

# Documentation Latex

Exemples et références pour  
la bibliothèque *Utils*

Nicolas Le Guerroué

<b>1</b>	<b>Préambule</b>	<b>5</b>
<b>2</b>	<b>Introduction</b>	<b>6</b>
2.1	Présentation . . . . .	6
2.2	Installation . . . . .	7
2.3	Organisation du projet . . . . .	7
2.4	Fusion de projets . . . . .	9
2.5	Compilation du projet . . . . .	9
2.5.1	Première compilation . . . . .	10
2.5.2	Compilation classique . . . . .	10
2.5.3	Vérification orthographique . . . . .	12
2.5.4	Mise à jour Git . . . . .	13
2.5.5	Création d'un nouveau projet . . . . .	13
2.5.6	Mise à jour de l'autocomplétion . . . . .	13
2.5.7	Compilation avec VSCode . . . . .	13
2.6	Conventions . . . . .	14
<b>3</b>	<b>Bibliothèque Adding</b>	<b>15</b>
3.1	Création d'une nomenclature . . . . .	15
<b>4</b>	<b>Bibliothèque Badges</b>	<b>16</b>
4.1	Création de badges avec des couleurs . . . . .	16
<b>5</b>	<b>Bibliothèque Electronic</b>	<b>17</b>
5.1	Création de chronogrammes fixes . . . . .	17
5.2	Création de chronogrammes flottants . . . . .	18
5.3	Création de schémas électriques . . . . .	18
<b>6</b>	<b>Bibliothèque Graphic</b>	<b>20</b>
6.1	Affichage d'un graphique 2D avec insertion des données depuis un fichier txt (csv) . . . . .	20
6.2	Affichage d'un graphique 2D avec insertion des données depuis une liste de points . . . . .	21
6.3	Affichage d'un graphique 2D avec insertion des données depuis une équation . . . . .	22
6.4	Affichage d'un graphique 2D avec insertion des données depuis plusieurs sources . . . . .	23
6.5	Affichage de deux graphiques . . . . .	24
<b>7</b>	<b>Bibliothèque Header</b>	<b>26</b>
7.1	Mise en forme de la page de garde avec une image . . . . .	26
7.2	Mise en forme de la page de garde sans image . . . . .	26
7.3	Mise en forme de la page des parties . . . . .	26
7.4	Ajout d'un trait entre l'en-tête et le corps de la page . . . . .	26
7.5	Ajout d'un trait entre le corps de la page et le bas de page . . . . .	26
7.6	Définition de la présentation globale des pages . . . . .	27

7.7	Redéfinition des titres des chapitres . . . . .	27
7.8	Mettre le document en pleine page . . . . .	27
7.9	Récupérer le chapitre courant . . . . .	27
<b>8</b>	<b>Bibliothèque Images</b>	<b>28</b>
8.1	Ajout d'une image non-flottante . . . . .	28
8.2	Ajout d'une image non-flottante avec une rotation . . . . .	28
<b>9</b>	<b>Bibliothèque Items</b>	<b>29</b>
9.1	Création d'un liste . . . . .	29
9.1.1	Options . . . . .	29
<b>10</b>	<b>Bibliothèque Labels</b>	<b>30</b>
10.1	Création de labels colorés . . . . .	30
<b>11</b>	<b>Bibliothèque Layout</b>	<b>31</b>
11.1	Mise en gras . . . . .	31
11.2	Mise en italique . . . . .	31
11.3	Mise en gras et italique . . . . .	31
11.4	Ajout d'un espace vertical . . . . .	31
<b>12</b>	<b>Bibliothèque Links</b>	<b>33</b>
12.1	Paramétrage des liens et des méta-données . . . . .	33
<b>13</b>	<b>Bibliothèque Maths</b>	<b>34</b>
13.1	Création d'une matrice 3*3 . . . . .	34
13.2	Création d'un vecteur à trois dimensions . . . . .	34
13.3	Création d'un torseur à trois dimensions . . . . .	34
<b>14</b>	<b>Bibliothèque MessageBox</b>	<b>35</b>
14.1	Création de boites de dialogues . . . . .	35
<b>15</b>	<b>Bibliothèque Pdf</b>	<b>36</b>
15.1	Insertion d'un document PDF . . . . .	36
15.2	Insertion d'un ensemble de pages d'un document PDF . . . . .	36
<b>16</b>	<b>Bibliothèque Programming</b>	<b>37</b>
16.1	Affichage d'un code C/C++ avec titre . . . . .	37
16.2	Affichage d'un code C/C++ sans titre . . . . .	37
16.3	Affichage d'un code Python avec titre . . . . .	38
16.4	Affichage d'un code Python sans titre . . . . .	38
16.5	Affichage d'un code Bash avec titre . . . . .	39
16.6	Affichage d'un code bash sans titre . . . . .	39
<b>17</b>	<b>Bibliothèque Tables</b>	<b>40</b>

<b>18 Bibliothèque Theorems</b>	<b>41</b>
18.1 Création d'une question . . . . .	41
18.2 Création d'une reponse . . . . .	41
18.3 Création d'une propriete . . . . .	41
18.4 Création d'une proposition . . . . .	41
18.5 Création d'une remarque . . . . .	42
18.6 Création d'un exemple . . . . .	42
18.7 Création d'une définition . . . . .	42
 <b>19 Bibliothèque Titles</b>	 <b>43</b>
19.0.1 Titre de chapitre . . . . .	43
19.1 Titre de section . . . . .	43
19.1.1 Titre de sous-section . . . . .	43

- ▶ Document réalisé en L<sup>A</sup>T<sub>E</sub>X par Nicolas Le Guerroué pour la documentation des bibliothèques Utils
- ▶ Permission vous est donnée de copier, distribuer et/ou modifier ce document sous quelque forme et de quelque manière que ce soit.
- ▶ Version du January 12, 2022
- ▶ Taille de police: 11pt
- ☎ 06.20.88.75.12
- ✉ [nicolasleguerroue@gmail.com](mailto:nicolasleguerroue@gmail.com)
- ▶ Dans la mesure du possible, évitez d'imprimer ce document si ce n'est pas nécessaire. Il est optimisé pour une visualisation sur un ordinateur et contient beaucoup d'images.

ADSR ADLN

## Présentation

Ce document a pour but de présenter les fonctionnalités de la bibliothèque Utils, qui n'est qu'un regroupement de bibliothèques pour simplifier l'utilisation de Latex.

Chaque bibliothèque doit être indépendante afin de fonctionner correctement.

Voici les bibliothèques disponibles:

- Badges
- Colors
- Debug
- Electronic
- Fonts
- Glossaries
- Graphics
- Header
- Images
- Index
- Items
- Labels
- Layout
- Links
- Maths
- MessageBox
- Nomenclature
- Objects3D
- Pdf
- Programming

- Theorems
- Titles
- Tree

## Installation

Latex est un logiciel assez volumineux<sup>1</sup> mais l'installation complète ne nécessite pas d'ajout de paquet supplémentaires. Il est disponible dans les dépôts **Debian/Ubuntu** avec les commandes suivantes<sup>2</sup> :

```
sudo apt-get update
sudo apt-get -y upgrade
sudo apt-get install texlive-full
```

Installation de Latex

La commande suivante permet de gérer Latex en français.

```
sudo apt-get install texlive-lang-european
```

Installation des langues

### Remarque

L'ensemble des outils présenté est optimisé pour une utilisation avec le logiciel Visual Studio Code

```
sudo snap install code
```

Installation de Visual Studio Code

Enfin, l'installation de PHP permettra de générer le code d'autocomplétion pour VSCode

```
sudo apt-get install php
```

Installation du module PHP

## Organisation du projet

Chaque projet est constitué de 6 dossiers et de 3 fichiers situés à la racine du projet.

