L'extension numprint

Harald Harders
h.harders@tu-bs.de

Version v1.37, 2007/01/08; printed 30 mars 2007

Résumé

Cette extension compose les nombres avec un séparateur tous les trois chiffres et convertit les nombres donnés comme `12345.6e789` en $12\,345{,}6 \cdot 10^{789}$. Les nombres sont composés dans le mode courant (texte ou mathématique) afin que la police correcte soit employée.

L'utilisateur peut changer de nombreuses choses y compris le signe décimal, le séparateur de milliers et le signe du produit pour obtenir p. ex. 12,345.6e789.

Si l'on donne un argument optionnel, il est imprimé en forme droite (*upright*) comme un symbole d'unité. Les nombres peuvent être arrondis à un nombre donné de décimale.

L'extension assure un changement automatique, dépendant de la langue, du format de nombre.

Dans les tableaux réalisés avec les environnements `tabular(*)`, `array`, `tabularx` et `longtable` on peut obtenir un alignement (semblable à ce que l'on obtient avec les extensions `dcolumn` et `rccol`) qui utilise toutes les capacités de numprint. On peut placer du texte supplémentaire avant et après les nombres formatés.

Si l'on veut imprimer des nombres avec des bases différentes (octal, hexadécimal, etc.), on utilisera l'extension `nbaseprt` qui accompagne l'extension `numprint`.

Traduction française par le TEXnicien de surface,
le.texnicien.de.surface@wanadoo.fr,
version 1.0 de la traduction, 2007-03-25.

Table des matières

Copyright

Remarques

! *Les commandes* `\fourdigitsep`*,* `\fourdigitnosep`*,* `\addmissingzero`*,* `\noaddmissingzero`*,* `\digits`*,* `\nodigits`*,* `\exponentsdigits`*, et* `\noexponentdigits` *ont été renommées en* `\npfourdigitsep`*,* `\npfourdigitnosep`*,* `\npaddmissingzero`*,* `\npnoaddmissingzero`*,* `\npdigits`*,*

! `\npnodigits`*,* `\npexponentsdigits` *et* `\npnoexponentdigits` *respectivement.*

De la version 1.00 à la 1.10, les types de colonnes ont changé, cf. la section 6. Si vous désirez conserver les anciens types de colonnes, utilisez cette extension avec l'option `oldcolumntypes`*.*

1 Charger l'extension

Pour utiliser cette extension, on place

```
\usepackage{numprint}
```

dans le préambule du document. Aucune option n'est indispensable mais certaines sont disponibles. On les citera quand on en décrira l'utilisation et aussi dans la section B.1.

2 Composer des nombres en mode texte et mode mathématique

`\numprint` Cette extension fournit la commande `\numprint[`⟨*unité*⟩`]{`⟨*nombre*⟩`}` qui compose le ⟨*nombre*⟩ donné comme argument obligatoire. Le nombre est imprimé dans le mode courant (mathématique ou texte) afin qu'une différence qu'on aurait voulue, avec p. ex. l'extension `eco`, entre les nombres du texte courant et ceux des formules mathématiques soit conservée.

Les nombres peuvent contenir ces caractères : « `+-\pm0123456789.,eEdD` ». Les espaces, « `\,` » ainsi que « `~` » contenu dans l'argument sont ignorés. On peut utiliser soit « `,` » soit « `.` » comme signe décimal. Par défaut, on n'autorise pas de séparateurs de milliers dans l'argument [1]. « `E` », « `e` », « `D` », ou « `d` » [2] sont convertis en format exponentiel (p. ex. $x \cdot 10^y$ ou $x \times 10^y$ suivant le choix de format qui sera décrit plus loin). « `\pm` » et « `+-` » produisent un ±.

Par exemple, en écrivant

```
\numprint{-123456}; \numprint{\pm 123456}; \numprint{+-3,1415927e-3.1}
```

1. La Section 8.2 explique comment changer ce comportement.
2. Cela est utile pour des nombres produits par FORTRAN.

on obtient « −123 456 ; ±123 456 ; ±3,141 592 7 · $10^{-3,1}$ ». Notez que l'on peut mélanger « . » et « , » dans un même nombre et qu'ils sont changés en le séparateur décimal choisi pour la sortie.

