Documentation pour la Base Partagée d'Exercice de Physique

(ou comment contribuer facilement à la BPEP)

29 juillet 2020

Introduction

Même si nous avons la particularité d'enseigner la physique au niveau CPGE, nous venons tous d'horizons différents avec des habitudes de composition en LaTeX qui sont certainement aussi nombreuses qu'il y a de compositeurs voulant contribuer à cette base. Ainsi, pour que tout le monde puisse s'y retrouver et profiter des contributions des autres, il est nécessaire de s'imposer un cadre qui permette à l'ensemble de fonctionner de concert. C'est la raison d'être de cette petite documentation qui vise, à partir d'exemples, à indiquer la structure à respecter ainsi que les bonnes habitudes (typographiques et de bonne composition pour que le fichier source soit facilement lisible au besoin) à prendre pour faciliter la vie de tous les futurs utilisateurs de la base.

Table des matières

1 Commandes à utiliser absolument **2**
- 1.1 Structure de base du fichier . . . 2
- 1.2 Mise en page des maths . . . 4
- 1.3 Produit vectoriel et produit scalaire . . . 5
- 1.4 Unités dans les applications numériques . . . 6
- 1.5 Composition des dérivées . . . 6
- 1.6 Composition des vecteurs . . . 7
- 1.7 Composition des moyennes . . . 7
- 1.8 Placement des figures . . . 7

2 Commandes à utiliser de préférence **9**
- 2.1 Mots de liaison . . . 9
- 2.2 Composition des indices . . . 10
- 2.3 Accolades en dessous ou au-dessus d'une expression . . . 10
- 2.4 Souligné et notation complexe . . . 11
- 2.5 Parenthèses, crochets et accolades . . . 11
- 2.6 Vecteurs de base . . . 11

3 Conseils divers **12**
- 3.1 Accents dans le texte . . . 12
- 3.2 Indentation . . . 12
- 3.3 Aération du fichier source . . . 12
- 3.4 Quelques environnements mathématiques utiles . . . 12

4 Aide pour adapter sa propre présentation **12**
- 4.1 Méthode générale pour adapter la forme . . . 12
- 4.2 Changement de la numérotation des questions . . . 13

1 Commandes à utiliser absolument

1.1 Structure de base du fichier

Pour structurer un exercice, et notamment pour pouvoir rendre disponible séparément (ou ensemble) la partie énoncé de la partie corrigé, il faut impérativement placer tout le contenu dans des commandes spécifiques :

`\titreExercice{Mon titre}` permet de donner un titre global à l'exercice ou au problème. L'usage en sera différent si vous construisez un DS/DM ou une feuille d'exercices de TD, mais ce sera réglé dans les fichiers spécifiques à chacun (voir section 4).

`\partie{Mon titre de partie}` et `\souspartie{Mon titre de sous-partie}` permettent de structurer l'exercice en parties et sous-parties (ce qui est principalement utile pour un problème de concours par exemple). Là encore, vous pourrez utiliser les fichiers spécifiques pour dire (par exemple) que `\partie` $\mapsto$ `\section` et `\sousPartie` $\mapsto$ `\subsection` ou au contraire que `\partie` $\mapsto$ `\subsection` et `\sousPartie` $\mapsto$ `\subsubsection` ou toute autre variation à votre convenance.

`\enonce` prend en argument le blabla contextualisant d'un énoncé qui n'est pas directement une question mais qui (contrairement à `\partie` ou `\sousPartie`) n'a pas forcément vocation à être présent dans un corrigé (sauf un corrigé où l'on remet l'énoncé bien sûr). Comme en général cette commande est amenée à contenir de grosses parties de texte, il est fortement conseiller de l'indenter de la manière suivante

```
\enonce{
  mon blabla
  sur
  plusieurs paragraphes
}
```

`\QR` est *la* commande sur laquelle tout le reste repose : il s'agit de donner en premier argument l'énoncé d'une question et en second argument le corrigé de la question (d'où le nom : QR $\mapsto$ Question-Réponse). Elle s'occupe de maintenir le compteur pour numéroter les questions en se basant sur `enumi` de sorte que l'on puisse customiser le rendu à volonté (par exemple pour ceux qui veulent précéder chaque question d'un « Q » ou bien mettre un cadre autour dans le corrigé) dans les fichiers décrits en section 4.

