an III (10 octobre 1794).
2. Portail national des thèses géré par l'ABES, accessible sur https://theses.fr.

1.5.7 Sauts de page

Quarto fournit le shortcode pour insérer un saut de page compatible avec les deux formats de sortie :

Code (markdown)

```
Fin du contenu de cette section.

{{< pagebreak >}}

Début de la section suivante.
```

En PDF, génère `\newpage`. En HTML, il insère une règle de style `page-break-after: always` — sans effet à l'écran, mais utile si la page HTML est imprimée depuis le navigateur.

`\newpage` vs `\clearpage`

Si vous avez besoin de `\clearpage` (vide les figures et tableaux flottants en attente avant de sauter) plutôt que `\newpage`, utilisez un bloc LaTeX brut dans un `{.content-visible when-format="pdf"}` :

Code (markdown)

````
::: {.content-visible when-format="pdf"}
```{=latex}
\clearpage
```
:::
````

Dans une thèse avec de nombreuses figures, `\clearpage` est souvent préférable pour éviter que les flottants s'accumulent et apparaissent bien après leur point d'insertion dans le texte.

Les sauts de page manuels sont à utiliser avec parcimonie

En règle générale, laissez LaTeX gérer la mise en page. Les sauts manuels résolvent un problème local mais peuvent en créer d'autres plus loin dans le document (pages blanches, flottants déplacés). Préférez les ajuster en toute fin de rédaction, une fois le contenu figé.

CHAPITRE

2

Figures et mise en page

Contenu

Ce chapitre documente l'insertion de figures statiques, la génération de figures par code Python, la mise en page multi-colonnes, et le cas particulier des figures PDF-only (TikZ, PGFplots).

2.1 Image statique avec caption et cross-référence

La syntaxe de base pour insérer une image est :

Code (markdown)

```
1 ![Légende de la figure.](../../../images/mon-image.png){#fig-label width=80%}
```

L'identifiant `#fig-label` permet de faire une cross-référence avec `@fig-label`. L'attribut `width` accepte des pourcentages ou des valeurs absolues (`5cm`, `0.5\linewidth`).

Voici un exemple avec le logo du Cnam :

Code (markdown)

```
1 ![Le logo officiel du Cnam.](../../../images/Cnam.jpg){#fig-logo-cnam width=40%}
```

La Figure 2.1 montre le logo officiel du Cnam, chargé depuis le dossier `images/` à la racine du projet.

FIGURE 2.1 – Le logo officiel du Cnam.

Formats d'images recommandés

- **PDF** : privilégier les formats vectoriels `.pdf` ou `.eps` pour les schémas et graphiques (net à toutes résolutions). Pour les photos, `.png` ou `.jpg` en 300 dpi minimum.
- **HTML** : tous les formats web standards (`.png`, `.jpg`, `.svg`). Les `.svg` sont particulièrement adaptés pour les schémas.
- Quarto gère automatiquement la conversion des formats selon la cible de rendu.

Chemins relatifs au fichier `.qmd`, pas à la racine

Les chemins d'images sont relatifs à l'emplacement du fichier `.qmd`. Depuis `content_fr/chapitres/`, le dossier `images/` à la racine est à `../../images/monfichier.png`. Ce comportement est identique en PDF et en HTML.

2.2 Figures générées par code Python

Quarto peut exécuter des cellules Python (via Jupyter) et inclure leur sortie graphique comme figures numérotées. La syntaxe du chunk utilise des options préfixées par `#|` :

Code (markdown)

````
```{python}
#| label: fig-sincos
#| fig-cap: "Courbes $\\sin(x)$ et $\\cos(x)$ sur $[0, 2\\pi]$."
#| echo: true
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2 * np.pi, 300)
fig, axes = plt.subplots(1, 2, figsize=(8, 3))
axes[0].plot(x, np.sin(x), color='#d20025')
axes[0].set_title(r'$\sin(x)$')
axes[1].plot(x, np.cos(x), color='#005EA5')
axes[1].set_title(r'$\cos(x)$')
for ax in axes:
    ax.set_xlabel('$x$')
    ax.grid(True, alpha=0.3)
plt.tight_layout()
```
````

Rendu :

Code (python)

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2 * np.pi, 300)
fig, axes = plt.subplots(1, 2, figsize=(8, 3))
axes[0].plot(x, np.sin(x), color='#d20025')
axes[0].set_title(r'$\sin(x)$')
axes[1].plot(x, np.cos(x), color='#005EA5')
axes[1].set_title(r'$\cos(x)$')
for ax in axes:
    ax.set_xlabel('$x$')
    ax.grid(True, alpha=0.3)
plt.tight_layout()
```

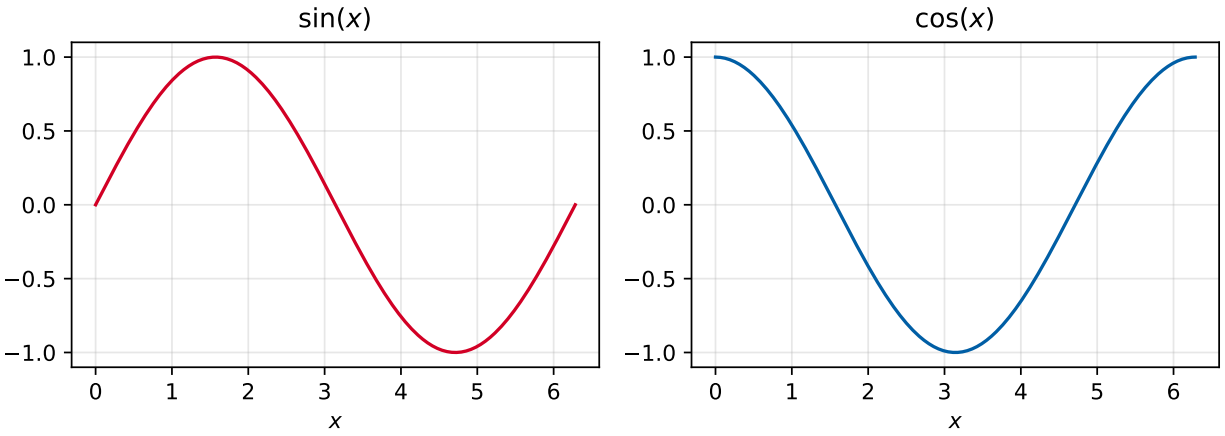


FIGURE 2.2 – Courbes $\sin(x)$ et $\cos(x)$ sur $[0, 2\pi]$.

La Figure 2.2 est générée à chaque compilation. Avec `execute: freeze: auto` dans `_quarto.yml`, Quarto met en cache les résultats dans `_freeze/` et ne réexécute le code que si le chunk a changé — utile pour les figures longues à calculer.

`plt.tight_layout()` est essentiel

Sans `plt.tight_layout()`, les étiquettes des axes se chevauchent fréquemment dans le PDF. Terminez toujours vos figures matplotlib par cette ligne.

`echo: false` pour le document final

Pendant la rédaction, `echo: true` affiche le code — pratique pour la relecture. Pour la version soumise à l'école doctorale, passez les chunks de figures en `#| echo: false` pour n'afficher que le graphique.

2.3 Mise en page multi-colonnes

La directive `layout-ncol` place plusieurs figures côte à côte. On l'applique à un div contenant des sous-figures :

Code (markdown)