Le nombre est composé dans le mode actif (mode texte ou mathématique). C'est important si les chiffres sont différents suivant qu'ils se trouvent dans le texte ou dans des formules mathématiques comme dans ce document qui utilise des chiffres « à l'ancienne » (*old style*) dans le texte et des chiffres courants pour les mathématiques [3]. Voyez la différence entre « 123 456,134 · 10^{123} » et « 123 456,134 · 10^{123} », produits par

```
\numprint{123456.134e123} et $\numprint{123456.134e123}$
```

Si l'on ne donne pas de nombre avant le caractère de l'exponentielle – « `e` », « `E` », « `d` » ou « `D` » – un format purement exponentiel est créé. En saisissant par exemple

```
\numprint{e4.3242}
```

on obtient « $10^{4,3242}$ ». Cela fonctionne également si l'on donne un signe, p. ex.

```
\numprint{-e4.3242}, \numprint{+-e4.3242}
```

produit « $-10^{4,3242}$, $\pm 10^{4,3242}$ ».

Comme `\numprint` développe son argument avant de le mettre en page, on peut également utiliser des commandes dans l'argument, p. ex.

```
\def\numberbefore{1234}%
\def\totalnumber{\numberbefore.5678}%
\numprint{\totalnumber}
```

produit « 1 234,567 8 ».

Si l'argument optionnel est donné, il est imprimé comme une ⟨*unité*⟩ en mode mathématique avec une police droite (`\mathrm`), p. ex.

```
\numprint[N/mm^2]{-123456}
```

produit « −123 456 $\mathrm{N/mm^2}$ ».

Par défaut, l'espace entre le nombre et l'unité est `\,`. Le symbole de degré fait exception car on l'imprime sans le séparer du nombre, p. ex. 360° à comparer avec 273,15 °C – seul le symbole du degré simple est placé sans séparation. L'extension `numprint` détecte automatiquement cela si l'on utilise la commande `\tcdegree` de l'extension `mathcomp`, la commande `\textdegree` de l'extension `textcomp` ou la commande `\degree` de l'extension `gensymb`, p. ex.

```
\numprint[\tcdegree]{360}, \numprint[\degree]{360}
```

Malheureusement si l'on saisit directement le signe « ° », il n'est pas détecté (toute aide à ce sujet est bienvenue).

On peut changer l'espace entre le nombre et le signe d'unité, pourcentage ou degré. Voir la section 8.1.

Si l'on désire utiliser un des symboles de `textcomp` – ℃ (`\textcelsius`), Ω (`\textohm`), µ (`\textmu`), ou ‰ (`\textperthousand`) – on doit charger soit l'extension `mathcomp` soit l'extension `gensymb` parce que les unités sont composées en

3. Cela n'apparaitra que si l'extension `eco` est disponible sur la machine.

mode mathématiques. Si l'on décide d'utiliser l'extension `textomp`, on se servira de `\tc`⟨*name*⟩ et non de `\text`⟨*nom*⟩, p. ex. `\tccelsius` et non `\textcelsius`. Si l'on utilise `gensymb`, on saisit simplement `\`⟨*nom*⟩ – p. ex. `\celsius`.[4]

`\np` Comme saisir la commande `\numprint` au long pour chaque nombre présent dans le texte prend du temps, l'abréviation `\np` peut être définie en passant l'option `np` à l'extension.

Par défaut le format d'écriture des nombres est $12\,345{,}123\,45 \cdot 10^{12\,345}$. Cela signifie : signe décimal « `,` », séparateur des milliers « `\,` » et signe du produit « `{}\cdot{}` ». Tout cela conforme au format des nombres en allemand, du fait de l'histoire de cette extension. La manière de changer ce comportement est décrite dans les sections suivantes. Regardez spécialement l'option `autolanguage` de l'extension à la section 5.

2.1 Imprimer des compteurs et des longueurs

Si l'on désire imprimer (la valeur) d'un compteur ou d'une longueur, on peut bien sûr écrire

```
\numprint{\arabic{page}}
```

ou

```
\makeatletter
\numprint[pt]{\strip@pt\textwidth}
\makeatother
```

respectivement. Mais ces méthodes ne sont guère conviviales.