Là aussi, comme l'usage est que énoncé et corrigé occupent souvent un espace conséquent, il vaut mieux l'indenter comme suit :

```
\QR{
  Quelle est la question posée ?
}{
  Voici en tous cas la réponse proposée.
}
```

Il est possible de créer des sous-questions (un seul niveau pour le moment) en plaçant du contenu (enonce, QR, titre, ...) dans l'environnement `blocQR`.

Bon à savoir !

Comme expliqué dans le préambule, il est important de placer tout le contenu de l'exercice à l'intérieur des balises précédentes et ce, afin de pouvoir en adapter l'apparence selon vos souhaites (séparation du fond et de la forme). De plus, chacune de ces commandes peuvent accepter un paramètre optionnel, permettant d'indiquer au compilateur si ces derniers devront ou non être inclus dans le pdf.

Considérons comme exemple introductif le cas suivant :

```
\titreExercice{mon titre} % item 1

\enonce[bonus]{ % item 2
  blabla
  }

\addQ[bonus,bis]{ % item 3
  question
}{
  réponse
}

\enonce[original]{ %item 4
 bloblo
}
```

— Par défaut, seuls les items 1 et 4 seront affiché (`original` est en faite la version utilisée par défaut).

— En ajoutant la ligne `\renewcommand{\version}{bonus}` juste avant l'import de l'exercice, seuls les items 1, 2 et 3 seront affichés.

— De même, en utilisant `bis` à la place de `bonus`, seuls les items 1 et 3 seront affichés.

Ce mécanisme permet de créer plusieurs variantes pour un même exercice, et ce, en conservant un unique fichier ! N'hésitez donc pas à l'utiliser pour adapter un exercice existant sans impacter la version originale. Vous pouvez aussi échanger avec l'auteur afin d'effectuer des modifications conséquentes sans utiliser ce mécanisme.

Rassemblons tout ce qui à été abordé dans un exemple minimal :

```
% Le titre est obligatoire
\titreExercice{Le titre de mon exercice}

% partie et sousPartie, en revanche, non, cela dépend des cas
\partie{Première partie}

\enonce{
  Le début des explications de l'énoncé
}

\QR{
  La première question
}{
  La première réponse
}

\QR{
  La deuxième question
}{
  La deuxième réponse
}

% Début d'une seconde partie
\partie{Deuxième partie}

% On peut réinitialiser les numéros des questions si on veut
```

```
\resetQ

\QR{
  La première question de la seconde partie
 % (renumérotée 1 grâce à la commande \resetQ)
}{
  La première réponse de la seconde partie
}

\begin{blocQR}
  \QR{
    Voici la question numéro 2.a
  }{
    et sa réponse
  }

  \QR{
    Voici la question numéro 2.b
  }{
    et sa réponse
  }
\end{blocQR}
```

1.2 Mise en page des maths

Lorsque des maths se glissent dans le texte, il suffit de les encadrer par de simples dollars (`$`) pour le faire apparaître correctement. Attention à toujours le faire dès que vous manipulez une grandeur mathématique car le rendu n'est pas le même selon que l'on est ou non en mode math (qui n'est *pas* équivalent à de l'italique, cela dépend de la police que l'on choisit dans son document). En particulier, parler de la fonction f (`$f$`) n'est pas la même chose que parler de la fonction f (`f`), seul le premier cas est typographiquement correct.

Pour les mathématiques centrées (ce qui arrive très souvent), nous vous demandons d'utiliser systématiquement la commande `\eq` qui est par défaut un raccourci vers l'environnement `{equation*}` (qui est équivalent aux doubles dollars ou au couple `\[...\]`). L'avantage d'une commande unique est, là encore, de pouvoir customiser, par exemple pour systématiquement réduire les espaces verticaux laissés par LaTeX avant et après les équations centrées. En outre, cela permet aussi de passer d'un environnement à l'autre (grâce à l'argument optionnel décrit juste après) si on décide de changer en cours de route. Un petit exemple :

```
\eq{
  1 = \cos^2\theta + \sin^2\theta
    %= \frac{1 + \cos(2\theta)}{2} + \sin^2\theta
    = \frac{1 + \cos(2\theta)}{2} + \frac{1 - \cos(2\theta)}{2}
    = \frac{1}{2} + \frac{1}{2}
    = 1
}
```

ce qui donne pour résultat

$$1 = \cos^2\theta + \sin^2\theta = \frac{1 + \cos(2\theta)}{2} + \frac{1 - \cos(2\theta)}{2} = \frac{1}{2} + \frac{1}{2} = 1$$

Notez que l'on a écrit le code source pour améliorer la relecture et permettre (comme c'est le cas ici) de facilement commenter un intermédiaire finalement jugé inutile (mais tout de même le garder sous le coude au cas où). Pour des calculs compliqués, il est utile de composer comme si on écrivait ces équations sur feuille à la main, même si LaTeX n'a que faire des retours à la ligne.