````
::: {#fig-duo layout-ncol=2}

```{python}
#| label: fig-duo-a
#| fig-cap: "Fonction $x^2$"
import matplotlib.pyplot as plt, numpy as np
x = np.linspace(-2, 2, 200)
plt.plot(x, x**2, color='#d20025'); plt.grid(True, alpha=0.3); plt.tight_layout()
```

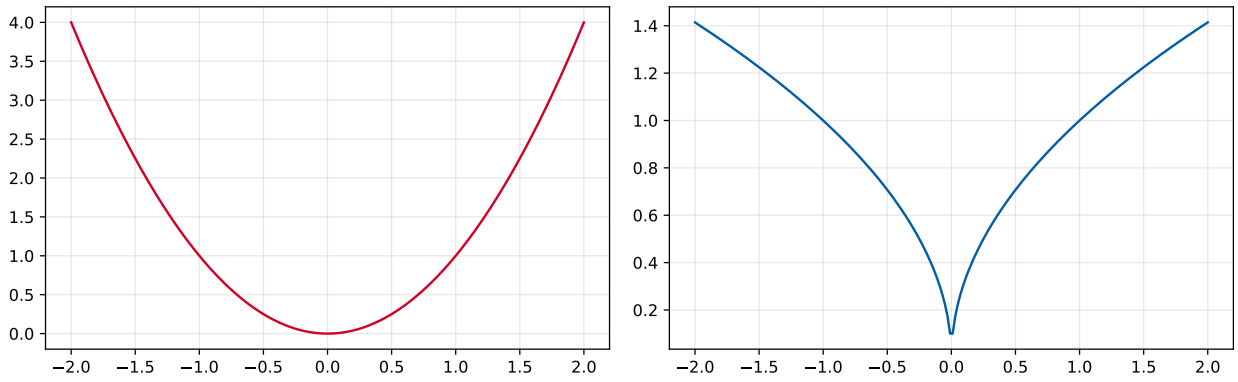
```{python}
#| label: fig-duo-b
#| fig-cap: "Fonction $\\sqrt{|x|}$"
import matplotlib.pyplot as plt, numpy as np
x = np.linspace(-2, 2, 200)
plt.plot(x, np.sqrt(np.abs(x)), color='#005EA5'); plt.grid(True, alpha=0.3);
↪  plt.tight_layout()
```

Deux fonctions élémentaires.
:::
````

Rendu :

Code

```
import matplotlib.pyplot as plt, numpy as np
x = np.linspace(-2, 2, 200)
plt.plot(x, x**2, color='#d20025'); plt.grid(True, alpha=0.3);
↪  plt.tight_layout()
import matplotlib.pyplot as plt, numpy as np
x = np.linspace(-2, 2, 200)
plt.plot(x, np.sqrt(np.abs(x)), color='#005EA5'); plt.grid(True,
↪  alpha=0.3); plt.tight_layout()
```



(a) Fonction x^2

(b) Fonction $\sqrt{|x|}$

FIGURE 2.3 – Deux fonctions élémentaires.

La Figure 2.3 regroupe les deux sous-figures Figure 2.3a et Figure 2.3b. La dernière ligne de texte dans le div devient la légende globale.

2.4 Figures PDF-only (TikZ, PGFplots)

Certaines figures complexes ne peuvent être générées qu'en LaTeX (TikZ, PGFplots, `circuitikz`...). On les insère via un bloc `{=latex}` dans un div figure, accompagné d'un contenu de remplacement visible en HTML :

Code (markdown)

```
:::{#fig-tikz fig-pos='h'}
```{=latex}
\centering
\begin{tikzpicture}
  \draw[thick, ->] (0,0) -- (3,0) node[right] {$x$};
  \draw[thick, ->] (0,0) -- (0,2) node[above] {$y$};
  \draw[blue, thick] plot[domain=0:2.8, samples=50] (\x, {sin(\x r)});
\end{tikzpicture}
```

::: {.content-visible when-format="html"}
(Figure TikZ - disponible dans la version PDF uniquement)
:::

Exemple de figure TikZ insérée en LaTeX brut.
:::
```

Rendu :

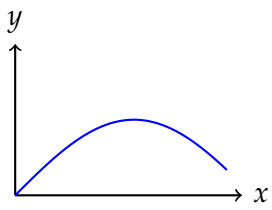


FIGURE 2.4 – Exemple de figure TikZ insérée en LaTeX brut.

Alternative : compiler la figure TikZ en PDF séparé

Pour les figures TikZ complexes, une approche plus robuste consiste à les compiler en fichier `.pdf` standalone, puis à les inclure comme une image normale :

Code (markdown)

```
1 ![Légende.](figures/mon-schema-tikz.pdf){#fig-tikz width=80%}
```

Cela évite les problèmes de packages en conflit et accélère la compilation.

2.5 Positionnement des flottants (PDF)

En LaTeX, les figures et les tables sont des *flottants* : LaTeX décide de leur position pour obtenir la meilleure mise en page possible. L'attribut `fig-pos` (figures) ou `tbl-pos` (tables) permet de lui donner des instructions.

Code (markdown)

```
1 ![Légende.](../../../images/Cnam.jpg){#fig-ex fig-pos='tb'}
```

Les spécificateurs se combinent librement dans une chaîne :

Spécificateur	Signification
`h`	*here* — à l'endroit exact dans le texte (si la place le permet)
`t`	*top* — en haut de la page courante ou suivante
`b`	*bottom* — en bas de la page courante ou suivante
`p`	*page* — sur une page dédiée aux flottants
`!`	Ignore les contraintes internes de LaTeX (densité, nombre de flottants)

Spécificateur	Signification
`H`	Force le placement *ici*, sans aucun flottement (nécessite `\usepackage{float}`)

La valeur par défaut de Quarto est `tbp`. Pour un chunk Python, le spécificateur se passe en option de cellule :

Code (markdown)

```
```{python}
#| label: fig-python
#| fig-cap: "Ma figure."
#| fig-pos: 'h'
```
```

Pour définir un spécificateur par défaut sur tout le document, ajouter dans `_quarto.yml` :

Code (yaml)