`\cntprint` Pour imprimer des compteurs, on peut utiliser `\cntprint[`⟨*unité*⟩`]{`⟨*compteur*⟩`}` où ⟨*unité*⟩ est une unité optionnelle qui est imprimée comme avec `\numprint`. ⟨*compteur*⟩ est le nom d'un compteur LaTeX, par exemple

```
$\cntprint{page}$
```

produit « 5 ».

`\lenprint` On peut imprimer les longueurs avec `\lenprint[`⟨*unité*⟩`]{`⟨*longueur*⟩`}`. ⟨*longueur*⟩ est une macro contenant une longueur de LaTeX ou une dimension de TeX, p. ex. `\textwidth`. ⟨*unité*⟩ joue ici un rôle différent. Si l'unité n'est pas donnée, `\lenprint` utilise la norme LaTeX et imprime la longueur en `pt` :

```
$\lenprint{\textwidth}$
```

produit « 355 pt ». En donnant `pt` on ne change rien. Mais si l'on donne une autre unité comme `mm`, `cm` ou `in`, la longueur est écrite dans ladite unité :

```
$\lenprint[pt]{\textwidth}$,
$\lenprint[in]{\textwidth}$,
$\lenprint[bp]{\textwidth}$,
$\lenprint[mm]{\textwidth}$,
$\lenprint[cm]{3\textwidth}$
```

4. Si l'on persiste à utiliser`\text`⟨*nom*⟩, numprint se servira de `\tc`⟨*nom*⟩ or `\`⟨*nom*⟩ et donnera un avertissement si l'une de ces commandes est disponible ou produira une erreur sinon.

produit « 355 pt, 4,913 1 in, 353,672 87 bp, 124,766 77 mm, 37,425 16 cm ».

Comme on le voit dans l'exemple, on peut également spécifier un facteur dans l'argument, p. ex. `3\textwidth`. Quand on imprime des longueurs, on désire souvent ne pas écrire toutes les décimales. `numprint` peut arrondir les nombres, cela est décrit dans la section 3.4.

`\lenprint` connait les unités `pt`, `bp`, `in`, `ft`, `mm`, `cm`, `m` et `km`. La section 4 décrit comment ajouter ou changer les unités.

3 Personnalisation

3.1 Éliminer les séparateurs pour les nombres de quatre chiffres

Au moins en allemand, il est fréquent de ne pas placer de séparateur de milliers dans les nombres de quatre chiffres des textes non-techniques, p. ex. on écrit « 1234 » plutôt que « 1 234 », mais les nombres plus longs sont coupés : « 12 345 ». Si, en revanche, un nombre a moins de cinq chiffre d'un côté du séparateur décimal mais cinq chiffres ou plus de l'autre côté, des séparateurs sont insérés de chaque côté, p. ex. « 1234,1234 » mais « 1 234,123 45 ».

`\npfourdigitsep` `\npfourdigitnosep`

On obtient ce comportement à l'aide de la commande `\npfourdigitnosep`. En plaçant ce commutateur dans un groupe, le changement est local. On peut revenir à la séparation avec `\npfourdigitsep`. Par exemple

```
\npfourdigitnosep$\numprint{1234.1234}$, $\numprint{12345.12345}$ --
\npfourdigitsep$\numprint{1234.1234}$, $\numprint{12345.12345}$
```

produit « 1234,1234, 12 345,123 45 – 1 234,123 4, 12 345,123 45 ». On fixe les valeurs par défaut en passant à l'extension l'option `sepfour` ou `nosepfour`.

3.2 Ajouter des zéros avant ou après le signe décimal

`\npaddmissingzero` `\npnoaddmissingzero`

Parfois les gens n'écrivent pas un zéro initial ou un zéro après le signe décimal comme dans « `123.` » ou « `.123` ». Numprint peut ajouter ces zéros omis lorsque l'on utilise `\addmissingzero`. Si toutefois on ne saisit pas de signe décimal, comme dans « `123` » numprint n'ajoute ni zéro ni signe décimal. On peut supprimer l'ajout de zéros avec `\noaddmissingzero`. Les options correspondantes sont `addmissingzero` et `noaddmissinzero`. Par défaut on a `addmissingzero`.