Bon à savoir !

La commande `\eq` admet un argument optionnel qui permet l'utilisation d'autres environnements mathématiques de LaTeX comme `{align*}` ou `{gather*}` par exemple. Dans le code précédent, on pourrait choisir de composer comme dans une copie d'élève avec des signes égal alignés les uns en dessous des autres. `{align*}` est alors tout indiqué et le code précédent s'adapte facilement en ajoutant juste des esperluettes (`&`) et des fins de ligne (`\\`). Ainsi,

```
\eq[align*]{
  1 &= \cos^2\theta + \sin^2\theta  \\
    %= \frac{1 + \cos(2\theta)}{2} + \sin^2\theta
    &= \frac{1 + \cos(2\theta)}{2} + \frac{1 - \cos(2\theta)}{2} \\
    &= \frac{1}{2} + \frac{1}{2} \\
  1 &= 1
}
```

donne

$$\begin{aligned} 1 &= \cos^2\theta + \sin^2\theta \\ &= \frac{1+\cos(2\theta)}{2} + \frac{1-\cos(2\theta)}{2} \\ &= \frac{1}{2} + \frac{1}{2} \\ 1 &= 1 \end{aligned}$$

En revanche, si on veut superposer deux équations sans lien particulier entre elles, l'une au-dessus de l'autre, on peut être tenté d'écrire

```
\eq[align*]{
  A+B+C+\cdots+C+B+A = D \\
  E = F+G
}
```

ce qui donne pour résultat

$$\begin{aligned} A+B+C+\cdots+C+B+A &= D \\ E &= F+G \end{aligned}$$

alors qu'on aimerait que les deux équations soient centrées sur la ligne, indépendamment l'une de l'autre, ce que fait l'environnement `{gather*}`. On peut alors écrire

```
\eq[gather]{
  A+B+C+\cdots+C+B+A = D \\
  E = F
}
```

ce qui a bien l'effet voulu

$$\begin{gathered} A+B+C+\cdots+C+B+A = D \\ E = F \end{gathered}$$

À retenir : la plupart du temps, c'est directement `\eq{}` qu'il faut utiliser, mais si jamais vous avez des choses plus compliquées à faire, c'est possible grâce à l'argument optionnel. Allez voir à la section 3.4 pour plus de détails sur les divers environnements mathématiques que LaTeX nous offre et comment les utiliser avec `\eq{}`.

1.3 Produit vectoriel et produit scalaire

Pour que chacun puisse choisir d'utiliser $\times$ (`\times`) ou $\wedge$ (`\wedge`) pour noter son produit vectoriel, il est convenu que l'on utilise à chaque fois la commande `\vectoriel` que chacun pourra redéfinir à sa guise par la suite (en outre, c'est plus facile de relire le code correspondant). On écrira donc par exemple `$\vec{v}\vectoriel\vec{B}$`.

De la même manière, le produit scalaire est usuellement noté avec un $\cdot$ (`\cdot`) mais on peut aussi préférer un simple point. À nouveau, pour que chacun puisse choisir selon sa préférence, on notera systématiquement `\scalaire` comme par exemple dans `$\vec{F}\scalaire\vec{v}$`.

1.4 Unités dans les applications numériques

Les unités doivent être typographiées en caractères droits (contrairement au reste des mathématiques) mais doivent être en mode math car il y a souvent des exposants à y rattacher. En bref, cela peut vite devenir le cauchemar si on essaie de bidouiller des entrées/sorties du mode maths à la main. Il s'agit donc d'utiliser un outil hautement configurable qui permettra à chacun de gérer comme il l'entend le rendu des unités. Cet outil, c'est le package `{siunitx}` qui le fournit. Ce package fournit trois commandes qui permettent de gérer correctement les unités, à savoir les commandes `\num{}`, `\si{}` et `\SI{}{}`. Voici les cas d'utilisation :

`\num{valeur}` permet de s'assurer que l'espacement après la virgule de séparateur décimal est le bon. Comparez par exemple $3,0$ (`$3,0$`) à 3,0 (`$\num{3,0}$`). Dans le premier exemple, le 0 est bien trop éloigné de la virgule[1]. Il permet en outre de ne pas s'enquiquinner avec les puissances de 10 en proposant de les composer lui-même. Par exemple, la fameuse constante de structure fine s'écrit $7{,}297 \times 10^{-3}$ et se compose simplement[2] avec `$\num{7.297e-3}$`.