```
fig-pos: 'H'
```

h n'est pas une garantie

Le spécificateur `h` demande à LaTeX de placer le flottant *ici si possible*, mais LaTeX peut l'ignorer si la place est insuffisante. La combinaison `ht` (ici, puis haut de page) est généralement plus robuste. `H` force le placement sans condition, mais peut créer des pages presque vides si la figure est grande — à réserver aux cas où la position est vraiment critique.

Les flottants ne s'appliquent qu'au PDF

En HTML, `fig-pos` et `tbl-pos` sont ignorés : les figures et tableaux sont insérés dans le flux du document à leur point d'insertion, comme tout autre élément.

CHAPITRE

3

Équations et tableaux

Contenu

Ce chapitre couvre la syntaxe mathématique (équations inline, blocs numérotés, environnements avancés) et les deux niveaux de tableaux : les pipe tables Markdown simples et les tableaux enrichis via l'extension `quartable`.

3.1 Équations inline

Les équations inline s'écrivent entre signes dollar : `$...$`. En PDF elles utilisent le mode mathématique LaTeX natif, en HTML elles sont rendues par KaTeX.

Code (markdown)

```
La transformée de Fourier d'une fonction $f$ est définie par
$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\,e^{-2\pi i x\xi}\,\mathrm{d}x$.
```

Rendu :

La transformée de Fourier d'une fonction f est définie par $\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x\xi}\,\mathrm{d}x$.

Espaces fins en mathématiques

Utilisez `\,` pour les espaces fins : `f(x)\,\mathrm{d}x` donne $f(x)\,\mathrm{d}x$. Utilisez `\mathrm{d}` (romain) plutôt que `d` (italique) pour les différentielles — c'est la norme ISO 80000-2.

3.2 Équations en bloc numérotées

Les équations en bloc s'encadrent par `$$...$$`. Pour les numéroter et créer une cross-référence, on ajoute `{#eq-label}` :

Code (markdown)

```
$$
i\hbar\frac{\partial \Psi}{\partial t} = \hat{H}\Psi
$$ {#eq-schrodinger}

L'équation de Schrödinger (@eq-schrodinger) décrit l'évolution temporelle...
```

Rendu :

$$i\hbar\frac{\partial \Psi}{\partial t} = \hat{H}\Psi \tag{3.1}$$

L'équation de Schrödinger (Équation 3.1) décrit l'évolution temporelle d'un système quantique.

`@eq-label` sans crochets

La cross-référence d'une équation s'écrit `@eq-label`, **sans crochets**. Les crochets `[@eq-label]` sont réservés aux citations bibliographiques et ne fonctionneront pas pour les équations.

3.3 Environnements mathématiques avancés

Les environnements LaTeX comme `aligned`, `cases`, `bmatrix` fonctionnent nativement dans Quarto, en PDF (LaTeX pur) comme en HTML (KaTeX) :

Code (markdown)

```
$$
\begin{cases}
  \dot{x}(t) = Ax(t) + Bu(t) \\
  y(t) = Cx(t) + Du(t)
\end{cases}
$$ {#eq-statespace}
```

Rendu :

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{3.2}$$

La représentation d'état (Équation 3.2) est la forme standard en automatique.

Code (markdown)

```
$$
\begin{aligned}
  \nabla \cdot \mathbf{E}  &= \frac{\rho}{\varepsilon_0} \\
  \nabla \cdot \mathbf{B}  &= 0 \\
  \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\
  \nabla \times \mathbf{B} &= \mu_0\mathbf{J} + \mu_0\varepsilon_0\frac{\partial
↪  \mathbf{E}}{\partial t}
\end{aligned}
$$ {#eq-maxwell}
```

Rendu :

$$\begin{aligned} \nabla \cdot \mathbf{E} &= \frac{\rho}{\varepsilon_0} \\ \nabla \cdot \mathbf{B} &= 0 \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{B} &= \mu_0 \mathbf{J} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \end{aligned} \tag{3.3}$$

Les équations de Maxwell (Équation 3.3) en notation vectorielle — un seul numéro pour l'ensemble du bloc.

Pour attribuer **un numéro à chaque ligne**, on utilise `align` (sans le d final) à la place de `aligned`. L'environnement `align` est autonome — il ne s'imbrique pas dans `$$... $$`. Pandoc enveloppe tout `$$... $$` dans `\[... \]` ou `\begin{equation}`, et nester `\begin{align}` à l'intérieur est invalide (erreur amsmath). Il faut un bloc `{=latex}` brut pour le PDF et `$$... $$` pour le HTML :

Code (markdown)

````
::: {.content-visible when-format="pdf"}
```{=latex}
\begin{align}
  \nabla \cdot \mathbf{E}  &= \frac{\rho}{\varepsilon_0} \\
  \nabla \cdot \mathbf{B}  &= 0 \\
  \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\
  \nabla \times \mathbf{B} &= \mu_0\mathbf{J}
    + \mu_0\varepsilon_0\frac{\partial \mathbf{E}}{\partial t}
\end{align}
```
:::

::: {.content-visible when-format="html"}
$$
\begin{align}
  \nabla \cdot \mathbf{E}  &= \frac{\rho}{\varepsilon_0} \\
  \nabla \cdot \mathbf{B}  &= 0 \\
  \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\
  \nabla \times \mathbf{B} &= \mu_0\mathbf{J}
    + \mu_0\varepsilon_0\frac{\partial \mathbf{E}}{\partial t}
\end{align}
$$
:::
````

Rendu :

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0} \tag{3.4}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{3.5}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{3.6}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \tag{3.7}$$

Chaque `\\` reçoit son propre numéro séquentiel. Pour supprimer le numéro d'une ligne, ajouter `\notag` avant le `\\`. Pour référencer une ligne individuelle, utiliser `\label{mon-label}` dans le bloc `{=latex}` et `\eqref{mon-label}` dans le texte (PDF uniquement — KaTeX ne génère pas de labels

adressables).

⚠ `\begin{align}` est incompatible avec `$$... $$` en Quarto

Quarto (Pandoc) enveloppe tout `$$... $$` dans `\[... \]` (sans label) ou `\begin{equation}... \end{equation}` (avec `{#eq-}`). Les deux formes créent une imbrication invalide avec `\begin{align}`, que amsmath rejette :

```
Package amsmath Error: Erroneous nesting of equation structures
```

La solution est systématiquement le pattern `{.content-visible}` dual : bloc `{=latex}` brut pour le PDF, `$$\begin{align}... \end{align}$$` pour le HTML (KaTeX supporte `align` nativement).

`aligned` vs `align` — quelle différence ?

Environnement	Syntaxe Quarto	Numérotation	`@eq-`
`aligned`	`$$\begin{aligned} ... \end{aligned}$$`	un numéro pour le **bloc**	✓
`align`	`{=latex}` + `$$... $$` (dual)	un numéro **par ligne**	×
`align*`	`{=latex}` + `$$... $$` (dual)	aucune	—

Règle pratique : développement de calcul → `aligned` (un numéro, `@eq-` possible). Équations indépendantes → `align` avec le pattern dual.

Numérotation hiérarchique (3.1a), (3.1b) avec `subequations`

Pour la numérotation **(3.1a)**, **(3.1b)**. . ., combiner `subequations` et `align` dans un bloc `{=latex}` (KaTeX ne supporte pas `subequations`) :

Code (markdown)