3.3 Ajouter un signe plus

`\npaddplus` `\npnoaddplus`

Avec la commande `\npaddplus` ou l'option `addplus` on peut ajouter un signe plus devant un nombre qui est donné sans signe. On peut supprimer ce comportement avec la commande `\npnoaddplus` ou l'option `noaddplus`.

`\npaddplusexponent` `\npnoaddplusexponent`

Les commandes `\npaddplusexponent` et `\npnoaddplusexponent` et les options correspondantes `addplusexponent` et `noaddplusexponent` font pour les exposants ce que les commandes et options décrites ci-avant font pour les nombres.

3.4 Arrondir les nombres

`\nprounddigits` `\nproundexpdigits` `\npnoround` `\npnoroundexp`

Par défaut, il y a autant de chiffres imprimés après le signe décimal que de décimales dans l'argument donné à `\numprint`. On peut changer ce comportement

et obtenir un nombre donné de décimales en arrondissant le nombre en argument ou en complétant par des zéros si nécessaire.

On met en route ce comportement avec la commande `\nprounddigits{`⟨*chiffres*⟩`}` pour les nombres ordinaires et `\nproundexpdigits{`⟨*chiffres*⟩`}` pour les exposants. On met fin à l'arrondissement [5] avec `\npnoround` ou `\npnoroundexp` selon les cas. Par exemple

```
\nprounddigits{2}$\numprint{1.123}$, $\numprint{1.149}$,
$\numprint{1}$, $\numprint{9.999}$ $\numprint{-9.999}$ --
\npnoround$\numprint{1.123}$, $\numprint{1.149}$,
$\numprint{1}$, $\numprint{9.999}$, $\numprint{-9.999}$
```

produit « 1,12, 1,15, 1,00, 10,00 −10,00 – 1,123, 1,149, 1, 9,999, −9,999 ».

3.5 Remplissage du côté gauche

`\nplpadding` `\npnolpadding`

Parfois il est souhaitable que les nombres aient une longueur fixe les chiffres manquants étant remplacés par un caractère – le plus souvent le « 0 » qui est donc la valeur par défaut. On obtient cela avec `\nplpadding[`⟨*caractère*⟩`]{`⟨*chiffres*⟩`}`. Cela n'affecte que la mantisse, on ne compte pas la partie qui suit le signe décimal ni les signes ! Si le nombre originel a plus de chiffres que ⟨*chiffres*⟩ aucun caractère n'est ajouté. Par exemple

```
\nplpadding{6}%
$\numprint{1234}$, $\numprint{-1234}$,
$\numprint{12345678}$, $\numprint{1234.5678}$ --
\nplpadding[x]{6}%
$\numprint{1234}$, $\numprint{-1234}$,
$\numprint{12345678}$, $\numprint{1234.5678}$
```

produit « 001 234, −001 234, 12 345 678, 001 234,567 8 – $xx1$ 234, $-xx1$ 234, 12 345 678, $xx1$ 234,567 8 ».

`\npnolpadding` met fin au remplissage.

3.6 Remplacer zeros par d'autres caractères

`\npreplacenull` `\npprintnull`

Pour les sommes d'argent on remplace parfois les zéros qui suivent le signe décimal par différents symboles comme « — ». On obtient cela en appelant la commande `\npreplacenull{`⟨*replacement*⟩`}`, p. ex.

```
\npreplacenull{\mbox{---}}
```

Ici `\mbox` assure que « — » est imprimé en mode texte. [6]

On y met fin avec `\npprintnull`.

3.7 Changer le format des unités

`\npunitcommand`

Par défaut, l'unité est imprimé en mode mathématique avec une fonte droite (upright). Cela est obtenu à l'aide de la macro `\npunitcommand` qui, par défaut, est définie comme suit :

```
\newcommand*\npunitcommand[1]{\ensuremath{\mathrm{#1}}}
```

5. Pour ceux qui en douterait, je dégaine le Littré pour signaler que le premier sens d'arrondissement c'est : action d'arrondir [TdS].