Attention, en général, on n'utilise `\num` *que* pour les applications numériques de grandeurs sans dimensions, car s'il y a une unité, ce sera `\SI{}{}` qu'il faudra utiliser

`\si{unité}` permet quant à elle de composer les unités directement dans le texte, par exemple quand on dit qu'une vitesse s'exprime en $\mathrm{m\,s^{-1}}$, on écrit simplement

```
on dit qu'une vitesse s'exprime en \si{m.s^{-1}}
```

Cela permet aussi de préciser l'unité d'une mesure entre parenthèse sans qu'un espace en trop ne soit ajouté. Par exemple, pour parler de l'intensité i ($\mathrm{C\,s^{-1}}$), on écrira `$i$ (\si{\coulomb\per\second})`

`\SI{valeur}{unité}` est *la* forme qui devrait être employée le plus souvent. Elle s'occupe tout à la fois de gérer la valeur numérique (comme `\num`) ainsi que la composition des unités (comme `\si`) tout en respectant l'espace obligatoire entre valeur et unité. Bref, que du bonheur. Par exemple, si on compose

```
\eq{
  v = \SI{3.0e-1}{m.s^{-1}}
    = \SI{3.0e2}{\milli\meter\per\second}
}
```

on obtient

$$v = 3{,}0 \times 10^{-1}\,\mathrm{m\,s^{-1}} = 3{,}0 \times 10^{2}\,\mathrm{mm\,s^{-1}}$$

Notez que l'on peut, pour les unités, soit composer en tapant directement les unités avec les puissances adéquates (en n'oubliant pas de mettre des points entre deux sous-unités), soit taper des formes explicites (mais anglaises) que `\SI` se débrouille pour traduire correctement.

En résumé, utilisez toujours la forme majuscule pour vos applications numériques dimensionnées et sachez que tout est paramétrable : le séparateur décimal, le signe devant la puissance de 10 ou encore ce qu'il faut mettre entre deux sous-unités (espace, point ou point médian...). Tout sera expliqué en section 4.

1. ce qui est le comportement voulu quand on écrit le point (x, y) (`$(x,y)$`) mais pas quand on veut une valeur numérique.
2. Si le signe de multiplication ne vous convient pas, vous pouvez le changer dans les options du package, voir la section 4

1.5 Composition des dérivées

Il y a des règles typographiques assez strictes sur la composition des dérivées. Plutôt que d'avoir à toutes les retenir, mieux vaut utiliser un package (en l'occurrence `{esdiff}`, maintenu par un membre de l'UPS) qui fait tout cela pour nous. La commande à utiliser pour les dérivées droites est `\diff{}{}`, celle pour les dérivées partielle est `\diffp{}{}` (≪ p ≫ pour *p*artielle). Les deux commandes acceptent un argument optionnel pour l'ordre de la dérivée. Les versions étoilées permettent de préciser un point d'application (pour une dérivée droite) ou quelle variable est gardée constante (pour les dérivées partielles, particulièrement en thermo). Voici quelques exemples d'utilisation

`\diff{x}{t}`	⟶	$\dfrac{\mathrm{d}x}{\mathrm{d}t}$
`\diff[2]{x}{t}`	⟶	$\dfrac{\mathrm{d}^2x}{\mathrm{d}t^2}$
`\diff*{x}{t}{t_0}`	⟶	$\left(\dfrac{\mathrm{d}x}{\mathrm{d}t}\right)_{t_0}$
`\diffp{x}{t}`	⟶	$\dfrac{\partial x}{\partial t}$
`\diffp[2]{x}{t}`	⟶	$\dfrac{\partial^2 x}{\partial t^2}$
`\diffp*{H}{T}{P}`	⟶	$\left(\dfrac{\partial H}{\partial T}\right)_P$
`\diffp{x}{{t}{\tau^2}}`	⟶	$\dfrac{\partial^3 x}{\partial t \partial \tau^2}$

Comme on le voit dans le dernier exemple, les dérivées croisées sont aussi possibles à condition d'imbriquer les accolades.