````
::: {.content-visible when-format="pdf"}
```{=latex}
\begin{subequations}\label{eq-navier}
\begin{align}
  \rho\,\frac{D\mathbf{u}}{Dt}
    &= -\nabla p + \mu\,\nabla^2\mathbf{u} + \mathbf{f}
    \label{eq-navier-momentum} \\
  \nabla \cdot \mathbf{u} &= 0
    \label{eq-navier-incomp}
\end{align}
\end{subequations}
```
:::
````

`\label{eq-navier}` référence le groupe ; `\label{eq-navier-momentum}` et `\label{eq-navier-incomp}` référencent les lignes — accessibles avec `\eqref{}`. Ajouter un bloc `{.content-visible when-format="html"}` frère avec `$$\begin{align}...\end{align}$$` pour le rendu HTML.

3.4 Tableaux Markdown simples

La syntaxe *pipe table* de Pandoc/Quarto crée des tableaux avec alignement par colonne. La légende, précédée de `:`, doit être placée immédiatement après le tableau :

Code (markdown)

```markdown
| Commande          | Format PDF     | Format HTML   |
|:------------------|:--------------:|:-------------:|
| `@fig-label`      | numéro figure  | lien hypert.  |
| `@tbl-label`      | numéro table   | lien hypert.  |
| `@eq-label`       | numéro éq.     | lien hypert.  |
| `@sec-label`      | numéro section | lien hypert.  |

: Préfixes de cross-références Quarto. {#tbl-crossref}
```

Rendu :

TABLE 3.1 – Préfixes de cross-références Quarto.

Commande	Format PDF	Format HTML
`@fig-label`	numéro figure	lien hypert.
`@tbl-label`	numéro table	lien hypert.
`@eq-label`	numéro éq.	lien hypert.
`@sec-label`	numéro section	lien hypert.

La Table 3.1 récapitule les quatre préfixes à connaître.

Alignement des colonnes

— `:---` — aligné à gauche
— `:---:` — centré
— `---:` — aligné à droite

La ligne `|:---|:---:|---:|` définit l'alignement de chaque colonne.

⚠ La légende doit être collée au tableau

La ligne `: Légende. {#tbl-label}` doit être placée **immédiatement** après le tableau, sans ligne vide intermédiaire. Une ligne vide brise l'association tableau-légende et produit une légende orpheline sans numérotation.

3.5 Tableaux avancés avec `quartable`

L'extension `quartable` (listée dans `filters:` de `_quarto.yml`) ajoute trois capacités aux tableaux Markdown :

— **Rowspan** : `[texte]{rs=N}` fusionne N lignes ; `^` continue la cellule fusionnée

— **Colspan** : `[texte]{cs=N}` fusionne N colonnes

— **Midrules** : `===` insère un séparateur horizontal ; `===2-4` une règle partielle sur les colonnes 2 à 4 .

💡 Documentation complète de l'extension `quartable`

Pour voir toutes les fonctionnalités de cette extension (inspirée du package `booktabs` LaTeX), consultez https://github.com/zinc75/quarto-quartable/ et https://zinc75.github.io/quarto-quartable/test_quartable.pdf/.

</> Code (markdown)

```
| Famille              | Fonctionnalité       | Chapitre |
|:---------------------|:---------------------|:--------:|
| ===                  |                      |          |
| [Structure]{rs=3}    | Titres et niveaux    | 1        |
| ^                    | Cross-références     | 1        |
| ^                    | Mise en forme inline | 1        |
| ===                  |                      |          |
| [Visuels]{rs=3}      | Figures statiques    | 2        |
| ^                    | Figures Python       | 2        |
| ^                    | Multi-colonnes       | 2        |
| ===                  |                      |          |
| [Math & données]{rs=3} | Équations          | 3        |
| ^                    | Tableaux simples     | 3        |
| ^                    | `quartable`          | 3        |

: Récapitulatif des fonctionnalités par famille. {#tbl-recap}
```

Rendu :

TABLE 3.2 – Récapitulatif des fonctionnalités des chapitres 1 à 3.

Famille	Fonctionnalité	Chapitre
Structure	Titres et niveaux	1
	Cross-références	1
	Mise en forme inline	1
Visuels	Figures statiques	2
	Figures Python	2
	Multi-colonnes	2
Math & données	Équations	3
	Équations avancées	3
	Tableaux simples	3

⚠ `quartable` doit être dans `filters:`

La syntaxe `[texte]{rs=N}` n'est reconnue que si `quartable` est listée dans la clé `filters:` de `_quarto.yml`. Sans cela, les accolades restent affichées telles quelles dans le rendu.

ℹ Règles partielles avec `===N-M`

`===2-4` insère une règle qui ne s'étend que sous les colonnes 2 à 4 (incluses). Utile pour souligner un sous-groupe de colonnes sans séparer toute la ligne.

CHAPITRE

4

Callouts et blocs de code

Contenu

Les callouts permettent de mettre en évidence des informations importantes. Les blocs de code permettent d'afficher ou d'exécuter du code. Ce chapitre les présente tous avec des exemples tirés du contexte doctoral.

4.1 Les cinq types de callouts

Quarto définit cinq types de callouts, chacun avec une sémantique précise :

Code (`markdown`)

```
::: {.callout-specification}
## Titre (optionnel) du callout
Contenu du callout, avec `specification` qui peut prendre les valeurs
`note`, `tip`, `warning`, `important`, ou `caution`, chacun étant associé à
un code couleur et une icône en entête du bloc de callout.
:::
```

4.1.1 `callout-note` — information neutre

Code (markdown)

```
::: {.callout-note}
## Composition du jury

Le jury d'une thèse Cnam comprend au minimum un président, deux rapporteurs
(ayant l') et un ou deux examinateurs. La soutenance dure environ
45 minutes de présentation, suivies des questions du jury.
:::
```

Rendu :

> **Composition du jury**
>
> Le jury d'une thèse Cnam comprend au minimum un président, deux rapporteurs (ayant l'HDR) et un ou deux examinateurs. La soutenance dure environ 45 minutes de présentation, suivies des questions du jury.

4.1.2 `callout-tip` — conseil actionnable

Code (markdown)

```
::: {.callout-tip}
## Commencez à rédiger tôt

Commencez à rédiger votre thèse dès la deuxième année de doctorat. La rédaction
tardive est la première cause de dépassement de la durée contractuelle de thèse.
Même des ébauches imparfaites valent mieux que des sections vides.
:::
```

Rendu :

> **Commencez à rédiger tôt**
>
> Commencez à rédiger votre thèse dès la deuxième année de doctorat. La rédaction tardive est la première cause de dépassement de la durée contractuelle de thèse. Même des ébauches imparfaites valent mieux que des sections vides.

4.1.3 `callout-warning` — piège à éviter

Code (markdown)

```
::: {.callout-warning}
## Ne renommez pas les chapitres sans mettre à jour le profil