6. On devrait plutôt utiliser `amsmath` et la commande `\text` qui permet aussi de conserver la taille de caractères correcte.

Si l'on veut changer cela, on redéfinit la commande `\npunitcommand`. On devrait vraiment utiliser soit `\ensuremath` soit `\text`[7] pour garantir le mode mathématique ou texte respectivement. `\npunitcommand` doit prendre un argument obligatoire.

Par exemple, on obtient une unité bleu avec

```
\renewcommand*\npunitcommand[1]{\ensuremath{\color{blue}\mathrm{#1}}}
```

Et voici le résultat : 300 N/mm^2

4 Ajouter des unités à `\lenprint`

`\npdefunit` On peut utiliser la commande `\npdefunit{`⟨*nom d'unité*⟩`}{`⟨*unité*⟩`}{`⟨*échelle*⟩`}` pour définir de nouvelles unités ou redéfinir des unités existantes. Par exemple, `mm` est défini comme suit :

```
\npdefunit{mm}{mm}{0.35145980351}
```

Le premier argument ⟨*nom d'unité*⟩ est le nom interne à LaTeXqui sera donné comme unité à la commande `\lenprint`. Le deuxième argument ⟨*unité*⟩ est le texte qui sera imprimé.

L'⟨*échelle*⟩ vient du fait que l'unité par défaut pt est définie comme 1 pt = 1/72,27 in = 0,013 837 000 13 in et 1 in = 25,4 mm. Ainsi 1 pt = 0,013 837 000 13× 25,4 mm = 0,351 459 803 51 mm. Pour définir une nouvelle unité, on doit calculer la valeur en pt de la nouvelle unité et donner ce nombre comme troisième argument de `\npdefunit`. Pour redéfinir une unité sans changer l'échelle, utiliser `*` comme troisième argument. Par exemple

```
\npdefunit{in}{\!''}{*}
```

redéfinit `in` pour que ″ soit imprimé comme symbole d'unité au lieu du in obtenu par défaut. On se sert de `\!` pour supprimer le séparateur – `\,` – placé entre le nombre et l'unité. Cela ne fonctionne correctement que si `\,` et `\!` utilisent la même mesure.

```
$\lenprint[in]{\textwidth}$,
\npdefunit{in}{\!''}{*}%
$\lenprint[in]{\textwidth}$
```

produit « 4,913 1 in, 4,913 1″ ».

5 Internationalisation

Comme signalé ci-dessus, `numprint` utilise la présentation allemande des nombres : séparateur de milliers « `\,` », signe décimal « `,` », signe de produit « `\cdot` », séparateur d'unité `\,` et aucun séparateur de degré par défaut. Cela restera inchangé pour assurer la compatibilité avec les versions plus anciennes même si c'est illogique puisque la langue par défaut de LaTeX est l'anglais.

Avec l'option `autolanguage` de l'extension on peut remédier à cela. Si l'on utilise cette option sans l'extension `babel` les réglages sont ceux de l'anglais dès le `\begin{document}`, cf. la table 1.

Table 1 – Valeurs par défaut pour les réglages de langue

langue	sép. de millier	signe décimal	signe produit	sép. d'unité	sép. de pourcent	sép. de degré
anglais	,	.	\times	\,	\,	aucun
allemand	\,	,	\cdot	\,	\,	aucun
portuguais	.	,	\cdot	\,	\,	aucun
néerlandais	\,	,	\cdot	\,	aucun	aucun
français	~	,	\cdot	\,	\,	aucun

\selectlanguage

Si l'on utilise l'extension babel avec l'option autolanguage, le comportement de \numprint s'adapte à la langue active. Par exemple, en utilisant

```
\selectlanguage{ngerman}
```

on sélectionne les réglages allemands. Si l'on repasse à l'anglais ensuite, les réglages anglais seront de nouveau actifs.

La présente version gère l'anglais, l'allemand, le portugais, le néerlandais et le français. Malheureusement je ne sais pas vraiment comment écrire les nombres dans d'autres langues que l'allemand. Je suis presque certain que la version anglaise est également correcte. [8] Mais aidez-moi s'il vous plait à ajouter d'autres langues.