1.6 Composition des vecteurs

Pour que chacun puisse choisir dans son fichier de style propre de la manière de composer les vecteurs (avec une flèche ou en gras par exemple), les vecteurs doivent être composés à l'aide des commandes `\vec` ou `\vect` [3]. Les deux sont des redirections vers la commande idoine [4] du package `{esvect}` [5] dont vous pourrez changer l'option d'appel dans vos fichiers de style si jamais la flèche par défaut ne vous convient pas. On composera donc par exemple `$\vec{v}\vectoriel\vec{B}$` pour obtenir $\vec{v} \wedge \vec{B}$ ou encore `$\vect{F}\scalaire\vect{v}$` pour obtenir $\vec{F} \cdot \vec{v}$.

Pour composer les vecteurs avec un indice, il existe une version étoilée qui prend deux arguments : la quantité à ≪ vectoriser ≫ et son indice. On composera par exemple `$\vec*{e}{r}$` pour $\vec{*}er$ ou `$\vect*{L}{\Delta}$` pour $\vec{*}L\Delta$.

1.7 Composition des moyennes

Une commande `\moyenne` permet de composer automatiquement `$\moyenne{u(t)}$` ce qui donne $\langle u(t) \rangle$. Un usage systématique dans tous les exercices permettra à chacun de la noter comme il veut si besoin.

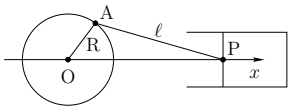
1.8 Placement des figures

Nous savons tous que, dans l'absolu, il faudrait utiliser l'environnement `{figure}` pour que LaTeX s'occupe tout seul du placement des figures... Mais tout ancien thésard sait par expérience que l'algorithme de placement des figures de LaTeX est tout autant efficace qu'il est difficile à cerner pour un néophyte, donc en pratique on opte

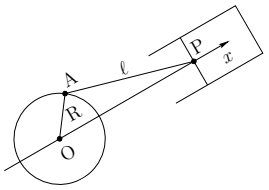
3. La commande `\vect` est un simple synonyme car l'abréviation peut paraître plus naturelle que `\vec`.
4. À ne *pas* utiliser directement, c'est pourquoi on ne la nomme pas ici !
5. Lui aussi maintenu par le même collègue de l'UPS.

souvent pour un placement « en dur » des figures, généralement sous forme centrée. À cet effet, les commandes `\fig` et `\figRaw` ont été crées pour harmoniser les choses :

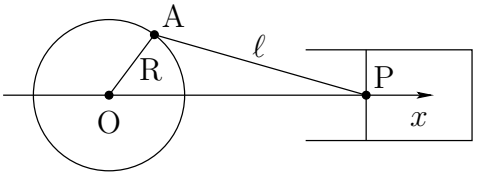
`\fig` prend deux arguments, le premier est un nombre (généralement entre 0 et 1) correspondant à la fraction de largeur du texte sur laquelle doit s'étaler la figure et le second est le nom du fichier à inclure (forcément placée dans le dossier `images/`). Ainsi, pour inclure le fichier `mon_image.jpg` bien placé dans le dossier `images/` de sorte qu'il occupe 20% de la largeur du texte courant, on écrira `\fig{0.7}{mon_image.pdf}`.



En argument optionnel, on peut rajouter tout argument supplémentaire qui serait compris par la commande `\includegraphics` qui est appelée en sous-main. Par exemple, pour tourner la figure précédente de 30°, on utilisera la commande `\fig[angle=30]{0.2}{mon_image.pdf}`.



`\figRaw` est la version à un seul argument de la commande précédente : l'image est directement donnée à `\includegraphics` sans imposer de largeur. C'est la commande à utiliser pour tous ceux qui se débrouillent pour que leurs images aient déjà la bonne dimension. Voici l'effet de `\figRaw{mon_image.pdf}`



Comme la précédente, on peut donner en argument optionnel tout argument qu'il faudrait donner à `\includegraphics` pour gérer la figure comme on l'entend. En particulier la commande `\fig{0.2}{mon_image.pdf}` est strictement équivalente à `\figRaw[width=0.2\textwidth]{mon_image.pdf}`

Les deux commandes précédentes existent aussi en version « Cap » appelée `\figCap` et `\figCapRaw` qui prennent chacune un argument supplémentaire par rapport à leurs versions normales qui est la légende (automatiquement numérotée) qu'on voudrait leur donner. Voici les appels respectifs à `\figCap{0.2}{mon_image.pdf}{Une belle légende}` et `\figCapRaw{mon_image.pdf}{Une belle légende\label{monLabel}}`