Si vous renommez un fichier `.qmd`, mettez à jour la liste `chapters:` dans
`_quarto-fr.yml` (ou `_quarto-en.yml`). Quarto ne détecte pas automatiquement
les fichiers manquants et échouera silencieusement.
:::
```

Rendu :

> **Ne renommez pas les chapitres sans mettre à jour le profil**
>
> Si vous renommez un fichier `.qmd`, mettez à jour la liste `chapters:` dans `_quarto-fr.yml` (ou `_quarto-en.yml`). Quarto ne détecte pas automatiquement les fichiers manquants et échouera silencieusement.

4.1.4 `callout-important` — point non négociable

Code (markdown)

```
::: {.callout-important}
## PDF/A-1b obligatoire pour theses.fr

L' exige un PDF/A pour l'archivage sur theses.fr. Ce template
génère automatiquement un PDF/A-1b via `\usepackage[a-1b]{pdfx}`. Ne retirez pas
ce package du template.
:::
```

Rendu :

> **PDF/A-1b obligatoire pour theses.fr**
>
> L'ABES exige un PDF/A pour l'archivage sur theses.fr. Ce template génère automatiquement un PDF/A-1b via `\usepackage[a-1b]{pdfx}`. Ne retirez pas ce package du template.

4.1.5 `callout-caution` — danger avec conséquences

Code (markdown)

```
::: {.callout-caution}
## N'utilisez pas `date:` pour la date de soutenance

Quarto interprète la clé `date:` comme une date JavaScript et produit
« Invalid Date » pour les dates françaises (ex. « 12 juin 2026 »). Utilisez
exclusivement la clé personnalisée `date-soutenance:` prévue dans ce template.
:::
```

Rendu :

> **N'utilisez pas `date:` pour la date de soutenance**
>
> Quarto interprète la clé `date:` comme une date JavaScript et produit « Invalid Date » pour les dates françaises (ex. « 12 juin 2026 »). Utilisez exclusivement la clé personnalisée `date-soutenance:` prévue dans ce template.

4.2 Callouts avec titre et repliement

Le titre s'ajoute avec une ligne `## Titre` à l'intérieur du callout (comme dans les exemples ci-dessus). Pour rendre un callout replié par défaut (cliquable pour l'ouvrir), on utilise l'attribut `collapse` :

Code (markdown)

```
::: {.callout-tip collapse="true"}
## Cliquer pour voir le conseil (replié par défaut)

Ce callout est replié à l'ouverture du document HTML. En PDF, il s'affiche
toujours en entier (le repliement est une fonctionnalité HTML uniquement).
:::
```

Rendu :

> **Cliquer pour voir le conseil (replié par défaut)**
>
> Ce callout est replié à l'ouverture du document HTML. En PDF, il s'affiche toujours en entier (le repliement est une fonctionnalité HTML uniquement).

4.3 Blocs de code — affichage seul

Pour afficher du code sans l'exécuter, on utilise soit un bloc délimité par des backticks sans spécificateur de langage entre accolades, soit un chunk avec `eval: false` :

Code (markdown)

````
```python
# Ce bloc est affiché mais non exécuté
import numpy as np
x = np.linspace(0, 10, 100)
print(x.mean())
```
````

Rendu :

Code (python)

```python
# Ce bloc est affiché mais non exécuté
import numpy as np
x = np.linspace(0, 10, 100)
print(x.mean())
```

Pour afficher le code d'un chunk exécutable sans l'exécuter (utile pour des exemples d'installation ou de configuration) :

Code (markdown)

````
```{python}
#| eval: false
#| echo: true
# Installation des dépendances
pip install matplotlib numpy pandas
```
````

Rendu :

Code (python)

```python
# Installation des dépendances du template
# pip install matplotlib numpy pandas
```

Options de chunk les plus utiles

TABLE 4.1 – Options de chunk Quarto fréquemment utilisées.

Option	Valeurs	Effet
`echo`	`true/false`	affiche ou masque le code
`eval`	`true/false`	exécute ou non le code
`output`	`true/false`	affiche ou masque la sortie
`warning`	`true/false`	affiche ou masque les avertissements
`fig-cap`	texte	légende de la figure produite
`label`	`fig-...` ou `tbl-...`	identifiant pour cross-référence

4.4 Blocs de code — avec exécution Python

Dans ce document, tous les exemples de code exécutable — y compris les figures matplotlib du Section 2.2 — utilisent `#| echo: true` pour afficher à la fois le code source et le résultat. Dans une thèse finalisée, on masque généralement le code en passant `#| echo: false` : seule la figure ou le tableau apparaît, sans le code qui l'a produit.

Langages pris en charge nativement par Quarto

Quarto prend en charge trois langages de calcul scientifique **sans configuration supplémentaire** : **Python**, **R** et **Julia**. La syntaxe des options de chunk (`#| echo:`, `#| fig-cap:`, etc.) et le mécanisme de cross-référence (`#| label:`) sont identiques pour les trois langages. Ce document n'illustre que Python.
Documentation officielle :

- Python : https://quarto.org/docs/computations/python.html
- R : https://quarto.org/docs/computations/r.html
- Julia : https://quarto.org/docs/computations/julia.html

Un chunk Python avec `eval: true` (valeur par défaut) exécute le code et affiche la sortie. Voici un exemple avec `pandas` :

Code (markdown)

````
```{python}
#| echo: true
#| label: tbl-pandas
#| tbl-cap: "Statistiques descriptives des fonctions trigonométriques."
import numpy as np, pandas as pd
from IPython.display import Markdown

x = np.linspace(0, 2 * np.pi, 1000)
df = pd.DataFrame({r'$\sin(x)$':           np.sin(x),
                   r'$\cos(x)$':           np.cos(x),
                   r'$\tan(x)$ (tronqué)': np.clip(np.tan(x), -5, 5)})
Markdown(df.describe().round(3).to_markdown())
```
````

Rendu :

Code (python)

```python
import numpy as np, pandas as pd
from IPython.display import Markdown

x = np.linspace(0, 2 * np.pi, 1000)
df = pd.DataFrame({r'$\sin(x)$':           np.sin(x),
                   r'$\cos(x)$':           np.cos(x),
                   r'$\tan(x)$ (tronqué)': np.clip(np.tan(x), -5, 5)})
Markdown(df.describe().round(3).to_markdown())
```

TABLE 4.2 – Statistiques descriptives des fonctions trigonométriques.

	$\sin(x)$	$\cos(x)$	$\tan(x)$ (tronqué)
count	1000	1000	1000
mean	0	0.001	-0
std	0.707	0.708	2.335
min	-1	-1	-5
25%	-0.706	-0.705	-0.997
50%	-0	0.002	-0
75%	0.706	0.708	0.997
max	1	1	5

`freeze: auto` dans `_quarto.yml`

Avec `execute: freeze: auto`, Quarto stocke les résultats des cellules dans `_freeze/` et ne les recalcule que si le code du chunk a changé. Cela accélère considérablement les rendus successifs pour les thèses avec beaucoup de calculs.

Pensez à versionner le dossier `_freeze/`

Si vous versionnez votre thèse sur GitHub et souhaitez que les relecteurs puissent compiler sans Python installé, committez le dossier `_freeze/`. Les résultats mis en cache y sont stockés et seront utilisés à la place de l'exécution.

CHAPITRE

5

Références, glossaire et annotations

Contenu

Ce chapitre présente le système de citations bibliographiques (style IEEE via `IEEEtran-francais.bst`), le système de glossaire et d'acronymes, les commentaires collaboratifs, et les pages liminaires spéciales.

5.1 Citations bibliographiques

5.1.1 Syntaxe de base

Les citations s'insèrent avec `[@clé]`. La clé correspond à l'identifiant BibTeX défini dans `references.bib` :

Code (markdown)