Si l'on veut d'autres réglages que ceux fournis par défaut – repris dans le tableau 1 – on se reportera à la section 8.1.2.

Tant que numprint ne gère pas une langue, on peut ajouter les définitions soi-même. On décrit comment procéder dans la section 8.1.2.

6 Imprimer des nombres alignés dans des tableaux

L'alignement de nombres dans les tableaux est fournit par les extension dcolumn et rccol. Mais elles ont deux inconvénients. D'abord, elles ne gèrent pas la composition des nombres comme le fait \numprint. Ensuite, elles oblignet à saisir les nombres en mode mathématique. Aussi, cette extension fournit ses propres mécanismes pour l'alignement des nombres.

Dans les versions précédentes et jusqu'à la version 1.00, le mécanisme d'alignement dans les tableaux était plutôt faible car il obligeait l'auteur à répéter l'appel à \numprint dans chaque cellule du tableau. On a amélioré la situation dans la version 1.10. Pour des raisons de compatibilité, les types de colonnes n et N sont conservés si l'on passe l'option oldcolumntypes à l'extension. On les présentera à la section 6.2.

6.1 Les nouveaux types de colonne

Le type de colonne n

L'extension numprint fournit le type de colonne n qui prend deux arguments obligatoires. Ils définissent le nombre de chiffres avant et après le signe décimal. On peut voir ce qui en vient dans la colonne de gauche du tableau ci-dessous. On peut utiliser ce type de colonne de la même façon que les types normaux, p. ex.

7. Fournie par amsmath.

8. Peut-être que le séparateur du signe de pourcentage \, n'est pas commun mais j'en laisserai la valeur inchangée pour conserver la compatibilité avec les versions précédentes.

```
\begin{tabular}{n{3}{4}n{4}{2}}
```

Les nombres sont imprimés dans l'espace réservé avec la largeur nécessaire aux nombres de chiffres spécifiés, alignés sur le signe décimal. Si une colonne contient des nombres qui ont un exposant, ce dernier est ajouté avec un alignement à gauche et la largeur de la colonne est étendu de l'espace requis. On le voit dans la première colonne de l'exemple ci-dessous.

Si, en outre, on veut réserver de l'espace pour les chiffres de l'exposant, on peut insérer un – fixant le nombre de chiffres avant le signe décimal – ou deux – nombre de chiffres avant et après le signe décimal – arguments facultatifs comme on peut le voir dans les colonnes 2 et 3 de l'exemple ci-dessous. Si l'on réserve de l'espace pour l'exposant, les exposants trop long peuvent déborder de la cellule – ce que l'on peut voir dans la 2e colonne.

Cet exemple de tableau [9]

$123{,}45 \cdot 10^{12}$	$123{,}45 \cdot 10^{12}$	$123{,}45 \cdot 10^{12}$	$123{,}45 \cdot 10^{12}$
$12\,345{,}678 \cdot 10^{123}$	$12\,345{,}678 \cdot 10^{123}$	$12\,345{,}678 \cdot 10^{123}$	$12\,345{,}678 \cdot 10^{123}$
$123{,}45 \cdot 10^{12{,}3}$	$123{,}45 \cdot 10^{12{,}3}$	$123{,}45 \cdot 10^{12{,}3}$	$123{,}45 \cdot 10^{12{,}3}$
$12\,345{,}678 \cdot 10^{123{,}3}$	$12\,345{,}678 \cdot 10^{123{,}3}$	$12\,345{,}678 \cdot 10^{123{,}3}$	$12\,345{,}678 \cdot 10^{123{,}3}$

est produit avec le code suivant

```
\tabcolsep0mm\small
\begin{tabular}{|n{5}{3}|n[3]{5}{3}|n[3][1]{5}{3}|N{5}{3}|}
  \hline
  123.45e12& 123.45e12& 123.45e12 & 123.45e12 \\
  12345.678e123& 12345.678e123& 12345.678e123& 12345.678e123 \\
  123.45e12.3& 123.45e12.3& 123.45e12.3& 123.45e12.3 \\
  12345.678e123.3& 12345.678e123.3& 12345.678e123.3& 12345.678e123.3 \\
  \hline
\end{tabular}
```

Le type de colonne `n` imprime les nombres en mode mathématique c'est pourquoi les trois premières colonnes de l'exemple montrent des chiffres courants (lining).