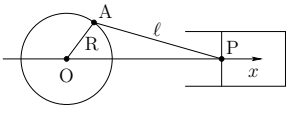


FIGURE 1 – Une belle légende

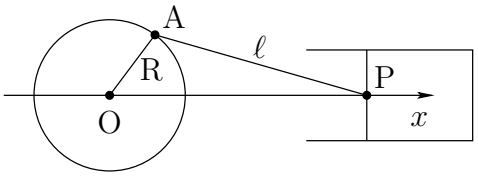
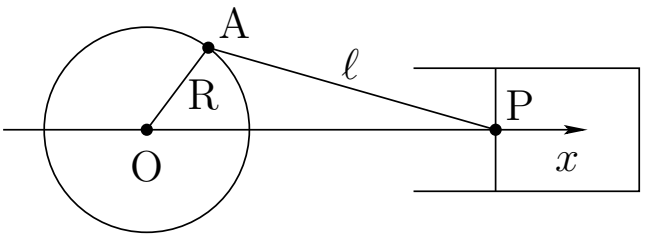


FIGURE 2 – Une belle légende

On remarque que les figures ont leur numéro automatiquement pris en charge par LaTeX et si on rajoute la commande `\label{}` *à l'intérieur* de la légende (comme fait ici pour la seconde figure), on peut y faire référence dans le texte à l'aide de la commande `\ref{}` (ici `\ref{monLabel}`) pour que LaTeX fasse directement référence à la figure 2 (composé `figure~\ref{monLabel}` dans le fichier source) sans que nul n'ait besoin de savoir à quel numéro on en est (utile si on rajoute soudainement une partie avec une figure numérotée supplémentaire, il n'y a rien à penser, LaTeX s'occupe de tout).

Dernière possibilité pour inclure une figure, on peut la couler autour d'un paragraphe de texte à l'aide de l'environnement `{wrapfigure}`. Une fois que la figure a occupé la place qui lui est due, le texte revient bien gentiment se fondre en dessous sans que vous ayez à intervenir de quelque façon que ce soit.



Le résultat que vous observez peut être produit soit par le raccourci `\figWrapped{0.5}{mon_image.pdf}` placé juste avant votre paragraphe de texte où le `0.5` donne la fraction de la largeur du texte que doit occuper la figure sur la droite (sachant que la vraie figure occupe 90% de cet espace pour assurer un peu de blanc sur le tour), soit directement avec le code

```
\begin{wrapfigure}{R}{0.5\textwidth}
\fig{0.9}{mon_image.pdf}
\end{wrapfigure}
```

2 Commandes à utiliser de préférence

Le fichier de raccourcis définit un certain nombre de commandes utiles pour faciliter la composition mais dont l'utilisation n'est que conseillée et pas obligatoire. Il s'agit principalement de se simplifier la vie avec les choses qui reviennent souvent et qui peuvent être pénible à taper en LaTeX.

2.1 Mots de liaison

Il arrive souvent que, pour gagner un peu de place sur une feuille et ne pas se sentir coupable de faire imprimer trop de feuilles un peu trop blanches aux élèves, on se permette de lier deux équations sur une même ligne par un mot de liaison (du type « donc », « soit », « et », etc.). Pour ce faire, le mieux est d'utiliser une construction du type

```
\eq{
  a = b
  \qquad\text{donc}\qquad
  a = c
}
```

qui donne

$$a = b \qquad \text{donc} \qquad a = c$$

On vous fournit les raccourcis `\qqdonc`, `\qqsoit`, `\qqcar`, `\qqet`, `\qqou` et `\qqavec` qui sont les mots de liaisons les plus courants. On dispose en outre d'une command `\qqMath{}` qui permet d'en rajouter facilement avec les mêmes règles d'espacement, par exemple `\qqMath{c'est-à-dire}`.

Pour ne mettre que des cadratins (`\quad`) plutôt que des doubles cadratins (`\qquad`), il y a aussi `\qdonc`, `\qsoit`, `\qcar`, `\qet`, `\qou` et `\qavec` ainsi que `\qMath{}` pour faire ceux qui manqueraient. L'usage typique est

pour séparer logiquement deux propositions dont l'une est aussi constituée de deux parties comme dans l'exemple suivant

```
\eq{
  a = b \qet b = c
  \qqdonc
  a = c
}
```

$$a = b \quad \text{et} \quad b = c \qquad \text{donc} \qquad a = c$$

L'espace laissé est bien plus court autour du « et » qu'autour du « donc ».

2.2 Composition des indices

Concernant les variables indicée, il existe une règle typographique simple :

— si l'indice correspond lui-même à une variable (par exemple quand on parle de x_i la i^{e} valeur d'un certaine suite), alors l'indice est en mode math et on compose simplement `$x_i$` ;

— en revanche, si l'indice est l'abréviation d'un mot (par exemple quand on parle de x_{i} la position *i*nitiale d'une particule), alors l'indice doit être en mode texte et on composera `$x\ind{i}$` qui est un raccourci pour `$x_{\text{i}}$`.