```
La composition typographique informatique a été révolutionnée par Knuth [@Knuth1984],
dont le système \TeX{} a été rendu accessible par Lamport avec \LaTeX{}
↪ [@Lamport1994].
Quarto [@Allaire2022] unifie ces outils dans un environnement de publication moderne.
```

Rendu :

La composition typographique informatique a été révolutionnée par Knuth [1], dont le système TEX a été rendu accessible par Lamport avec LATEX [2]. Quarto [3] unifie ces outils dans un environnement de publication moderne.

5.1.2 Références multiples

Code (markdown)

```
Les outils de visualisation Python [@Hunter2007] et les ouvrages de référence
typographiques [@Bringhurst2004; @Mittelbach2023] complètent la palette de l'auteur.
```

Rendu :

les outils de visualisation Python [5] et les ouvrages de référence typographiques [4, 6] complètent la palette de l'auteur.

⚠ Toujours utiliser `[@clé]` avec les crochets

Le style `IEEEtran-francais.bst` utilise `natbib`. La syntaxe `@clé` sans crochets appelle `\citet{}` qui est cassé avec IEEEtran — elle produit le nom de l'auteur suivi de `[?]`. **Utilisez toujours `[@clé]`** pour obtenir `[N]` (numérotation IEEE).

5.1.3 Fichier `references.bib`

Le fichier `references.bib` doit se trouver **à la racine du projet**. Pandoc en mode natbib génère `\bibliography{references}` (sans chemin), et BibTeX cherche le fichier dans le répertoire courant. Un fichier dans `content_fr/references.bib` ne sera pas trouvé.

Gestionnaires bibliographiques

Zotero (avec le connecteur Better BibTeX), JabRef ou Mendeley exportent directement en format `.bib`. L'entrée `@IEEEtranBSTCTL` en tête de fichier configure le style IEEEtran (nombre d'auteurs avant « et al. », etc.) — ne la supprimez pas.

5.2 Glossaire et acronymes

5.2.1 Déclarer des entrées dans `glossaire-entries.qmd`

Toutes les entrées sont déclarées dans `content_fr/liminaire/glossaire-entries.qmd` via deux shortcodes :

Code (markdown)

```
{{< acr label SIGLE "Développé complet" >}}
{{< terme label "mot" "Définition complète du terme" >}}
```

Les shortcodes écrivent automatiquement `_glossaire-entries.tex` (lu par LaTeX) et alimentent les tables HTML. Les labels doivent être uniques.

5.2.2 Utiliser les acronymes dans le texte

Quatre shortcodes permettent de référencer un acronyme selon la forme souhaitée :

TABLE 5.1 – Shortcodes de référence du glossaire.

Shortcode	Sortie PDF	Sortie HTML
`{{< gls cnam >}}`	forme courte (1re occur. : longue)	forme courte
`{{< acrs cnam >}}`	toujours forme courte	toujours forme courte
`{{< acrl cnam >}}`	toujours forme longue	toujours forme longue
`{{< acrf cnam >}}`	long (court)	long (court)

Exemples en situation :

Code (markdown)

```
Le {{< acrs cnam >}} ({{< acrl cnam >}}) a été fondé en 1794.
Le format {{< acrf qmd >}} est le format natif de Quarto.
La numérotation  est générée automatiquement par Quarto.
```

Rendu :

le Cnam (Conservatoire national des arts et métiers) a été fondé en 1794. Le format Quarto Markdown Document (QMD) est le format natif de Quarto.

Comportement PDF vs HTML

En PDF, expanse à la première occurrence (forme longue + sigle entre parenthèses) puis utilise la forme courte. En HTML, cette logique de première occurrence n'est pas gérée par le shortcode — il produit toujours la forme courte. Pour la cohérence, préférez à la première occurrence dans le texte, et ensuite.

5.3 Commentaires collaboratifs (quarto-comments)

5.3.1 Configuration dans _quarto.yml

L'extension quarto-comments (dossier _extensions/comments/) est configurée via la clé extensions.quarto- dans _quarto.yml :

Code (yaml)

```yaml
extensions:
  quarto-comments:
    enabled: true
    authors:
      eb:
        name: "Eric Bavu"
      ag:
        name: "Abbé Grégoire"
      # Couleurs auto depuis la palette Bootstrap 5 ; surcharger si besoin :
      # nc:
      #   name: "Nicolas Conté"
      #   color_html: "#198754"
      #   color_latex: "green!30"
```

5.3.2 Les quatre types de commentaires

Code (markdown)

```
{{< comment "Reformuler cette phrase, elle est trop longue." author="ag" >}}
{{< todo "Ajouter une figure illustrant ce point." author="eb" >}}
{{< note "Voir aussi Bringhurst 2004, §2.4." author="ag" >}}
{{< question "Ce résultat est-il reproductible sur un autre jeu de données ?"
↪  author="ag" inline=true >}}
```

Rendu en situation :

Voici une phrase avec un commentaire du directeur de thèse (qui a ajouté directement celui-ci dans le fichier `.qmd` parce qu'il n'est pas frileux !).

Ces commentaires ne sont visible **que** si l'option `quarto-comments` est à `enabled : true` dans le fichier `yaml`, ce qui est le cas pour cette documentation.

Il est aussi possible d'ajouter des commentaires en ligne : Une note bibliographique

Abbé Grégoire : Voir Knuth 1984, chapitre 12.

sans que cela occupe de la place dans la marge.

Abbé Grégoire : Ceci est un commentaire de type 'comment' par le directeur.

Eric Bavu : Penser à passer l'option à 'false' avant soumission aux rapporteurs!

`inline=true` occupe toute la ligne en PDF

En HTML, les annotations `inline=true` apparaissent comme un span coloré compact, exactement à la largeur de leur contenu. En PDF, le package `todonotes` rend `\todo[inline]` sous la forme d'un bloc coloré pleine largeur — c'est une contrainte de conception du package, sans contournement possible sans perdre la capacité de couper les annotations longues sur plusieurs lignes. Utilisez `inline=true` pour de courtes notes qui doivent rester dans le flot du texte ; pour des annotations plus longues, omettez `inline` pour placer la note en marge.

Désactiver les commentaires avant le dépôt final

Avant de soumettre la thèse à l'école doctorale et de la déposer sur theses.fr, passez `extensions.quarto-comments.enabled: false` dans `_quarto.yml`. Les annotations ne doivent pas figurer dans la version officielle archivée.

5.4 Annotations web avec Hypothesis (opt-in)

Pour une relecture à distance, la version HTML peut être enrichie d'annotations web via Hypothesis. Le directeur ou les relecteurs sélectionnent du texte dans le navigateur et ajoutent des commentaires sans toucher aux fichiers source. Les annotations sont stockées sur les serveurs Hypothesis et liées à l'URL de la page — elles persistent d'un rendu à l'autre tant que l'URL reste stable (déploiement GitHub Pages).

Pour activer, décommenter le bloc suivant dans `_quarto.yml` :

Code (yaml)