`Type de colonne N` Pour imprimer les nombres en mode texte, on peut utiliser le type de colonne `N` comme le montre la 4e colonne du tableau. [10] Il prend les mêmes arguments que le type de colonne `n`.

On peut ajouter du texte dans les cellules du tableau. L'extension `numprint` utilise un algorithme pour déterminer quelle partie est un nombre et quelle partie est du texte supplémentaire. Afin de préserver les espaces ou pour pouvoir utiliser dans le texte des caractères qui pourraient être des parties d'un nombre – c.-à-d. des chiffres ou les lettres « e », « E », « d » ou « D » – on doit placer le texte entre accolades, par exemple « `{hello}` 1234 » au lieu de « `hello` 1234 ». Dans quelques rares cas, il faut même deux paires d'accolades. C'est le cas si on veut utiliser un caractère unique qui pourrait être également utilisé dans un nombre, p. ex. « `{{3}}` 1234 ». [11] Si l'on n'utilise pas d'accolades, d'étranges effets peuvent être observés. On doit s'arranger pour que le texte qui précède ait la même longueur pour toutes les lignes de la même colonne. Regardez l'exemple suivant :

9. Ici, les tableaux sont affreux mais l'important est de montrer les effets des alignements.

10. Si l'on utilise un environnement de tableau qui imprime ses arguments en mode mathématique, p. ex. l'environnement `array`, `N` imprime également en mode mathématique.

11. Il en est de même si le texte commence avec quelques une des commandes spéciales qui n'apparaissent pas d'ordinaire au début d'une cellule : `\end`, `\tabularnewline`, `\nprt@end`, `\endtabular`, `\csname` et `\relax`.

```
\begin{tabular}{n{2}{1}n{2}{1}n{2}{1}n[1]{2}{1}}
  \toprule
  {sans accolades}&
  {avec accolades}&
  {accolades et boite}&
  \multicolumn{1}{l}{accolades, exp et boite}
  \\
  \midrule
  abc def 12,3e3 rt&
  {abc def } 12,3e3 { rt}&
  {\npmakebox[abc def ][l]{abc def }} 12,3e3 { rt}&
  {\npmakebox[abc def ][l]{abc def }} 12,3e3 { rt}
  \\
  more 45,1 txt&
  {more } 45,1 { txt}&
  {\npmakebox[abc def ][l]{more }} 45,1 { txt}&
  {\npmakebox[abc def ][l]{more }} 45,1 { txt}
  \\
  \midrule
  not blue 45,1 txt&
  {\color{blue}blue } 45,1 { txt}&
  {\color{blue}\npmakebox[abc def ][l]{blue }} 45,1 { txt}&
  {\color{blue}\npmakebox[abc def ][l]{blue }} 45,1 { txt}
  \\
  \bottomrule
\end{tabular}
```

Le résultat apparait comme ceci :

sans accolades	avec accolades	accolades et boite	accolades, exp et boite
abc*de*f12,3e3rt mor$10^{45,1}$txt	abc def $12{,}3 \cdot 10^3$ rt more 45,1 txt	abc def $12{,}3 \cdot 10^3$ rt more 45,1 txt	abc def $12{,}3 \cdot 10^3$ rt more 45,1 txt
notblu$10^{45,1}$txt	blue 45,1 txt	blue 45,1 txt	blue 45,1 txt

Dans la première colonne, les textes précédant et suivant le nombre ne sont pas entourés d'accolades. D'étranges effets en résultent : dans la première ligne, « `de` » est interprété comme un nombre aussi est-il imprimé en mode mathématique et le reste de la cellule, « `f 12,3e3 rt` », est imprimé comme du texte à nouveau. Dans la deuxième ligne, la partie « `e 45,1` » est imprimée comme un nombre.

On peut éviter ce comportement étrange en plaçant les textes entre accolades comme on l'a signalé ci-dessus et comme le montre la deuxième colonne. De cette manière, les textes sont correctements imprimés. Un espace placé dans le texte entre accolades peut séparer le texte du nombre comme on le fait dans l'exemple.