En physique, il s'agit très souvent du second cas. En particulier les énergies cinétique E_{c}, potentielle E_{p} et mécanique E_{m} doivent être composées respectivement par `$E\ind{c}$`, `$E\ind{p}$` et `$E\ind{m}$` (vous êtes tout de même libre du symbole pour l'énergie, par exemple $\mathcal{E}_{\text{c}}$ avec `$\mathcal{E}\ind{c}$`). Une puissance efficace $\mathcal{P}_{\text{eff}}$ se compose `$\mathcal{P}\ind{eff}$`. Notez que le passage en mode texte permet d'utiliser des accents dans l'indice comme pour $x_{\text{éq}}$ (composé `$x\ind{éq}$`).

2.3 Accolades en dessous ou au-dessus d'une expression

Pour indiquer une notation au-dessus ou en dessous de certains termes, LaTeX fournit les commande `\overbrace` et `\underbrace` qui s'utilisent comme suit

```
\eq{
  E\ind{tot}
    = \overbrace{E\ind{c}^* + E\ind{p,int}}^{U}
    + \underbrace{\frac{1}{2} M{v_G}^2 + E\ind{p,ext}}_{E\ind{m,macro}}
}
```

$$E_{\text{tot}} = \overbrace{E_{\text{c}}^* + E_{\text{p,int}}}^{U} + \underbrace{\frac{1}{2} M {v_G}^2 + E_{\text{p,ext}}}_{E_{\text{m,macro}}}$$

Il faut se souvenir que pour mettre quelque chose au-dessus de `\overbrace`, il faut utiliser la mise en exposant via `^` alors que pour mettre quelque chose en dessous de `\underbrace`, il faut utiliser la mise en indice via `_`.

Comme cela peut être un peu pénible, on a défini les raccourcis `\ob{}{}` et `\ub{}{}` qui prennent tous les deux deux arguments : ce que l'on veut englober dans l'accolade et ce qu'il faut mettre au-dessus ou en dessous selon le cas. L'exemple précédent se réécrit donc

```
\eq{
  E\ind{tot}
    = \ob{E\ind{c}^* + E\ind{p,int}}{U}
    + \ub{\frac{1}{2} M{v_G}^2 + E\ind{p,ext}}{E\ind{m,macro}}
}
```

```
\eq{
  \ul{u}_2 = \frac{\ul{Z}_2}{\ul{Z}_1+\ul{Z}_2}\,\ul{u}
}
```

$$\underline{u}_2 = \frac{\underline{Z}_2}{\underline{Z}_1+\underline{Z}_2}\,\underline{u}$$

Remarquez qu'on préfèrera composer `\ul{Z}_2` (soit $\underline{Z}_2$) à `\ul{Z_2}` (soit $\underline{Z_2}$), c'est-à-dire qu'on garde les indices *à l'extérieur* du souligné.

Pour ceux qui préfèrent, on peut utiliser le raccourci `\cplx` (pour « ComPLeXe ») plutôt que `\ul`.

2.5 Parenthèses, crochets et accolades

En général, il faut adapter la taille des parenthèses, crochets ou accolades à ce qu'ils contiennent. LaTeX fournit un mécanisme simple (adaptable à n'importe quel couple de symboles) sous la forme `\left( a+b \right)`. Si on veut un symbole que d'un seul côté, on peut utiliser le point, comme pour la définition d'un système sous la forme `\left{ monSystème \left.` où il n'y aura rien d'affiché du côté droit du système.

Cela peut rapidement devenir assez pénible à maintenir à la main, surtout si l'on décide de rajouter un niveau de parenthèse et donc changer les parenthèse en crochets puis les crochets en accolades, LaTeX se fichant pas mal qu'il y ait correspondance entre la version qui ouvre et la version qui ferme...

