1 Présentation

Le projet CPGEKIT est un kit LaTeX qui permet de mettre en forme tous les types de documents utilisés dans les CPGE. Essentiellement, il propose deux fichiers de classe (utilisables avec la commande `\documentclass`). Le premier, `cpgedoc.cls`, est dédié au traitement des documents de cours et des planches d'exercices. Le second, `cpgedev.cls`, est dédié au traitement de tout type de devoir, avec possibilité d'y intégrer des quiz (en stade expérimental).

> **N.B.** Couramment seul `cpgedev.cls` est utilisable. La classe `cpgedoc.cls` est en cours de test est contient encore beaucoup de bogues.

Les fichiers de style (extension `.sty`) du kit sont implicitement appelés par les deux fichiers de classe. Chacun s'occupe d'un aspect particulier dans le document. Certains sont utilisables avec la commande standard `\usepackage` sans avoir besoin d'utiliser les classes officielles du projet. Une section de ce document leur sera dédiée.

Le kit est pensé pour faciliter la réutilisation des contenus déjà disponibles. Moyennant une mise en conformité initiale, il permettra d'en produire des documents avec des apparences différentes sans avoir besoin de les modifier. Il sera possible de produire des versions épurées et économiques pour l'impression mais aussi des versions dynamiques avec hyperliens avec un fond sombre pour la consultation sur écrans tout en prenant en compte les différents formats d'écrans disponibles sur le marché. Le kit permettra de produire indifféremment des documents courts aux thèmes précis (un chapitre ou un devoir) mais aussi des documents longs (tous les devoirs d'une année ou des annales de sujets concours par exemple).

La première section de ce document est un tutoriel dont le but est de présenter les fonctionnalités du kit. Les autres sections sont des manuels de référence pour les divers aspects gérés par CPGEKIT.

2 Pour les pressés

Quoique non nécessaire, il est vivement recommandé d'utiliser un fichier maître et un ou plusieurs fichiers de contenu qui seront inclus par des instructions spéciales dans le fichier maître. Les fichiers de contenu seront alors réutilisables dans plusieurs documents au cours du temps sans avoir besoin d'y apporter des modifications.

Typiquement, un fichier maître pour produire un document de cours commencera avec une instruction de la forme

code 1
```
\documentclass[11pt, proof]{cpgedoc}
```

Le document produit contiendra alors les démonstrations des résultats qu'ils contient. Sans l'option `proof` celles-ci ne seront pas produites. Tandis qu'un fichier qui servira à produire un devoir avec indications et solutions commencera par une instruction de la forme

code 2
```
\documentclass[11pt, hint, solution]{cpgedev}
```

`hint` indique ici de produire les indications et `solution` les solutions éventuelles.

Dans les deux cas cette instruction sera suivie des deux lignes

code 3
```
\cpgegeometry{tablet}
\cpgetheme{newcg}
```

La deuxième ligne indique le thème à utiliser. En son absence le thème par défaut est utilisé. La première ligne indique la géométrie à utiliser. Elle est indispensable car elle est responsable d'autres aspects de la mise en forme. Ici `tablet` qui produit un document optimisé pour une consultation sur une tablette numérique aux proportions 4x3. Exemples d'autres valeurs possibles :

- `print` pour impression sur papier A4;
- `lsprint` pour impression sur papier A4 en mode paysage et en deux colonnes;
- `2print` pour impression sur papier A4 en deux colonnes et en mod portrait;
- `phone` pour consultation sur smartphone avec un écran en 16 x 9.

Viendra ensuite le tour du renseignement des métadonnées du document avec quelque chose comme

code 4
```
\Matiere{Mathématique}
\Document{Cours}
\Theme{Séries de Fourier}
\Auteur{Ahmed Boucheta} %% Si vous voulez passer à la postérité
\Classe{MP1}
\Periode{Janvier 2024}
```

ces informations seront utilisables dans la page de garde et les entêtes/pieds de pages. Le rédacteur pourra ensuite charger les packages dont il a besoin et qui ne sont pas déjà chargés par la classe. Immédiatement après `\begin{document}`, l'instruction

code 5
```
\titre
```

s'occupera de la formation de la page de garde pour les documents destinés à une consultation sur écran ou bien (par souci d'économie) d'un simple titre en haut de la feuille pour les documents à imprimer. Le contenu du titre est paramétrable à l'aide d'un mécanisme intuitif qui est exposé dans ce document. Des commandes sont aussi prévues pour modifier le contenu des entête/pieds de page.