<sup>1</sup>Environ 1.5Go dans les dépôts Debian/Ubuntu

<sup>2</sup>Il faut saisir la commande dans un terminal

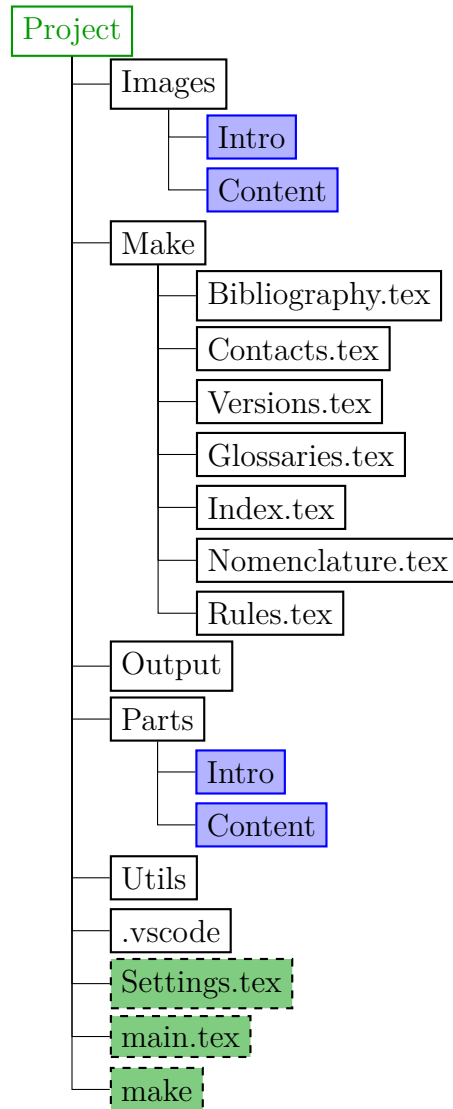


Figure 2.1: Arborescence du projet

- ▶ Le dossier DIR Images contient l'ensemble des images du projet. Chaque image doit faire partie de la même partie que son document source associé.
- ▶ Le dossier DIR Make contient les fichiers annexes du projet:
  - ▶ Le fichier FILE Bibliography.tex recense les bibliographies du projet. Le format Zotéro est compatible.
  - ▶ Le fichier FILE Contacts.tex est une page pour contacter l'auteur et contient les informations sur les droits et les licences du projet.
  - ▶ Le fichier FILE Versions.tex contient les différentes versions du projet.
  - ▶ Le fichier FILE Glossaries.tex contient le glossaire.
  - ▶ Le fichier FILE Index.tex contient l'index.



- ▶ Le fichier `FILE Nomenclature.tex` contient la nomenclature<sup>3</sup>
- ▶ Le fichier `FILE Rules.tex` contient les conventions pour le projet. Il peut contenir les types de commandes, les conventions de nommage du projet...
- ▶ Le dossier `DIR Output` contient les fichiers de compilation générés de manière automatique. **Vous n’aurez pas à modifier des fichiers à cette emplacement.**
- ▶ Le dossier `DIR Parts` contient les différentes parties du projet. Il est possible de scinder son projet en grandes parties (Introduction, Chapitre1, Chapitre2, Conclusion), chaque dossier contenu dans le dossier **Parts** représente ces parties.

Dans chacun de ces dossier, vous pouvez créer autant de fichier Latex que vous voulez, il seront compilés dans l’ordre alphabétique ou bien par ordre croissant si vous mettre un numéro au début du nom de fichier.

Pour chaque dossier crée dans le dossier `DIR Parts`, il faudra créer un dossier avec le même nom dans le dossier `DIR Images`, sous peine de voir une volée d’erreur lors de la compilation (Voir l’arborescence du projet).

- ▶ Le dossier `DIR Utils` contient les bibliothèques du projet.
- ▶ Le dossier `DIR .vscode` (dossier caché) contient les fichiers des paramètres VSCode.

Et voici les trois fichiers situés à la racine:

- ▶ Le fichier `FILE Settings.tex` regroupe les paramètres de mise en page du projet
- ▶ Le fichier `FILE main.tex` est le fichier principal du projet.
- ▶ Le fichier `FILE make` est le fichier de compilation.

## Fusion de projets

Le choix d’un dossier par partie (Parts/XXX) permet de fusionner très facilement des projets. Pour fusionner deux projets, il suffit de copier-coller le contenu du dossier `DIR Images` et `DIR Parts` du projet A dans le dossier de projet qui contiendra la fusion (projet B). Lors de la compilation, **make** va gérer la fusion automatiquement.

## Compilation du projet

### Première compilation

---

<sup>3</sup>Les unités et grandeurs physiques par exemple

La compilation du projet se fait grâce au fichier `make` situé à la racine du projet. Avant de faire la toute première compilation, il convient de rendre exécutable le fichier `make` en saisisant la commande suivante :

```
chmod +x make
```

Don des droits d'exécution sur le fichier `make`

Il ne reste plus qu'à compiler le fichier.

## Compilation classique

Une compilation classique a pour objectif de générer le fichier PDF de rendu, appelé `main.pdf` et situé à la racine du projet.

```
./make
```

Compilation du projet

Lors de la compilation, plusieurs fichiers sont générés à la racine, dont :

- ▶ Le fichier `.render_report.tex` (fichier caché) qui contient la première partie des fichiers journaux de compilation
- ▶ Le fichier `.render_report_logs.tex` qui contient la seconde partie des fichiers journaux de compilation<sup>4</sup>
- ▶ Une image `Part.png` qui affiche le nombre de ligne pour chaque fichier contenu dans le dossier **Parts**

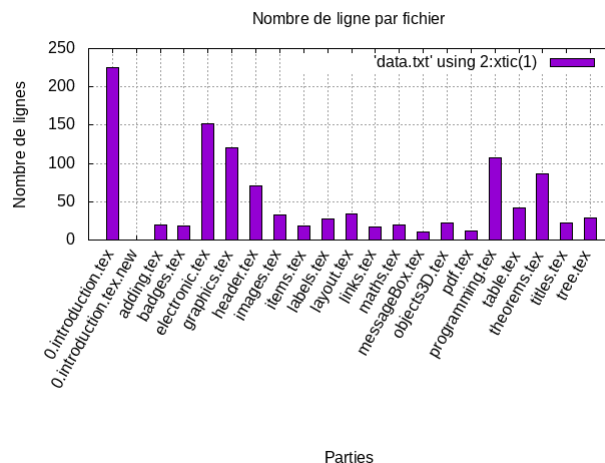


Figure 2.2: Nombre de ligne pour les parties

<sup>4</sup>Les messages de compilation générés par la bibliothèque Utils sont situés dans ce fichier.

