How to compile and work with `mazhe.tex` ?

Laurent Claessens

September 29, 2015

Contents

Tout commence par télécharger les sources à l'adresse

`https://github.com/LaurentClaessens/mazhe`

Pour information le numéro du commit qui correspond à ce qui a été envoyé à l'agrégation en septembre 2015 est `c038d9eee475b3a1d6e5414a698d50007dc69af0`.

1 Pour l'agrégation

Le plus facile est de télécharger la paquet `exocorr.sty` (voir plus bas) puis de compiler avec :

```
pdflatex -synctex=1 -shell-escape Inter_frido-mazhe_pytex.tex
```

et quelque fois `bibtex` et `makeindex`.

Ce faisant vous n'aurez cependant pas les moyens de changer l'ordre des chapitres [1] parce que ce fichier est créé automatiquement par quelque scripts de précompilation. En travaillant de la sorte vous aurez la possibilité d'effectuer des modifications dans les fichiers inclus.

Vu que de mon côté ce fichier est recréé automatiquement à chaque compilation, un `git pull` écrasera les modifications que vous y auriez apporté. Si vous voulez savoir la raison de ce fait, faites

```
pdflatex mazhe.tex
```

Voyant le résultat vous comprendrez pourquoi compiler le document destiné à l'agrégation demande un peu de travail.

1. Ou en tout cas pas de me soumettre vos changements parce que c'est dans `mazhe.tex` que je travaille.

1.1 Ce qu'il faut télécharger

Pour avoir une maîtrise plus fine de la compilation, vous devrez télécharger un certain nombre de choses.

Le paquet `exocorr` Vous devez récupérer le paquet `exocorr` à l'adresse

`https://github.com/LaurentClaessens/exocorr`

Seul le fichier `.sty` vous intéresse a priori. Mettez-le là où vous mettez vos paquets LaTeX.

Le module `LaTeXparser` Vous le téléchargez à l'adresse

`https://github.com/LaurentClaessens/LaTeXparser`

et vous le mettez quelque part là où Python pourra le retrouver.

Le script `pytex` Il est à l'adresse

`https://github.com/LaurentClaessens/pytex`

1.2 Compiler tout le document

Lorsque tout est téléchargé et correctement configuré (`LaTeXparser` doit être trouvable par python et `pytex` trouvable par votre terminal), vous compilez le Frido avec

```
pytex lst_frido.py --lotex
```

Le script s'occupe d'extraire de `mazhe.tex` les choses nécessaires au Frido, crée un fichier intermédiaire et le compile. Des passes de `bibtex` et `makeindex` sont également automatiquement effectuées.

Les `ref` et `eqref` ne correspondant à aucun `label` sont indiqués. Il ne devrait y en avoir aucun.

La compilation produit deux fichiers `pdf`. Le premier est `Inter_frido-mazhe_pytex.pdf` qui est créé par LaTeX lui-même durant la compilation. Le second est `0-lefrido.pdf` qui en est une simple copie effectuée après la compilation. Vous devriez ouvrir `0-lefrido.pdf` de façon à éviter que votre lecteur de `pdf` ne se mette en mode «rafraichissement» durant toute la durée de la compilation.

1.3 Compiler seulement une partie

Lisez le fichier `lst_exemple.py` :

lst_exemple.py

```
#! /usr/bin/python
# -*- coding: utf8 -*-

from __future__ import unicode_literals

import LaTeXparser
import LaTeXparser.PytexTools
import commons
import plugins_agreg

myRequest = LaTeXparser.PytexTools.Request("mesure")
myRequest.ok_hash=commons.ok_hash

myRequest.add_plugin(plugins_agreg.set_isAgreg,"before_pytex")
```

```
myRequest.original_filename="mazhe.tex"

myRequest.ok_filenames_list=["e_mazhe"]
myRequest.ok_filenames_list.extend(["gardeFrido"])
myRequest.ok_filenames_list.extend(["43_mesure"])
myRequest.ok_filenames_list.extend(["56_espace_vecto_norme"])
myRequest.ok_filenames_list.extend(["<++>"])
myRequest.ok_filenames_list.extend(["<++>"])
myRequest.ok_filenames_list.extend(["<++>"])
myRequest.ok_filenames_list.extend(["<++>"])
myRequest.ok_filenames_list.extend(["134_choses_finales"])


myRequest.new_output_filename="0-exemple.pdf"
```

A priori la seule chose qui vous intéresse est la liste des `ok_filename`. Je crois qu'elle est assez auto-explicative. Le fichier principal `mazhe.tex` contient une série de `input`. Seuls ceux de la liste seront effectués.