Il est temps d'inclure les fichiers de contenus. Une commande s'occupera de manière transparente de l'application d'une stratégie spécifique d'inclusion des fichiers de contenus :

```
\cpgeinclude{SerFour}
```
code 6

Avec cette instruction le fichier `SerFour.tex` sera chargé. Avec l'option `proof` fournie à la classe, les éventuelles démonstrations des résultats énoncés dans ce fichier seront collectées et insérées à la fin du document. Chaque résultat disposant d'une démonstration disposera d'un hyperlien qui mènera vers sa preuve et inversement l'intitulé de la preuve est un hyperlien qui ramènera vers l'énoncé correspondant. Pour produire un document sans démonstrations,on peut soit le faire globalement pour tout le document en supprimant l'option `proof` de la classe soit individuellement pour le fichier inclus avec

```
\cpgeinclude[proof=false]{SerFour}
```
code 7

Pour le traitement d'un devoir, une command similaire se chargera de l'inclusion

```
\cpgeinclude{dm3}
```
code 8

Un mécanisme d'inclusion assez évolué est mis en oeuvre ici (voir détails plus loin). En gros :

- TEX commence par vérifier s'il y a des fichiers nommés `dm3-eno.tex` et `dm3-cor.tex`, si oui il va les inclure ;
- sinon il se contente d'inclure le fichier indiqué `dm3.tex`
- il est aussi possible de fournir le nom du fichier des solutions en option avec une syntaxe de la forme

  ```
  \cpgeinclude[solution file=dm3-corrige]{dm3}
  ```
  code 9

Dans tous les cas les fichiers inclus pourront contenir les énoncés et leurs éventuels corrigés dans des environnement LATEX spécifiques et TEX sera capable de créer les liens hypertextes individuels entre questions et solutions quelque soit leurs positions dans les fichiers inclus. De plus un mécanisme de références croisés est mis en place pour augmenter l'usabilité des documents destinés à une lecture sur écran.

Il est possible de produire un document sans solutions en enlevant l'option `solution` de la classe ou bien en forçant sa désactivation pour le fichier en cours avec l'instruction

```
\cpgeinclude[solution=false]{dm3}
```
code 10

Évidemment, rien de cela ne sera possible sans une mise en conformité des fichiers de contenus selon les directives du kit. Pour un devoir, deux stratégies sont pos-

sibles, les questions et leurs solutions sont enchevêtrée dans le même environnement LaTeX ou bien elles sont saisies dans des environnements séparés. Ces environnements peuvent en outre résider dans des fichiers différents. Chaque sujet devra être saisi dans un environnement LaTeX, `exercice` pour les exercices et `probleme` pour les problèmes. En cas de solutions séparées, elles doivent être saisi dans un environnement `corrige`. Chacun de ces environnements exige un identifiant unique en argument. Cet argument sert de lien entre un énoncé et son corrigé. Le schema suivant est adopté :

- dans la première stratégie, on saisit les questions et leurs solutions dans le même environnement. Chaque question sera suivie immédiatement par un environnement qui contient sa solution ;
- dans la seconde les questions sont mises dans un environnement `probleme`, par exemple, et son corrigé dans un environnement `corrige`. D'où la nécessité d'un identifiant commun pour les lier.
- dans les deux une question qui possède une solution devra être précédé d'une commande `\xques`, une qui n'en possède pas (qui a des sous-questions par exemple) sera précédée d'une commande `\zques`
- pour les solutions enchevêtrées, chaque `\xques` est suivi du texte de la solution dans un environnement `\begin{solution}...\end{solution}`
- pour les solutions séparées, il suffit de faire précéder une solution par une commande `\xsol`.

> . TeX maintient une liste en interne qui contient tous les numéros des `\xques` dans laquelle il va piocher à chaque `\xsol`

Le code suivant donne en exemple les deux façons de traiter un même contenu selon l'une ou l'autre des deux stratégies.

code 11

EXERCICE 1

Le titre de l'exercice

1. La première question. le + sert à initier un environnement de type enumerate. Cette question possède une réponse dans le fichier de corrigé.
2. Cette unité possède des sous-questions et donc ne pose pas de question elle-même
   - **2.a.** on augmente d'un niveau (2.a par exemple, le style de numérotation est facilement réglable)
   - **2.b.** toujours dans le deuxième niveau
3. On revient au premier niveau, avec une question avec réponse

CORRIGÉ DE L'EXERCICE 0

code 11

1. Réponse de la question 1, le texte de la solution sera enregistré dans un fichier séparé et sera ensuite inclu une fois l'énoncé terminé.