Bref, on a aussi défini des raccourcis pour se faciliter la vie et passer rapidement d'un type à un autre sans avoir à tout retaper : `\pa` pour des *pa*renthèses, `\pac` pour des *c*rochets et `\paa` pour des *a*ccolades. On peut alors composer

```
\eq{
  u(t) = \exp\fracp{-t}{\tau}
         \paa{ A\cos\pac{\frac{\omega_1+\omega_2}{2}\pa{t-t_0}+\varphi}
              +B\sin\pac{\frac{\omega_1-\omega_2}{2}\pa{t-t_0}+\psi}
         }
}
```

$$u(t) = \exp\left(\frac{-t}{\tau}\right)\left\{A\cos\left[\frac{\omega_1+\omega_2}{2}\,(t-t_0)+\varphi\right]+B\sin\left[\frac{\omega_1-\omega_2}{2}\,(t-t_0)+\psi\right]\right\}$$

À noter que l'on a aussi défini `\fracp` pour mettre directement des parenthèse autour d'une fraction, cas pratique qui se retrouve très fréquemment en physique (par exemple en thermo mais pas seulement).

2.6 Vecteurs de base

En mécanique, on manipule très souvent les mêmes vecteurs de base. Ils ont donc des raccourcis associés.

```
\eq{
  \ex\vectoriel\ey=\ez
  \qqet
  \er\vectoriel\et=\ep
}
```

$$\vec{*}ex \wedge \vec{*}ey = \vec{*}ez \qquad \text{et} \qquad \vec{*}er \wedge \vec{*}e\theta = \vec{*}e\phi$$

3 Conseils divers

3.1 Accents dans le texte

La BPEP est conçue pour utiliser l'encodage UTF8 par défaut. Si vous pouvez régler votre éditeur pour l'utiliser, ce sera plus facile de relire les sources si les accents ne sont pas composés « à la TEX ».

3.2 Indentation

L'« exercice témoin » présente quelques « bonnes » pratiques en terme d'indentation et de mise en forme du fichier source : pensez à vous en inspirer.

3.3 Aération du fichier source

N'hésitez pas à laisser des lignes blanches entre les différentes `\QR` dans votre fichier source pour que l'on n'ait pas une impression de bloc monalythique.

De la même manière, sautez des lignes blanches lorsque vous commencez une nouvelle idée ou un nouvel argument par que l'on puisse facilement s'y retrouver.

N'hésitez pas non plus à laisser des lignes de signes pourcentage pour que l'on voit la structuration. Dans ce fichier par exemple, on peut lire

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\subsection{Aération du fichier source}

N'hésitez pas à laisser des lignes blanches entre les différentes
\verb|\QR| dans votre fichier source pour que l'on n'ait pas une
impression de bloc monalythique.

De la même manière, sautez des lignes blanches lorsque vous commencez
une nouvelle idée ou un nouvel argument par que l'on puisse facilement
s'y retrouver.

N'hésitez pas non plus à laisser des lignes de signes pourcentage pour
que l'on voit la structuration.
Dans ce fichier par exemple, on peut lire

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\subsection{Quelques environnements mathématiques utiles}
\label{maths}
```

3.4 Quelques environnements mathématiques utiles

Section pour Bruno

4 Aide pour adapter sa propre présentation

4.1 Méthode générale pour adapter la forme

L'idée principale de la syntaxe utilisée dans la BPEP est d'obliger l'utilisateur à utiliser un certain nombre de macro afin d'englober l'ensemble du contenu.

Une implémentation de base de ces dernières est fournie dans le fichier `raccourcis_communs.sty` qui doit être importé afin de pouvoir compiler un ou plusieurs exercices.

Cependant, ces macros peuvent très bien être modifiées par chacun à l'aide des commandes LaTeX suivantes :

```
\renewcommand{\nomDeLaMacro}[nombre de paramètres]{
  % code de la nouvelle version de la macro
  ...
}

\renewenvironment{\nomDeLEnvironement}{
  % code après \begin{...}
}{
  % code placé juste avant \end{...}
}
```

Ces dernières pourront être placées dans vos templates personnels (fichiers LaTeX dans lesquels les exercices vont être inclus). Ainsi, il est possible de grandement modifier la forme du document. Quelques modifications courantes sont détaillées dans la suite du document :

4.2 Changement de la numérotation des questions

L'environnement `enumerate` étant utilisé en interne pour gérer la numérotation des questions, vous pouvez donc choisir d'afficher ces derniers comme vous le souhaitez :

```
  %  Change les numéros des sous-questions en lettres majuscules A,B,C, ...
\renewcommand{\labelenumii}{\Alph{enumii}}

  %  Affiche les numéros sous la forme Q IV)
\renewcommand{\labelenumi}{Q \Roman{enumi})}
```

De même, redéfinir l'environnement `blocQR` dans vos templates personnels permet de ne pas utiliser de questions imbriquées sans changer le fichier `sujet.tex` :

```
\renewenvironment{\blocQR}{}{}
```