```yaml
format:
  cnam-thesis-html:
    comments:
      hypothesis:
        theme: clean   # clean = discret ; classic = barre latérale toujours visible
```

Créer ensuite un groupe privé sur hypothes.is et partager le lien d'invitation aux relecteurs. Hypothesis est désactivé par défaut dans tous les profils.

5.5 Pages liminaires spéciales

5.5.1 Résumé et abstract

Les pages `resume.qmd` et `abstract.qmd` utilisent des classes CSS qui ajoutent une bordure gauche en HTML :

Code (markdown)

```markdown
::: {.resume-block}
Texte du résumé en français.

**Mots-clés :** mot1, mot2, mot3
:::
```

! Résumé français obligatoire pour toute thèse Cnam

Même pour une thèse rédigée en anglais, un résumé en français est **obligatoire** selon les règles de l'ABES pour le dépôt sur theses.fr. C'est le rôle du fichier `content_en/frontmatter/resume.qmd` dans le profil anglais.

5.5.2 Le Comité de Suivi Individuel (CSI)

i Processus doctoral français — le CSI

Le Comité de Suivi Individuel (CSI) est obligatoire depuis 2016 pour tous les doctorants en France. Il se réunit au moins une fois par an, indépendamment du directeur de thèse, pour vérifier l'avancement, les conditions de travail et le respect du calendrier. Le compte-rendu du CSI doit être joint au dossier de soutenance.
Le template HTML de cette thèse est particulièrement adapté pour les réunions de CSI : il permet de partager un lien vers la version en ligne, plus facile à parcourir qu'un PDF de 200 pages.

Conclusion

Ce guide a présenté les fonctionnalités du template `quarto-cnam-thesis` en suivant le principe *code + rendu* : chaque syntaxe est montrée telle qu'on la tape, puis immédiatement rendue dans le document. Cette approche auto-référentielle vise à rendre le template autonome comme documentation.

Récapitulatif des fonctionnalités

TABLE 5.2 – Récapitulatif de toutes les fonctionnalités documentées.

Fonctionnalité	Syntaxe clé	Chapitre
Structure	Hiérarchie `#` / `##` / `###` / `####`	1
	`{.unnumbered}`, `{.toc-black}`	1
	`{#sec-label}` + `@sec-label`	1
	`short-title="... "`	1
Figures	`![cap](img){#fig-label width=X%}`	2
	Chunk `{python}` avec `#|` `label: fig-...`	2
	`layout-ncol=2`	2
	Bloc `{=latex}` dans div `:::{#fig-}`	2
Math & tables	`$... $` / `$$... $$` `{#eq-label}`	3
	`\begin{aligned}`, `\begin{cases}`	3
	Pipe table + `: Légende.` `{#tbl-label}`	3
	`[texte]{rs=N}`, `^`, `===` (`quartable`)	3
Callouts	`{.callout-note}`	4
	`{.callout-tip}`	4
	`{.callout-warning}`	4
	`{.callout-important}`	4
	`{.callout-caution}`	4

TABLE 5.2 – Récapitulatif de toutes les fonctionnalités documentées.

Fonctionnalité	Syntaxe clé	Chapitre
Références	`[@clé]` / `[@clé1; @clé2]`	5
	`{{{< acr >}}}`, `{{{< terme >}}}`, `{{{< acrs >}}}`	5
	`{{{< comment >}}}`, `{{{< todo >}}}`, `{{{< note >}}}`	5
	`.resume-block`, `.abstract-block`	5

Workflow recommandé

Cycle de rédaction quotidien

1. **Rédiger en HTML** : `quarto render --profile fr --to cnam-thesis-html --no-clean` — rendu rapide, navigation aisée, idéal pour la relecture.
2. **Vérifier le PDF** avant chaque réunion CSI : `quarto render --profile fr` — compile tout et génère le PDF/A-1b.
3. **Versionner régulièrement** : committez au moins après chaque session de rédaction, idéalement avant et après chaque réunion de direction.
4. **Désactiver les commentaires** avant le dépôt final : `extensions.quarto-comments.enabled: false` dans `_quarto.yml`.

Le template est maintenu sur GitHub. Pour signaler un problème ou proposer une amélioration : zinc75/quarto-cnam-thesis.

Bibliographie

[1] D. E. Knuth, *The TEXbook*, ser. Computers & Typesetting. Reading, Massachusetts : Addison-Wesley, 1984, vol. A.

[2] L. Lamport, *LATEX : A Document Preparation System*, 2e éd. Reading, Massachusetts : Addison-Wesley Professional, 1994.

[3] J. Allaire, C. Teague, Y. Xie et C. Dervieux, "Quarto," 2022.

[4] R. Bringhurst, *The Elements of Typographic Style*, 3e éd. Vancouver : Hartley & Marks, 2004.

[5] J. D. Hunter, "Matplotlib : A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, p. 90–95, 2007.

[6] F. Mittelbach et U. Fischer, *The LATEX Companion*, 3e éd. Reading, Massachusetts : Addison-Wesley Professional, 2023.

ANNEXE

A

Configuration YAML

Contenu

Cette annexe documente les deux fichiers de configuration principaux du template : `_quarto.yml` (métadonnées communes à tous les rendus) et `_quarto-fr.yml` / `_quarto-en.yml` (métadonnées propres à chaque langue).

A.1 Métadonnées dans `_quarto.yml` — page de garde

`_quarto.yml` contient les champs **communs à tous les profils** : jury, encadrants, informations institutionnelles. Le titre, l'auteur et la date de soutenance se trouvent dans le profil de langue (voir la section Profils de langue ci-dessous).

Code (yaml)

```yaml
book:
  downloads: [pdf]                  # bouton PDF dans la barre HTML

discipline: "60e section CNU - Mécanique, génie mécanique, génie civil"
specialite: "Acoustique"
ecole-doctorale: "Sciences des Métiers de l'Ingénieur"
laboratoire: "Laboratoire MACS, Cnam Paris"
```

```yaml
directeur: "Pr. Prénom NOM, Professeur des Universités, Cnam Paris"
cotutelle: false

dedicace: "Dédicace optionnelle... "   # supprimer si absent
```

A.2 Composition du jury pour la page de garde

Code (yaml)