- Une image *Utils.png* qui affiche le nombre de ligne pour chaque fichier contenu dans le dossier **Utils**

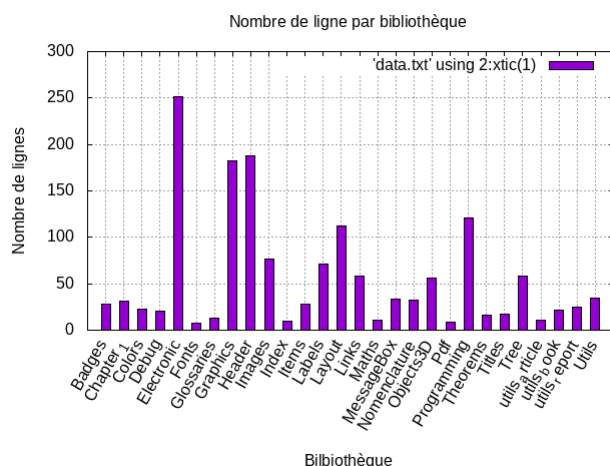


Figure 2.3: Nombre de ligne pour les bibliothèques

Lors de la compilation, différents messages s'affichent :

```
>>> Messages :
>>> Utils : Babel package is loaded
>>> Utils - [MData] : title='Tutoriel Latex
>>> Utils - [MData] : author(s)='Nicolas Le Guerroué'
>>> Utils - [MData] : subject='Bibliothèque Utils'
>>> Utils - [MData] : creator='Nicolas Le Guerroué'
>>> Utils - [MData] : keywords='Latex'
>>> Utils - [MData] : link colors='green'
>>> Utils - [MData] : bib links colors='blue'
>>> Utils - [MData] : link file colors='blue'
>>> Utils : Image 'Images/content/Part.png' [size=0.5,id 1] loaded !
>>> Utils : Image 'Images/content/Utils.png' [size=0.5,id 2] loaded !
>>> Utils : Image 'Images/content/check.png' [size=0.5,id 3] loaded !
>>> Utils : Image 'Images/content/check2.png' [size=0.5,id 4] loaded !
>>> Utils : Image 'Images/content/tux.png' [size=0.5,id 5] loaded !
>>> Utils : Image 'Images/content/tux.png' [size=0.5,id 6,angle=45] loaded !
>>> Utils : MessageBox 'Message' [id 1] created !
>>> Utils : MessageBox 'Message' [id 2] created !
>>> Utils : MessageBox 'Message' [id 3] created !

>>> Compilation terminée !
```

Figure 2.4: Message d'ajout d'éléments de la bibliothèque Utils

```
>>> Warnings :
Package Utils Warning: Image 'Images/content/Parts.png' no loaded on input line
```

Figure 2.5: Message d'avertissements

## Vérification orthographique

En invoquant le paramètre `-check`, il est possible de faire une vérification orthographique avec le logiciel aspell. Ci ce dernier n'est pas installé, il suffit de lancer la commande suivante :

```
sudo apt-get install -y aspell
```

Installation de aspell

Enfin, si vous lancer la commande

```
./make --check
```

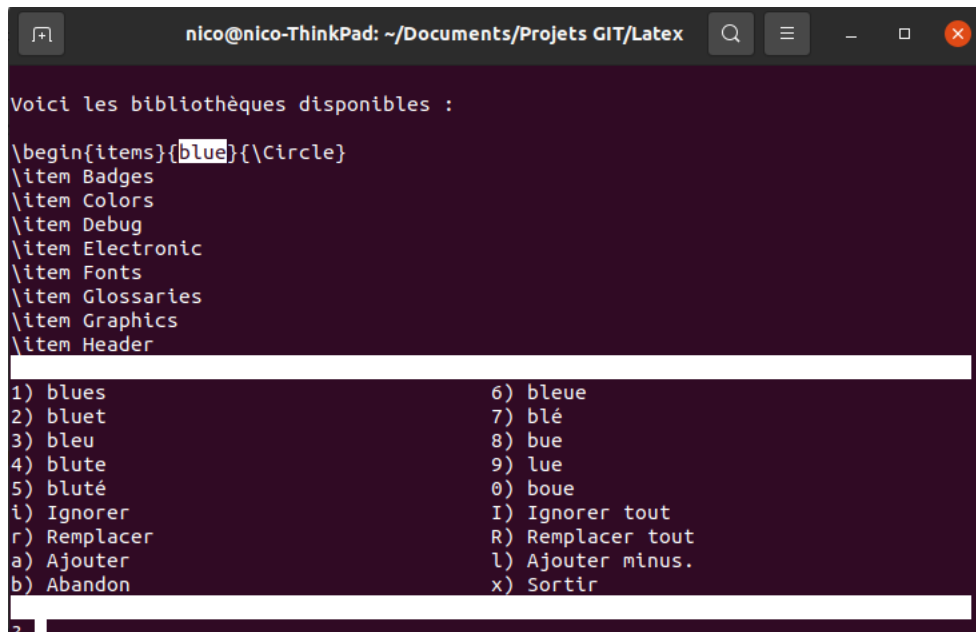
Vérification orthographique

Le fichier *make* vous demande si vous souhaitez corriger les fichiers contenus dans le dossier **Parts**.

```
(base) nico@nico-ThinkPad:~/Documents/Projets GIT/Latex$ ./make --check
Vérification orthographique du répertoire Parts...
>>> Dossier Parts/0.content en cours d'analyse !
Voulez-vous analyser le fichier Parts/0.content/0.introduction.tex ? (y/n)
```

Figure 2.6: Vérification orthographique

Veillez saisir KEY y si vous souhaitez corriger le fichier indiqué. Ensuite, il ne vous reste plus qu'à être guidé par le logiciel aspell.



```
Voici les bibliothèques disponibles :
\begin{items}{\blue}{\Circle}
\item Badges
\item Colors
\item Debug
\item Electronic
\item Fonts
\item Glossaries
\item Graphics
\item Header

1) blues                6) bleue
2) bluet                7) blé
3) bleu                 8) bue
4) blute                9) lue
5) bluté               0) boue
I) Ignorer tout        R) Remplacer tout
l) Ajouter minus.      x) Sortir
```

Figure 2.7: Commande de vérification orthographique

Les commandes sont à saisir au clavier (KEY Ctrl+I pour ignorer le mot par exemple).

## Mise à jour Git

Pour les projets Latex étant sur Git, il est possible de mettre à jour le dépôt en saississant la commande suivante :

```
./make --git
```

Mise à jour Git

## Création d'un nouveau projet

Pour créer un nouveau projet, il suffit de copier le fichier `FILE make` et de le mettre là où on souhaite créer le nouveau projet.

### Remarque

Il faut auparavant que le dossier `.utils_lib` soit situé à la racine de votre espace personnel (\$HOME)

```
./make --init
```

Nouveau projet

## Mise à jour de l'autocomplétion

Il est possible de mettre à jour l'autocomplétion sous VScode pour les bibliothèques Utils. En invoquant le paramètre `--snippet`, il est possible de générer le fichier VScode qui va ajouter l'autocomplétion.

Ce fichier est appelée `FILE output.snippet-code` et se situe dans le dossier `DIR .vscode`

```
./make --snippet
```

Mise à jour de l'autocomplétion

## Compilation avec VSCode

Toutes les commandes invoquées avec *make* sont accessibles en ouvrant le répertoire du projet avec VScode; Il suffit de se placer à la racine du projet et de faire la commande suivante:

```
code .
```