La ligne `new_output_filename` donne le nom du fichier de sortie. En l'omettant, ce sera un nom compliqué du type `0-Inter_...`. Pour compiler :

```
pytex lst_exemple.py --lotex
```

Le `--lotex` est seulement là pour dire qu'il faut faire tourner la compilation autant de fois qu'il faudra pour que les références soient justes [2].

Après compilation, une liste des références incorrectes est donnée. Bien entendu si vous ne compilez qu'une partie, il risque d'y avoir beaucoup de références *manquantes*, mais il ne devrait n'y avoir aucune références *duplicate* !

2 Les politiques éditoriales

Certaines parties de ce texte ne respectent pas les politiques éditoriales. Ce sont des erreurs de jeunesse, et j'en suis le premier triste.

2.1 Licence libre

Je crois que c'est clair.

2.2 pdflatex

Tout est compilable avec pdfLaTeX. Pas de pstricks, de psfrag ou de ps<quoiquecesoit>.

2.3 utf8

Je crois que c'est clair.

2.4 Notations

On essaie d'être cohérent dans les notations et les conventions. Pour la transformée de Fourier par exemple, je crois que la définition du produit scalaire dans L^2, des coefficients de Fourier, de la transformation et de la transformation inverse sont cohérents. Cela demande, lorsqu'on suit un livre qui ne suit pas les conventions utilisées ici, de convertir parfois massivement.

2. Pour que la bibliographie soit correcte, il faut encore compiler une fois ou deux.

2.5 De la bibliographie

On évite d'écrire en haut de chapitre «les références pour ce chapitre sont ...». Il est mieux d'écrire au niveau des théorèmes entre parenthèse, les références. De préférence en ligne.

2.6 Faire des références à tout

Lorsqu'un utilise le théorème des accroissements finis, il ne faut pas écrire «d'après le théorème des accroissements finis, blablabla». Il faut écrire un `\ref{}` explicite vers le résultat. Cela alourdit un peu le texte, mais lorsqu'on joue avec un texte de plus de 1000 pages, il est parfois vraiment laborieux de trouver le résultat qu'on cherche (surtout si il existe plusieurs version d'un résultat et que l'on veut faire référence à une version particulière).

2.7 Pas de références vers le futur

Dans le Frido, *aucune* preuve ne peut faire une référence vers un résultat prouvé plus bas. On n'utilise pas le théorème 10 dans la démonstration du théorème 7. Cela est une contrainte forte sur le découpage en chapitre et sur l'ordre de présentation des matières.

Il est bien entendu accepté et même encouragé de mettre des notes du type «Nous verrons plus loin un théorème qui ...». Tant que ce théorème n'est pas *utilisé*, ça va. Il y a dans le texte plusieurs listes de résultats «analogues». Par exemple au début du chapitre sur les séries de Fourier, une liste de références vers les différents théorèmes qui disent que la série de Fourier de f converge vers f. Je crois que ces listes sont bien et qu'il faut en faire plein.

En faisant

```
pytex lst_frido.py --verif
```

vous aurez une liste des références vers le bas. Cette liste doit être vide ! Ce programme cherche tous les `\ref{}` et `\eqref{}` ainsi que les `\label{}` correspondants et vous prévient lorsque le `\label{}` est après le `\ref{}`.

Si vous pensez qu'une référence pointée doit être accepté (par exemple parce c'est dans une des listes de résultats), alors vous ajoutez son hash dans la liste du fichier `commons.py`. Si il s'agit vraiment d'une référence vers un résultat que vous utilisez, alors vous devez déplacer des choses. Soit votre résultat vers le bas, soit celui que vous utilisez vers le haut. Parfois cela demande de déplacer ou redécouper des chapitres entiers...Si il n'y a vraiment pas moyen, bravo, vous venez de prouver que la mathématique est logiquement inconsistante.