2.a. Réponse de la question 2.a, sans avoir besoin d'indiquer son numéro

2.b. Réponse de la question 2.b

3. Réponse de la question 3

```
\cpgesetuplists{
     enumi={n=|1|, d=\fbox, m=0pc},
     enumii={n=|Q|.|a|, d*=\bfseries #1., m*=2pc}
}
\begin{exercice}{IDEXE}(Le titre de l'exercice)
     <... presentation de l'exercice ...>
\xques+ La première question. le + sert à initier un environnement
    de type enumerate. Cette question possède une réponse dans le
    fichier de corrigé.
\begin{solution}
     Réponse de la question 1, le texte de la solution sera
    enregistré dans un fichier séparé et sera ensuite inclu une
    fois l'énoncé terminé.
\end{solution}
\zques Cette unité possède des sous-questions et donc ne pose pas
de question elle-même
     \xques+ on augmente d'un niveau (2.a par exemple, le style de
    numérotation est facilement réglable)
     \begin{solution}
          Réponse de la question 2.a, sans avoir besoin d'indiquer
    son numéro
     \end{solution}
     \xques toujours dans le deuxième niveau
     \begin{solution}
          Réponse de la question  2.b
     \end{solution}
\xques- On revient au premier niveau, avec une question avec
    réponse
\begin{solution}
     Réponse de la question 3
\end{solution}
\exit % On ferme tous les niveaux
\end{exercice}
```

code 12

```
\begin{exercice}{IDEXE}(le titre de l'exercice)
     <... presentation de l'exercice ...>
\xques+ La première question. le + sert à initier un environnement
    de type enumerate. Cette question possède une réponse dans le
    fichier de corrigé.
\zques Cette unité possède des sous-questions et donc ne pose pas
    de question elle-même
     \xques+ on augmente d'un niveau (2.a par exemple, le style de
    numérotation est facilement réglable)
     \xques toujours dans le deuxième niveau
\xques- On revient au premier niveau, avec une question avec
    réponse
\exit % On ferme tous les niveaux
\end{exercice}

\begin{corrige}{IDEXE}
\xsol Réponse de la question 1, le numéro est pioché dans une pile
\xsol Réponse de la question 2.a, sans avoir besoin d'indiquer son
    numéro
\xsol Réponse de la question  2.b
\xsol Réponse de la question 3
\end{corrige}
```

Dans ce deuxième exemple, il n'est pas nécessaire de mettre les environnements `exercice` et `corrige` dans le même fichier. C'est l'identifiant `IDEXE` qui fera le lien entre les deux. Il n'est même pas nécessaire d'inclure le fichier qui contient l'énoncé avant celui qui contient le corrigé. Le corrigé est systématiquement enregistré dans un fichier externe et c'est l'énoncé qui sera responsable de son inclusion ou non. Les deux méthodes se rejoignent dans le fait que même si les solutions sont enchevetrées, elles sont momentanémment écrite dans un fichier externe qui sera inclu suivant la même technique que dans l'autre stratégie. Ce qui fait qu'on peut mélanger sans souci entre des environnemnts qui utilisent chacun une stratégie différente.

La première méthode offre beaucoup de souplesse lorsqu'on veut réutiliser partiellement du contenu dans un nouveau document : les questions vont avec leurs solutions sans se soucier de leurs bonne numérotation. La seconde convient mieux lorsqu'on dispose déjà de fichiers de sujets et de corrigés séparés qu'on veut rendre conforme aux spécificités du kit.

3 Les devoirs

La creation de devoir se fait avec la classe `cpgedev.cls`. Celle-ci possède plusieurs options pour produire ou non les indications et les solutions dans le document final.

Il est préférable (mais pas obligatoire) d'utiliser un fichier maître qui inclura (au sens LaTeX) plusieurs fichiers esclaves pour former le document final. En outre un mécanisme est mis en place pour permettre de compiler individuellement les fichiers esclaves en allant chercher le préambule dans le fichier maître. Cela permet de raccourcir le temps de compilation et de rester concentré sur le contenu en cours de rédaction.

Un document "devoir" peut contenir un ou plusieurs problèmes et execice. Chaque `exercice` ou `probleme` peut être jumelé avec son `corrige` à l'aide d'un identifiant qui devrait être unique. L'énoncé et son corrigé peuvent résider dans des fichiers séparés qui seront chargé dans le fichier maître grâce à la commande `\iginclude`. Cette stratégie permettra de facilement recycler les documents déjà rédigés en des documents CPGEDEV.