```yaml
jury:
  - nom: "Mme Prénom NOM"
    titre: "Titre, Unité de recherche, Établissement"
    role: "Présidente"
  - nom: "M. Prénom NOM"
    titre: "Titre, Unité de recherche, Établissement"
    role: "Rapporteur"
  - nom: "Mme Prénom NOM"
    titre: "Titre, Unité de recherche, Établissement"
    role: "Rapportrice"
  - nom: "M. Prénom NOM"
    titre: "Maître de conférences HDR, Unité, Établissement"
    role: "Évaluateur"
  - nom: "M. Prénom NOM"
    titre: "Professeur des Universités, MACS, Cnam Paris"
    role: "Directeur de thèse"
```

A.3 Options de la section Infrastructure

Code (yaml)

```yaml
glossaire: true          # true = charge le package LaTeX glossaries
pagestyle-sections: true # true = numérotation romaine/arabe/Romain par section
bibliography-manual: true
minitoc-newpage: false   # true = saut de page après chaque mini-TOC
chapter-openright: true  # false = pour version numérique sans page de droite
index: false             # true = génère un index via \index{terme}
```

A.4 Commentaires collaboratifs (quarto-comments)

Code (yaml)

```yaml
extensions:
  quarto-comments:
    enabled: true    # false avant le dépôt final sur theses.fr
    authors:
      doc:
        name: "Prénom NOM"       # doctorant·e
      dir:
```

```yaml
        name: "Directeur·rice"
      # Couleurs auto-assignées depuis la palette Bootstrap 5 ; surcharger si besoin :
      # codir:
      #   name: "Co-directeur·rice"
      #   color_html: "#198754"
      #   color_latex: "green!30"
```

Voir Section 5.3 pour la syntaxe des shortcodes et le comportement PDF/HTML.

A.5 Annotations web Hypothesis (opt-in)

Pour permettre aux relecteurs d'annoter directement la version HTML dans leur navigateur, décommenter ce bloc dans `_quarto.yml` :

Code (yaml)

```yaml
format:
  cnam-thesis-html:
    comments:
      hypothesis:
        theme: clean   # clean = discret ; classic = barre latérale toujours visible
```

Créer un groupe privé sur hypothes.is et partager le lien d'invitation. Les annotations sont liées à l'URL — fonctionne mieux sur un déploiement stable (GitHub Pages). Voir Section 5.3 pour plus de détails.

A.6 Exécution Python

Code (yaml)

```yaml
execute:
  freeze: auto    # recalcule seulement si le code du chunk change
  echo: true      # affiche le code par défaut (surchargeable par chunk)
```

A.7 Profils de langue : `_quarto-fr.yml` et `_quarto-en.yml`

Les profils de langue **surchargent** `_quarto.yml` au moment du rendu avec `--profile fr` ou `--profile en`. Ils définissent les métadonnées propres à la langue — **titre, auteur, date de soutenance** — ainsi que la liste des chapitres et les options techniques du rendu.

A.7.1 Liste des chapitres (première chose à adapter)

C'est la première chose à modifier lors de la prise en main du template. La liste complète des fichiers qui constituent le livre se trouve dans `_quarto-fr.yml` (ou `_quarto-en.yml`), sous la clé `book.chapters` :

Code (yaml)

```yaml
book:
  chapters:
    - index.qmd                                  # page d'accueil HTML - ne pas supprimer
    - content_fr/liminaire/remerciements.qmd
    - content_fr/liminaire/resume.qmd
    - content_fr/liminaire/abstract.qmd
    - content_fr/liminaire/tables.qmd            # TOC + listes PDF - ne pas supprimer
    - content_fr/liminaire/glossaire-entries.qmd # supprimer si glossaire: false
    - content_fr/liminaire/acronymes.qmd         # supprimer si glossaire: false
    - content_fr/liminaire/glossaire.qmd         # supprimer si pas de termes
    - content_fr/chapitres/00-introduction.qmd
    - content_fr/chapitres/01-chapitre1.qmd      # ← ajouter vos chapitres ici
    - content_fr/chapitres/02-chapitre2.qmd
    - content_fr/postliminaire/conclusion.qmd
    - content_fr/postliminaire/bibliographie.qmd
  appendices:
    - content_fr/postliminaire/annexes.qmd
    - content_fr/postliminaire/annexes-pdfa.qmd
```

Pour ajouter un chapitre : créer un fichier `.qmd` dans `content_fr/chapitres/` et l'insérer dans cette liste à l'emplacement souhaité.

⚠ Ne pas supprimer `index.qmd` ni `tables.qmd`

`index.qmd` est le point d'entrée du livre Quarto — le supprimer casse le rendu HTML. `tables.qmd` génère la table des matières, la liste des figures et la liste des tableaux dans le PDF.

A.7.2 Autres options du profil

Code (yaml)

```yaml
thesis-lang: fr      # langue de rédaction : fr ou en
validate: false      # mettre à true avant dépôt sur theses.fr (voir @sec-pdfa)

book:
  title: "Titre de la thèse"
  subtitle: "Sous-titre éventuel"   # supprimer si absent
  author: "Prénom NOM"
  output-file: these_fr             # nom de base du PDF (these_fr_<auteur>.pdf)

date-soutenance: "10 Octobre 1794"  # NE PAS utiliser date:

project:
  output-dir: _these_fr     # répertoire de sortie
  post-render:
    - ./_scripts/postrender.ts fr _these_fr
```

ANNEXE

B

Validation PDF/A-1b

Contenu

B.1 Dépôt sur theses.fr

La plateforme theses.fr exige un fichier PDF/A-1b (ISO 19005-1). Le PDF produit par ce template est conçu pour satisfaire les critères retenus par le CINES — l'organisme qui gère l'archivage institutionnel des thèses françaises.

B.2 Validateur officiel : facile.cines.fr

facile.cines.fr (CINES) est le validateur de référence pour le dépôt sur theses.fr. C'est lui — et non des outils tiers — qui détermine si votre PDF sera accepté. Il suffit d'y uploader le PDF pour obtenir un rapport immédiat.

B.3 Validation automatique depuis le script

Le script post-render peut interroger le webservice CINES automatiquement. Il suffit de passer `validate: true` dans `_quarto-fr.yml` :

Code (yaml)

```yaml
# Dans _quarto-fr.yml - avant dépôt sur theses.fr :
validate: true
```

Puis lancer le rendu normalement :

Code (bash)

```bash
quarto render --profile fr --to cnam-thesis-pdf
```

Le script envoie le PDF au webservice CINES et affiche le résultat :

```
Validating PDF/A on facile.cines.fr (CINES)...
[OK]  PDF/A-1b valide -- archivable sur theses.fr.
```

En cas d'échec :

```
[!!]  PDF/A-1b non valide.
      -> Corriger via : https://facile.cines.fr/#correction
```

Après la validation

Remettre `validate: false` dans `_quarto-fr.yml` pour les rendus quotidiens — l'appel au webservice n'est nécessaire qu'avant le dépôt final.

B.4 Si la validation échoue

Utiliser le service de correction officiel : **facile.cines.fr/#correction**. Il corrige automatiquement les non-conformités les plus courantes sans aucune installation supplémentaire.