Ouverture de l'arborescence avec VSCode

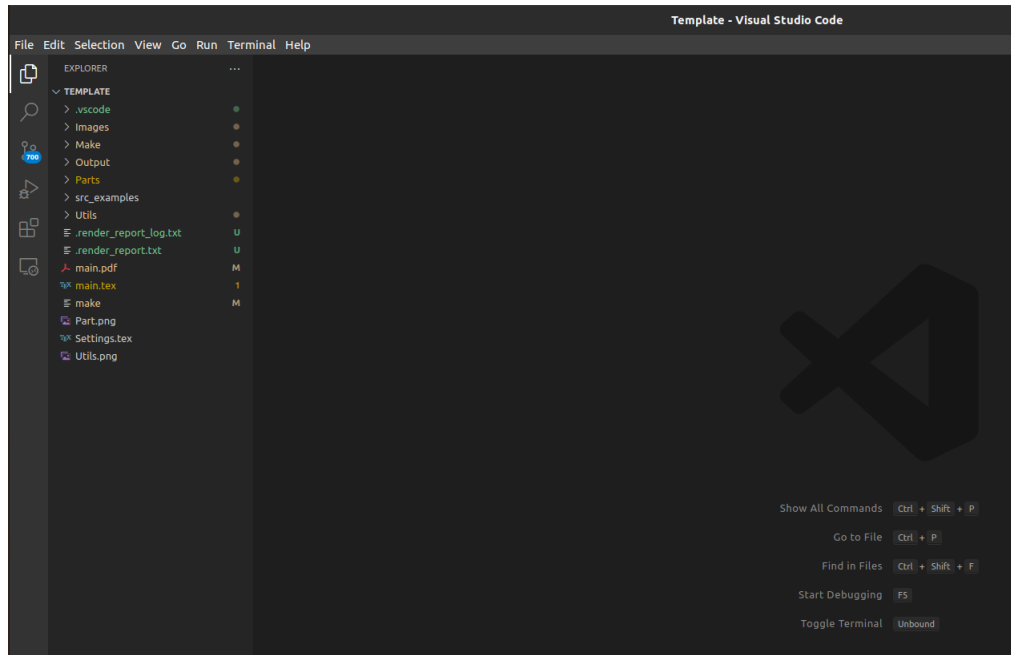


Figure 2.8: Ouverture de VSCode

### Remarque

L'ensemble des commandes et outils sont disponibles en saisisant le raccourci

**KEY** CTRT+SHIFT+B

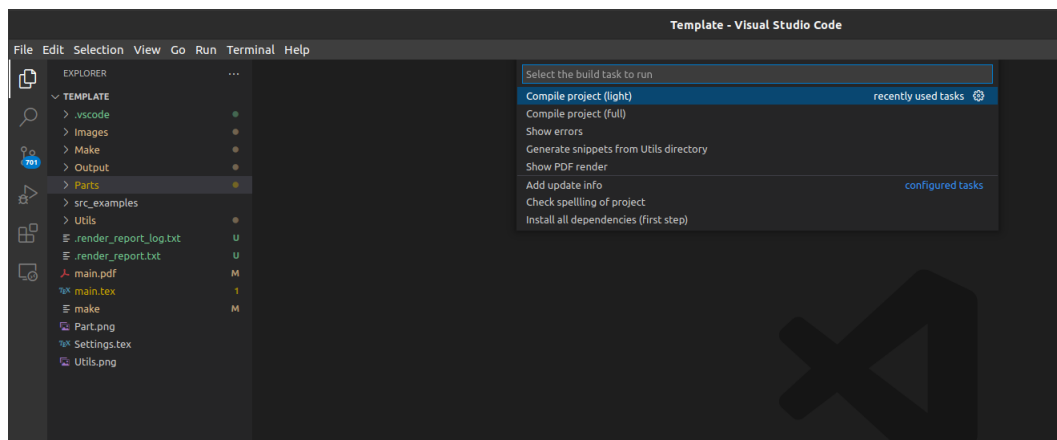


Figure 2.9: Visualisation des commandes

## Conventions

**LOC** Header veut dire que le code est à mettre avant `\begin{document}`

**LOC** Body veut dire que le code est à mettre entre `\begin{document}` et `\end{document}`

La Bibliothèque **Adding** permet de générer des nomenclatures.

## Création d'une nomenclature

 Body

```
\nomenclature[E]{$r$}{Rapport cyclique d'un signal périodique}
\nomenclature[A]{$A_d$}{Coefficient d'amplification, gain différentiel }
\nomenclature[A]{$\varepsilon$}{Tension différentielle $(\varepsilon = E_+ - E_-)$
  $\addUnit{V}$}
\nomenclature[A]{$E_+$}{Tension entrée non inverseuse \addUnit{V}}
\nomenclature[A]{$E_-$}{Tension entrée inverseuse \addUnit{V}}
\nomenclature[E]{$\eta$}{Rendement d'un mécanisme \addUnit{\%}}
\nomenclature[E]{$\varphi$}{Déphasage entre deux signaux \addUnit{rad}}
```

Code pour la création d'une nomenclature

## Création de badges avec des couleurs

Electronique

Mécanique

Informatique

Loc Body

```
\badge{white}{black}{Electronique}  
\badge{white}{blue}{Mécanique}  
\badge{white}{green}{Informatique}
```

Code pour la création de badges avec des couleurs



La bibliothèque **Electronic** permet de générer des chronogrammes et des schémas électriques

## Création de chronogrammes fixes

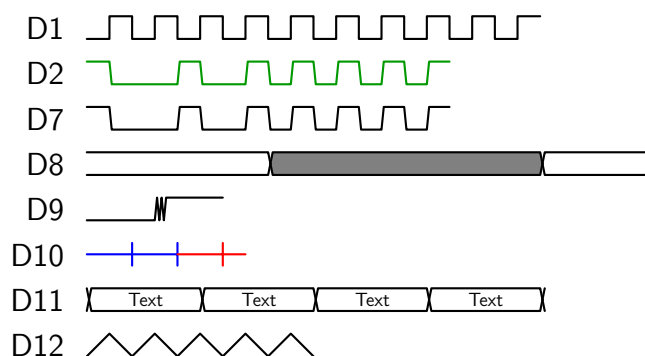


Figure 5.1: Exemple 1 chronogramme fixe

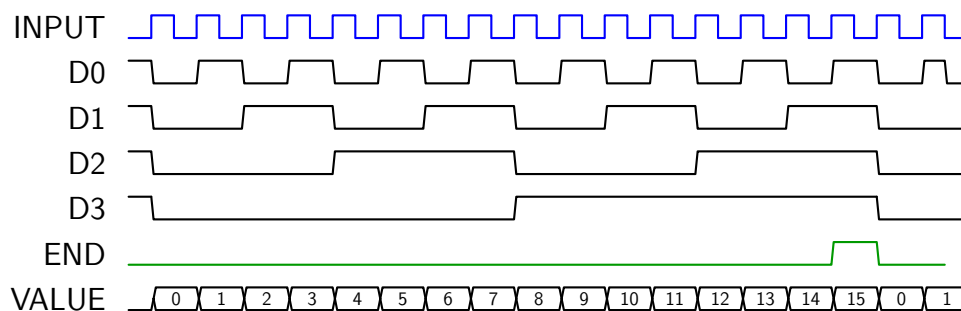


Figure 5.2: Exemple 2 - Chronogramme du compteur 4 bits

**loc** Body

```
\begin{numeric}{exemple 1 - chronogramme fixe}
  D1 & 20{C}  \\
  D2 & [green] 1H1L1L1L1H1L1L1H1L1H1L1H1L1H  \\
  D7 & [black] 1H1L1L1L1H1L1L1H1L1H1L1H1L1H  \\
  D8 & 8D5U7U5D  \\
  D9 & LLL 2{0.1H 0.1L} 0.6H HH  \\
  D10 & ZZ G ZZ G XX G X  \\
  D11 & [d] 4{5D{Text}} 0.2D  \\
  D12 & [L][timing/slope=1.0] HL HL HL HL HL  \\
\end{numeric}
```

Code pour la création de chronogrammes fixes [exemple 1]

**loc** Body


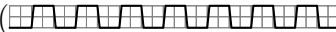
```

\begin{numeric}{Exemple 2 - Chronogramme du compteur 4 bits}
  INPUT & CC [blue]16{CC} CCC \\
  D0 & HL 8{LHHL} LHL \\
  D1 & H 4{LLLLHHHH} LLLL \\
  D2 & H 2{LLLLLLLLHHHHHHHH} LLLL \\
  D3 & H{LLLLLLLLLLLLLLLLHHHHHHHHHHHHHHHHHH} LLLL \\
  END & LL [green]14{LL} LHHLLL \\
  VALUE & L 2D{0} 2D{1} 2D{2} 2D{3} 2D{4} 2D{5} 2D{6} 2D{7} 2D{8} 2D{9} 2D{10}
  2D{11} 2D{12} 2D{13} 2D{14} 2D{15} 2D{0} 2D{1} \\
\end{numeric}%

```

Code pour la création de chronogrammes fixes [exemple 2]

## Création de chronogrammes flottants

Notre signal d'horloge () provient d'un oscillateur à quartz. Notre signal d'horloge () provient d'un oscillateur à quartz.