En outre le kit offre une deuxième stratégie. Elle consiste en la saisie des questions immédiatement suivis de leurs solutions. Quoique plus contraignante pour un éventuel recyclage d'anciens fichiers, cette dernière présente un avantage de taille : pour une réutilisation ultérieur du contenu, ou de seulement une partie de celui-ci, un simple copier/coller suffira.

On peut mixer les deux méthodes : dans un même fichier maître on peut inclure des fichiers avec des énoncés et des corrigés séparés et des fichiers qui mélangent les deux. C'est l'ordre d'inclusion des énoncés qui décidera de l'ordre d'apparition des sujets dans le document produit.

I.Options de la classe

Les options suivantes sont disponibles.

`solution` Chaque sujet est immédiatement suivi de son corrigé s'il est disponible.

`solution*` Les corrigés sont collectés jusqu'à ce que le compilateur rencontre une commande `\corriges`. Il insère alors tous les corrigés retenus. Éventuellement on peut recommencer d'autre séquences de collecte/insertion.

`hint` produit les indications d'un sujet immédiatement après le sujet.

`hint*` chaque indication est produite au niveau de l'énoncé, là où elle est insérée, mais reste invisible jusqu'à ce que l'utilisateur clique sur un bouton.

> **N.B.** *Les calques PDF (OCG) font partie du standard PDF depuis sa version 1.5 mais ne sont gérés que par une poignée de lecteurs. ADOBE READER et FOXIT PDF READER en sont des exemples mais malheureusement aucun lecteur PDF pour appareils mobiles ne les gère à ce jour.*

`draft` on entre dans un mode brouillon où le style reste basique et sans liens hypertexte. Le contenu apparaît là où il est inséré. La compilation s'en trouve plus rapide.

`straight` Il produit les corrigés immédiatement là où il les rencontre sans création de fichiers intermédiaires. En outre c'est un vrai mode de production contrairement au mode précédent, avec liens hypertexte et possibilité d'appliquer un thème.

II.Instructions de mise en forme

Les commandes suivantes sont à utiliser dans le préambule.

\cpgegeometry commande essentielle qui appliquera une géométrie de page spécifique selon l'argument utilisé. Elle est ignorée quand l'option `draft` de la classe est utilisée. Les valeurs possibles de l'argument sont les suivantes :

`print` document A4 en mode portrait à une colonne ;

`2print` document A4 en mode portrait à deux colonne ;

`lsprint` document A4 en mode paysage à deux colonne ;

`tablet` document en mode portrait adapté à une tablette au format 4x3 (tous les iPad) ;

`lstablet` la même chose en mode paysage en deux colonnes ;

`alttablet` document en mode portrait adapté à une tablette au format 16x9 (Samsung et d'autres) ;

`alttablet` la même chose en mode paysage en deux colonnes ;

`phone` pour petit écran au format 16x9 ;

`altphone` pour petit écran au format 19x9.

`\cpgegeometry` gère en outre des paramètres de type clé/valeur qu'on peut mettre entre [ ] avant l'argument obligatoire. Toute option gérée par la commande `\geometry` du package `geometry` peut y être utilisé. Par exemple

```
\cpgegeometry[left=2.4cm, right=1.8cm]{print}
```
code 13

utilisera la géométrie `print` tout en modifiant les marges latérales de la page.

`\cpgegeometry` a en plus ses propres paramètres optionnels à côté de ceux qu'on passe à `\geometry`.

`dark` ou `dark=false` pour activer ou désactiver le mode sombre. Normalement, les versions pour petits écrans sont automatiquement produites avec un fond sombre et le texte en clair. On peut désactiver cette fonctionnalité avec `dark=false` ou bien l'activer pour les autres types de géométrie avec `dark=true` ou simplement `dark`.

`force compact` ou `force compact=false` option qui n'a pas d'effet apparent. Son rôle est d'activer le commutateur TEX `\ifcompact`. Ce commutateur est activé automatiquement pour les versions pour petits écrans et pour les versions en deux colonnes. Dans le document on peut utiliser une instruction conditionnelle de la forme

code 14
```
\ifcompact
     < texte si mode compact >
\else
     < texte si mode normal >
\fi
```

si l'option `compact` a été activée c'est le premier texte qui sera restitué, sinon c'est le second qui le sera. Bien utile quand on a une formule centrée qui déborde de la zone de texte dans les petites géométries.

`force geometry` option qui va forcer le mode indiqué par `\cpgegeometry` quand l'option `draft` de la classe est active.