Loc Body

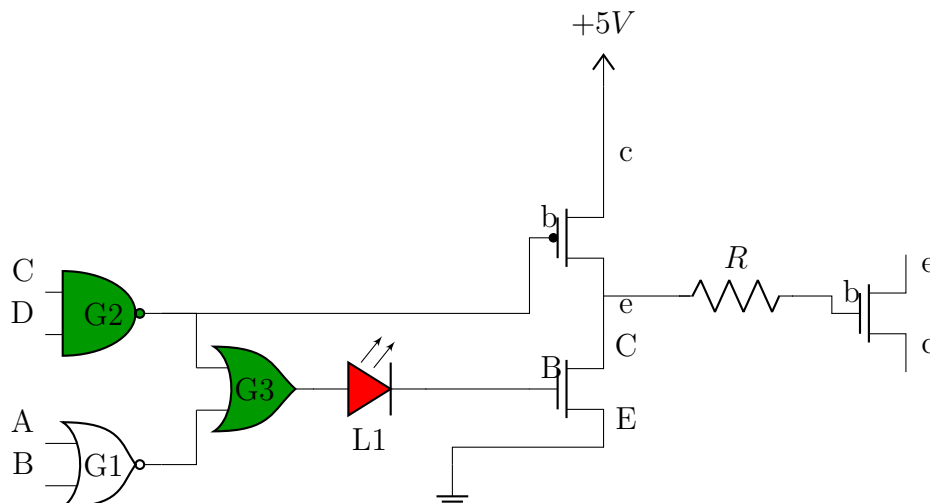
```

Notre signal d'horloge (\texttt{timing}{[blue]CCCCC}) provient d'un oscillateur à
quartz.
Notre signal d'horloge (\texttt{timing}[timing/draw grid]{LHLHLHLHLHLHL})
provient d'un oscillateur à quartz.

```

Code pour la création de chronogrammes flottants

## Création de schémas électriques



loc Body

```

\begin{schema} {Exemple de schéma électrique}

  \addPower{6,5}{power1}{${+5V$}}
  \addGround{4,0}{gnd1}{}

  \setDeviceBackgroundColor{white}
  \setRotate{0}
  \addLogicGate{0,0}{mynor}{nor}{}{A}{B}{G1}

  \setDeviceBackgroundColor{green}
  \addLogicGate{0,2}{mynand}{nand}{}{C}{D}{G2}
  \addLogicGate{2,1}{myor}{or}{}{}{}{G3}
  \resetColors

  \addTransistor{6,1}{npnA}{nmos}{B}{C}{E}
  \addTransistor{6,3}{pnpA}{pmos}{b}{e}{c}

  \resetColors
  \addTransistor{10,2}{npnR}{nmos}{b}{e}{c}

  \addWire{mynor.out}{myor.in 2}{\orthogonalWireA}
  \addWire{mynand.out}{myor.in 1}{\orthogonalWireA}

  \addWire{mynand.out}{pnpA.B}{\orthogonalWireA}
  \addWire{pnpA.C}{npnA.C}{\orthogonalWireA}

  \addWire{pnpA.E}{power1}{\orthogonalWireA}

  \addWire{npnA.E}{gnd1}{\orthogonalWireA}

  \addNode{$(pnpA.C)+(1,0)$}{node1}{}
  \addWire{pnpA.C}{node1}{\orthogonalWireA}

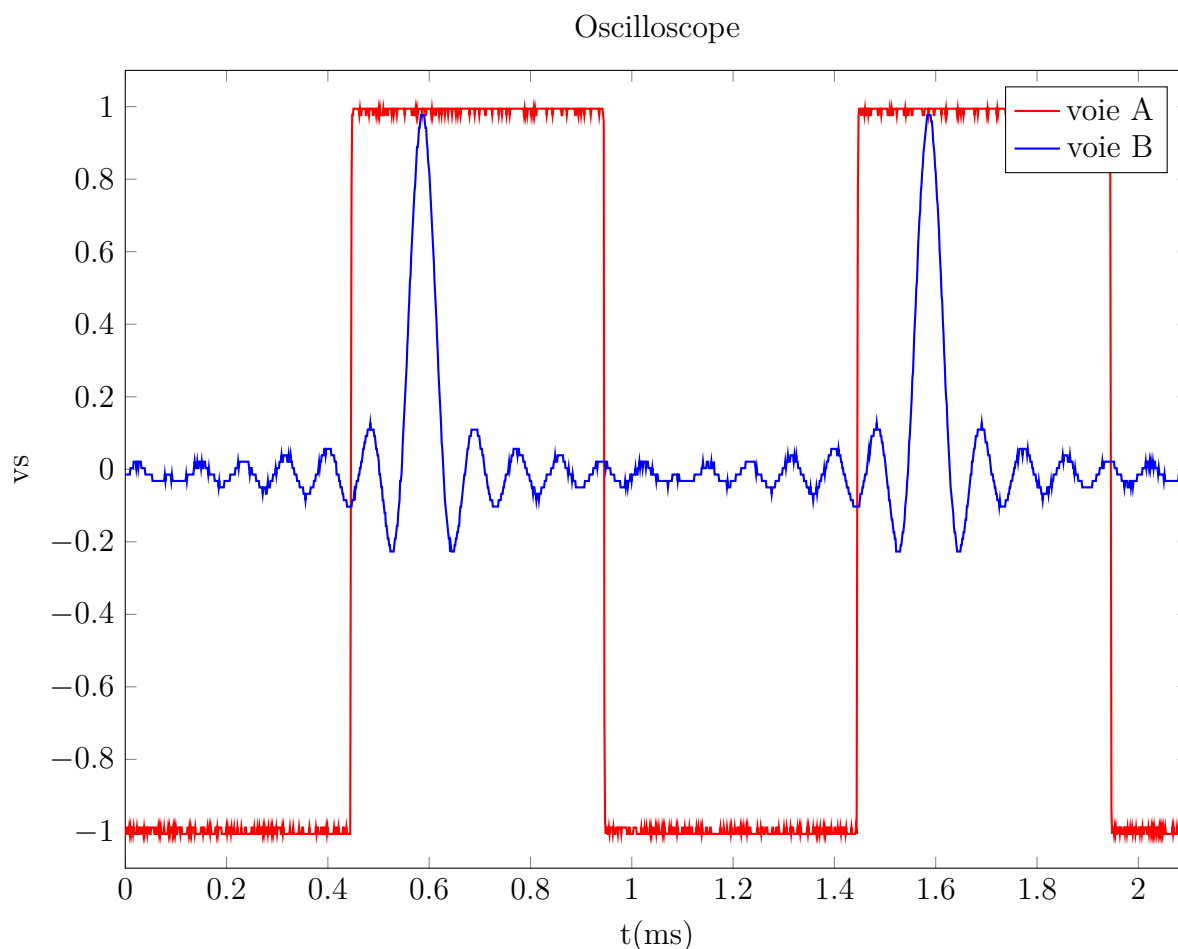
  \setDeviceBackgroundColor{red}
  \addLed{myor.out}{\Right}{npnA.B}{\orthogonalWireA}{L1}
  \addResistor{node1}{\Right}{npnR.B}{\orthogonalWireA}

\end{schema}

```

Code pour la création de schémas électriques

## Affichage d'un graphique 2D avec insertion des données depuis un fichier txt (csv)

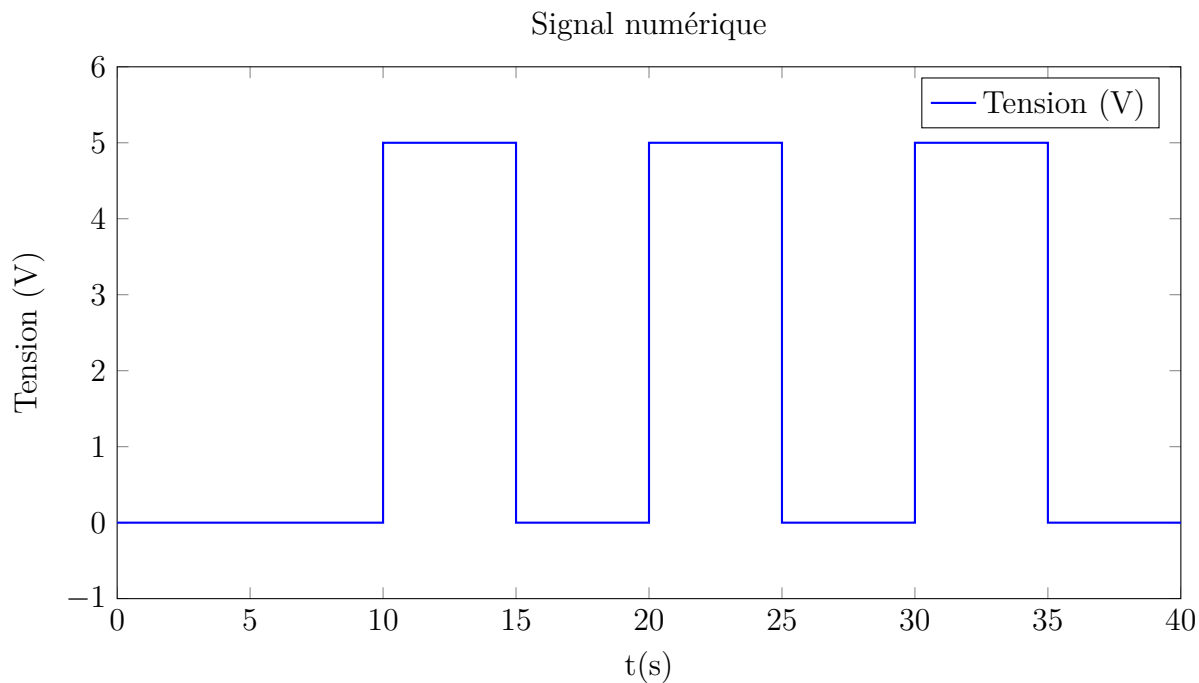


**loc** Body

```
\begin{graphic}{0.8}{0.6}{0}{2.1}{-1.1}{1.1}{t(ms)}{vs}{Oscilloscope}
\addPointsFromCSV{red}{comma}{src_examples/input_1.txt}
\addPointsFromCSV{blue}{comma}{src_examples/input_2.txt}
\addLegend{voie A, voie B}
\end{graphic}
```

Code pour l'affichage d'un graphique 2D avec insertion des données depuis un fichier txt (csv)

## Affichage d'un graphique 2D avec insertion des données depuis une liste de points



**LOC** Body

```
\begin{graphic}{0.8}{0.4}{0}{40}{-1}{6}{t(s)}{Tension (V)}{Signal numérique}
  \addPoints{blue}{(0,0)(10,0)(10,5)(15,5)(15,0)(20,0)(20,5)(25,5)(25,0)(30,0)
    (30,5)(35,5)(35,0)(100,0)}
  \addLegend{Tension (V)}
\end{graphic}
```

Code pour l'affichage d'un graphique 2D avec insertion des données depuis une liste de points

## Affichage d'un graphique 2D avec insertion des données depuis une équation

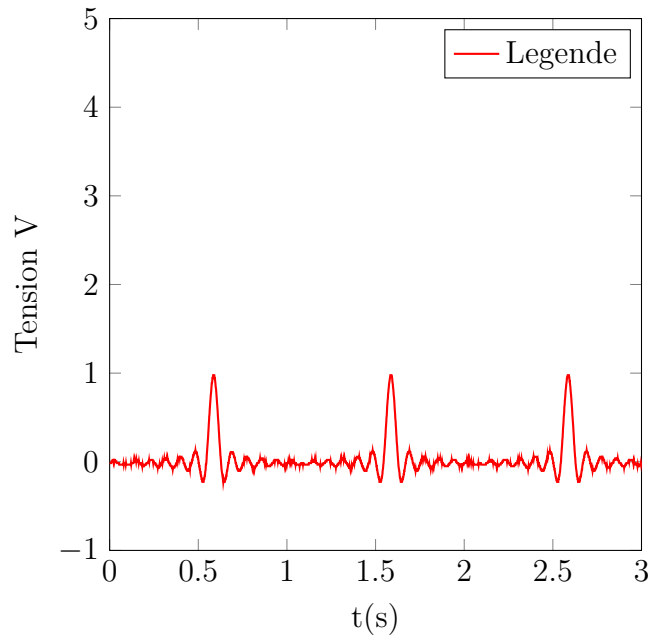


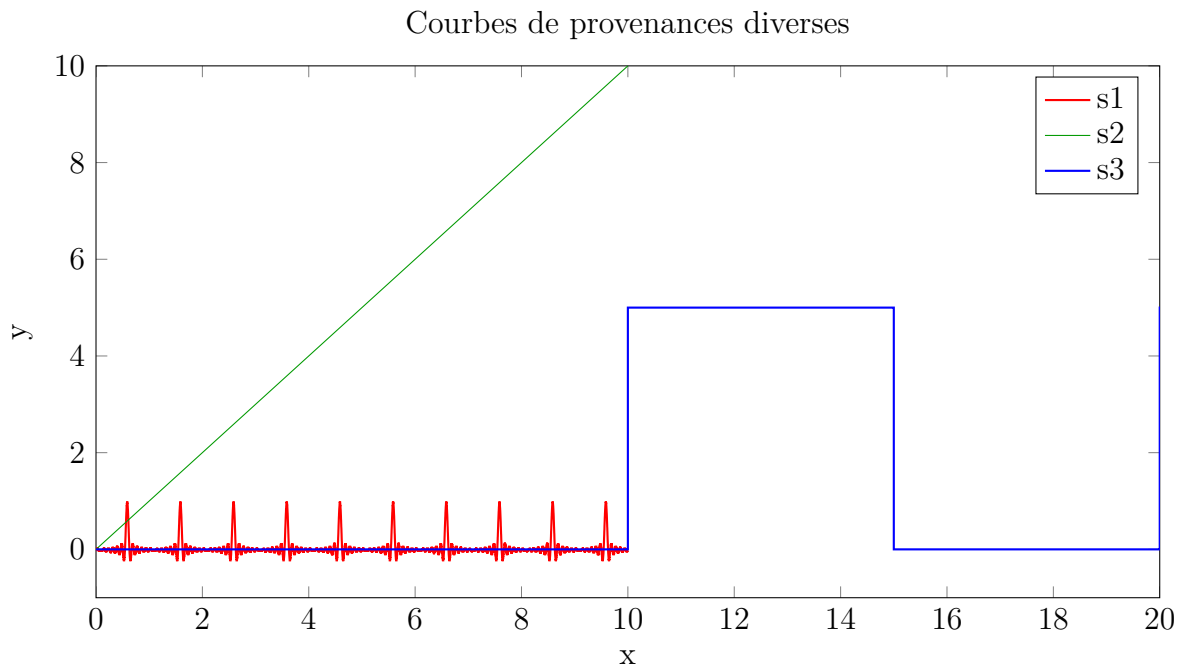
Figure 6.1: Signal analogique

**LOC** Body

```
\begin{graphicFigure}{0.4}{0.4}{0}{3}{-1}{5}{t(s)}{Tension V}{Signal analogique}  
  \addPointsFromCSV{red}{comma}{src_examples/input_2.txt}  
  \addLegend{Legende}  
\end{graphicFigure}
```

Code pour l’affichage d’un graphique 2D avec insertion des données depuis une liste de points

## Affichage d’un graphique 2D avec insertion des données depuis plusieurs sources



**Loc** Body

```
\begin{graphic}{0.8}{0.4}{0}{20}{-1}{10}{x}{y}{Courbes de provenances diverses}
\addPointsFromCSV{red}{comma}{src_examples/input_2.txt}
\addTrace{green}{-10}{10}{x}
\addPoints{blue}{(0,0)(10,0)(10,5)(15,5)(15,0)(20,0)(20,5)(25,5)(25,0)(30,0)
(30,5)(35,5)(35,0)(100,0)}
\addLegend{s1,s2,s3}
\end{graphic}
```

Code pour l'affichage d'un graphique 2D avec insertion des données depuis plusieurs sources

## Affichage de deux graphiques

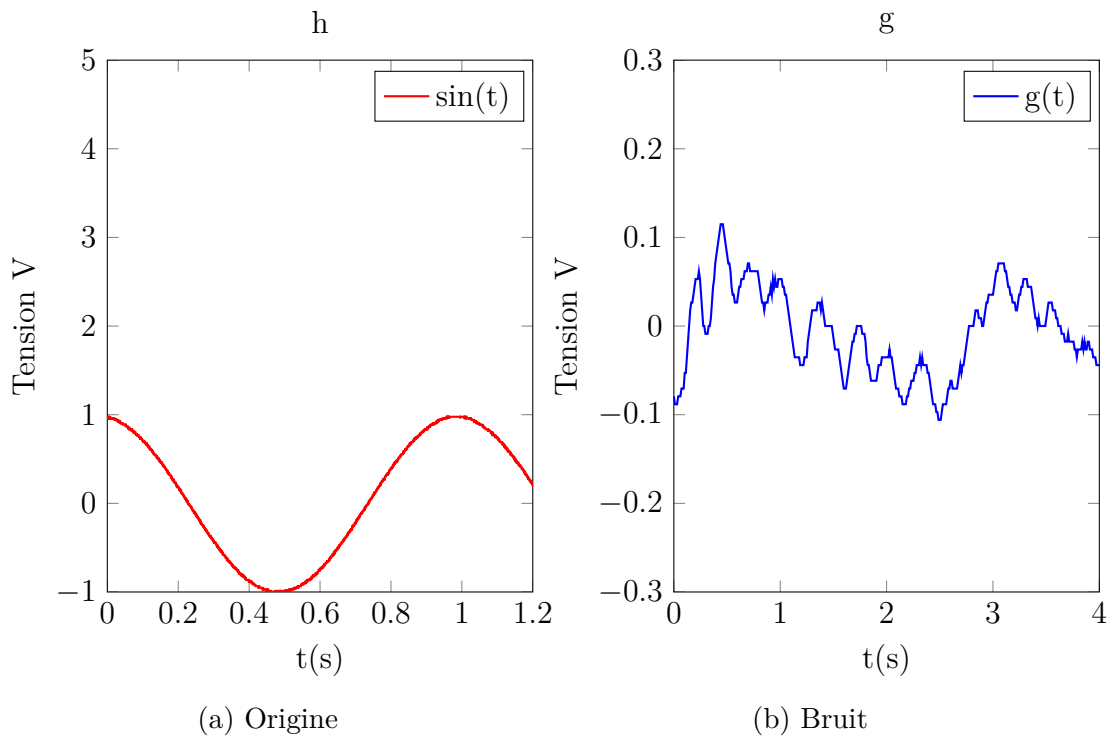


Figure 6.2: Les tensions de service

loc Body

```
\begin{figure}[h]
\centering
\begin{subfigure}[b]{0.4\linewidth}
\begin{graphic}{0.8}{1}{0}{1.2}{-1}{5}{t(s)}{Tension V}{h}
\addPointsFromCSV{red}{comma}{src_examples/sinus.txt}
\addLegend{sin(t)}
\end{graphic}%NO END LINE HERE
\caption{No interaction}
\end{subfigure}
\begin{subfigure}[b]{0.4\linewidth}
\begin{graphic}{0.8}{1}{0}{4}{-0.3}{0.3}{t(s)}{Tension V}{g}
\addPointsFromCSV{blue}{comma}{src_examples/jack01.txt}
\addLegend{g(t)}
\end{graphic}%NO END LINE HERE
\caption{Interaction}
\end{subfigure}
\caption{Les tensions de service}
```



```
\end{figure}
```

Code pour l’affichage d’un graphique 2D avec insertion des données depuis plusieurs sources

## Mise en forme de la page de garde avec une image

LOC Header

```
\setHeaderImage{Emplacement_image}{0.8}{Titre}{sous-titre}{Auteurs}{\today \\\  
pageref{LastPage} pages}
```

Code pour la mise en forme de la page de garde avec une image

## Mise en forme de la page de garde sans image

LOC Header

```
\setHeader{Titre}{Auteur 1 \\\ Auteur 2}{Date}
```

Code pour la mise en forme de la page de garde sans image

## Mise en forme de la page des parties

LOC Body

```
\partImg{Partie}{Images/file.png}{0.2}
```

Code pour la mise en forme de la page des parties

## Ajout d'un trait entre l'en-tête et le corps de la page

LOC Header

```
\setHeaderLine{0.2}
```

Code pour l'ajout d'un trait entre l'en-tête et le corps de la page

## Ajout d'un trait entre le corps de la page et le bas de page

LOC Header

```
\setFooterLine{0.2}
```

Code pour l'ajout d'un trait entre le corps de la page et le bas de page

## Définition de la présentation globale des pages

**LOC** Header

```
\addPresentation  
{Titre} {Centre} {\currentChapter}  
{Gauche} {} {\currentPage}
```

Code pour la définition de la présentation globale des page

## Redéfinition des titres des chapitres

**LOC** Header

```
\setAliasChapter{Section}
```

Code pour la redéfinition des titres des chapitres

## Mettre le document en pleine page

**LOC** Header

```
\setFullPage
```

Code pour mettre le document en pleine page

## Récupérer le chapitre courant

**LOC** Body

```
\currentChapter
```

Code pour récupérer le chapitre courant

## Ajout d'une image non-flottante



Figure 8.1: Légende de l'image

**loc** Body

```
\img{\rootImages/tux.png}{Légende de l'image}{0.5}
```

Code pour l'ajout d'une image non-flottante

## Ajout d'une image non-flottante avec une rotation



Figure 8.2: Légende de l'image

**loc** Body

```
\img{\rootImages/tux.png}{Légende de l'image}{0.5}{45}
```

Code pour l'ajout d'une image non-flottante avec une rotation

## Création d'un liste

- ▶ A
- ▶ B
- ▶ C

log Body

```
\begin{items}{orange}{\Triangle}  
  \item A  
  \item B  
  \item C  
\end{items}
```

Code pour la création d'une liste

## Options

- ▶ Triangle
- ▶ Circle
- ▶ Bullet

## Création de labels colorés



LOC Body

```
\lorange{LIB}{OUT} %Label orange
\lred{LIB}{OUT} %Label rouge
\lgreen{LIB}{OUT} %...
\lmagenta{LIB}{OUT}
\lpurple{LIB}{OUT}
\lcyan{LIB}{OUT}
\lblue{LIB}{OUT}
\lbrown{LIB}{OUT}
\lyellow{LIB}{OUT}
\lblack{LIB}{OUT}
```

Code pour la création de labels colorés

La mise en page est séparée en 4 parties : frontmatter (début du document, numérotation romaine), mainmatter (avant le premier chapitre), appendix (annexes) et backmatter avant les tables et bibliographies

## Mise en gras

Texte en gras

LOC Body

```
\bold{Texte en gras}
```

Code pour mettre le texte en gras

## Mise en italique

*Texte en italique*

LOC Body

```
\italic{Texte en gras}
```

Code pour mettre le texte en italique

## Mise en gras et italique

*Texte en gras et italique*

LOC Body

```
\ib{Texte en gras et italique}
```

Code pour mettre le texte en gras et italique

## Ajout d'un espace vertical

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu li-

bero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestib-

`\sn` Body

`\sn`

Code pour ajouter un espace vertical



## Paramétrage des liens et des méta-données

### Header

```
%@input Titre du PDF
%@input Auteur(s)
%@input Sujet du fichier PDF (courte phrase)
%@input Créateur du fichier PDF
%@input Producteur du fichier PDF
%@input Mots-clés (liste)
%@input Couleurs des liens
%@input Couleurs des citations dans la bibliographie
%@input Couleurs des liens de fichier
\setParameters {Tutoriel Latex} {Nicolas Le Guerroué} {Bibliothèque Utils} {
  Nicolas Le Guerroué}{Latex}{green}{blue}{blue}
```

Code pour paramétrer les liens et les métadonnées

## Création d'une matrice 3\*3

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

**LOC** Body

```
$$\emat{a & b & c}{d & e & f}{g & h & i} $$
```

Code pour la création d'une matrice 3\*3

## Création d'un vecteur à trois dimensions

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

**LOC** Body

```
$$\evect{a}{b}{c} $$
```

Code pour la création d'un vecteur à trois dimensions

## Création d'un torseur à trois dimensions

$$\begin{Bmatrix} a & d \\ b & e \\ c & f \end{Bmatrix}$$

**LOC** Body

```
$$\torsor{a & d}{b & e}{c & f} $$
```

Code pour la création d'un torseur à trois dimensions

## Création de boîtes de dialogues

Message

Voici un message

Message

Et un autre message

Message

bref...

loc Body

```
\messageBox{Message}{orange}{white}{Voici un message}{black}
```

Code pour la création de boîtes de dialogues

## Insertion d'un document PDF

LOC Body

```
\includepdf[page=1,2,3]
```

Code pour ajouter un document au format PDF

## Insertion d'un ensemble de pages d'un document PDF

LOC Body

```
\includepdf[page=1,2,3]
```

Code pour ajouter un ensemble de page d'un document au format PDF

## Affichage d'un code C/C++ avec titre

```
#include <iostream>

#define CONST 1

int var = 1;
float

int main() {

    call();
    return 0;

} //End main
```

Titre

**loc** Body

```
\begin{Cpp}{Titre}
    \include <iostream>

    \define CONST 1

    int var = 1;
    float

    int main() {

        call();
        return 0;

    } //End main

\end{Cpp}
```

Code pour l'affichage d'un code C/C++ avec titre

## Affichage d'un code C/C++ sans titre

```
#define CONST 1 #const var

int var = 1;
float g = 2.5;
...
```

loc Body

```
\begin{Cpp}
  \include <iostream>

  #define CONST 1 #const var

  int var = 1;
  float g = 2.5;
  ...

\end{Cpp}
```

Code pour l’affichage d’un code C/C++ sans titre

## Affichage d’un code Python avec titre

```
def call(input):

    """docstring"""
    a = input
    for elem in a:
        print(elem) #show
```

Titre du code

loc Body

```
\begin{Python}{Titre du code}
  def call(input):

      """docstring"""
      a = input
      for elem in a:
          print(elem) #show
\end{Python}
```

Code pour l’affichage d’un code Python avec titre

## Affichage d’un code Python sans titre

```
"""docstring"""
...
```

**loc** Body

```
\begin{Python}
  def call(input):

    """docstring"""
    ...
\end{Python}
```

Code pour l’affichage d’un code Python sans titre

## Affichage d’un code Bash avec titre

```
sudo apt-get -y update
sudo apt-get -y upgrade
echo -e "content"
```

Titre du code

**loc** Body

```
\begin{Bash}{Titre du code}
  sudo apt-get -y update
  sudo apt-get -y upgrade
  echo -e "content"
\end{Bash}
```

Code pour l’affichage d’un code Bash avec titre

## Affichage d’un code bash sans titre

Ce type d’affichage n’est pas encore supporté par la bibliothèque.

$U_A$ (V)	$U_B$ (V)	Sens du courant	$U_A - U_B$
10	5	De A vers B	5
5	10	de B vers A	-5
5	5	Aucun courant ne circule	0

Figure 17.1: Réponse sur le sens du courant en fonction des tensions  $U_A$  et  $U_B$ 

loc

Body

```

\begin{figure}[!h]
  \centering
  \begin{tabular}{|c|c|c|c|}
    \hline
    $U_A$ (V) & $U_B$ (V) & Sens du courant & $U_A-U_B$\\
    \hline
    10 & 5 & \colors{blue}{De A vers B} & 5\\
    \hline
    5 & 10 & \colors{blue}{de B vers A} & -5\\
    \hline
    5 & 5 & \colors{blue}{Aucun courant ne circule} & 0\\
    \hline
  \end{tabular}
  \caption{Réponse sur le sens du courant en fonction des tensions $U_A$ et $U_B$}
\end{figure}

```

Code d'exemple



## Création d'une question

**Question 1.** *Quelle heure est-il ?*

 Body

```
\begin{question}
  Quelle heure est-il ?
\end{question}
```

Code pour la création d'une question

## Création d'une reponse

>>> **1.** *il est 18 h.*

 Body

```
\begin{reponse}
  il est 18 h.
\end{reponse}
```

Code pour la création d'une reponse

## Création d'une propriete

**Propriété 1.** *Un produit scalaire est commutatif.*

 Body

```
\begin{propriete}
  Un produit scalaire est commutatif.
\end{propriete}
```

Code pour la création d'une propriete

## Création d'une proposition

**Proposition 1.** *Les chats sont des mammifères.*

 Body

```
\begin{proposition}
  Les chats sont des mammifères.
\end{proposition}
```

Code pour la création d'une proposition

## Création d'une remarque

**Remarque 1.** *remarque sur Latex*

 Body

```
\begin{remarque}
remarque sur Latex
\end{remarque}
```

Code pour la création d'une remarque

## Création d'un exemple

**Exemple 1.** *Ceci est un exemple d'exemple*

 Body

```
\begin{exemple}
  Ceci est un exemple d'exemple
\end{exemple}
```

Code pour la création d'une exemple

## Création d'une définition

**Definition 1.** *Une phrase est un ensemble de mots.*

 Body

```
\begin{definition}
  Une phrase est un ensemble de mots.
\end{definition}
```

Code pour la création d'une definition

## Titre de chapitre

LOC Body

```
\chapter{Titre}
```

Code pour l'ajout d'un titre

## Titre de section

LOC Body

```
\section{Titre de section}
```

Code pour l'ajout d'une section

## Titre de sous-section

LOC Body

```
\subsection{Titre de sous-section}
```

Code pour l'ajout d'une sous-section

## Titre de sous-sous-section

LOC Body

```
\subsection{Titre de sous-sous-section}
```

Code pour l'ajout d'une sous-sous-section