Parce que les textes de gauche ont des longueurs différentes, l'alignement des nombres est cassé. On peut remédier à cela en plaçant les contenus dans des boites à l'aide des commandes `\makebox` ou `\npmakebox` comme le montre la troisième colonne. La commande `\npmakebox` a une syntaxe semblable à celle de `\makebox` mais utilise un texte au lieu d'une longueur pour déterminer la largeur de la boite :`\npmakebox[`⟨*texte 1*⟩`][`⟨*justification*⟩`]{`⟨*texte 2*⟩`}`. La commande détermine qu'elle serait la largeur de ⟨*texte 1*⟩ et compose ⟨*texte 2*⟩ dans une boite de cette largeur.

`\npmakebox`

Si l'on place le texte précédent de chaque ligne dans une telle boite les nombres seront alignés comme on le désirait.

Cependant, le texte à droite du nombre n'est toujours pas aligné. On remédie à cela en précisant le nombre de chiffres de l'exposant comme on le montre dans la quatrième colonne.

La troisième ligne diffère quelque peu des autres. Ici, non seulement on insère du texte mais aussi une commande qui change l'aspect du nombre imprimé. Une telle commande doit être, elle aussi, placée entre accolades.

`\multicolumn` Pour produire des cellules qui ne contiennent pas de nombre on peut soit placer le contenu de la cellule entre accolade soit utiliser la commande `\multicolumn` comme le montre la première ligne de l'exemple ci-dessus.

Si l'on veut imprimer une ligne en gras avec la commande `\boldmath` en mode mathématique ou `\bfseries` en mode texte, l'alignement n'est plus correct une fois de plus. Cela est dû au fait que les caractères gras sont plus larges que les caractères normaux. On peut éviter ce problème si la famille de polices fournit une police grasse de même largeur que la police normale. Une telle police existe pour les Computer Modern. En mode texte, on l'utilise avec

```
\fontseries{b}\selectfont
```

`\mathversion` `\npboldmath` Si l'on veut se servir de cette police en mode mathématique aussi, on peut utiliser la version mathématique `npbold` à l'aide de `\mathversion{npbold}` ou `\npboldmath`. Pour épargner la mémoire, ces commandes et la version mathématique `npbold` ne sont disponibles que si l'on charge `numprint.sty` avec l'option `boldmath`.

Un exemple :

```
\begin{tabular}{lN{12}{3}n{12}{3}}
  \toprule
  normal :&
  123456123456.123e12&
  123456123456.123e12
  \\
  gras :&
  {\fontseries{b}\selectfont} 123456123456.123e12&
  {\npboldmath} 123456123456.123e12
  \\
  gras étendu:&                          %%% ce n'est PAS étendu !
  {\bfseries} 123456123456.123e12&
  {\boldmath} 123456123456.123e12
  \\
  \bottomrule
\end{tabular}
```

qui produit

normal :	$123\ 456\ 123\ 456{,}123 \cdot 10^{12}$	$123\ 456\ 123\ 456{,}123 \cdot 10^{12}$
gras :	$\mathbf{123\ 456\ 123\ 456{,}123 \cdot 10^{12}}$	$\mathbf{123\ 456\ 123\ 456{,}123 \cdot 10^{12}}$
gras étendu :	$\mathbf{123\ 456\ 123\ 456{,}123 \cdot 10^{12}}$	$\mathbf{123\ 456\ 123\ 456{,}123 \cdot 10^{12}}$

Si l'on veut ajouter le même texte ou les mêmes commandes à chaque ligne d'une colonne d'un tableau, on peut utiliser le spécificateur « > » dans la déclaration de la colonne, comme d'habitude. On doit cependant entourer l'argument d'une paire d'accolades supplémentaire comme le montre l'exemple ci-dessous. Mal-

`\npafternum` heureusement, le spécificateur « < » ne fonctionne pas convenablement. De ce fait,

`numprint` définit la commande `\npafternum` dont l'argument est imprimé après le nombre. L'exemple suivant montre du texte avant et après le nombre.

```
\begin{tabular}{%
  >{{avant \npafternum{ après}}}n[2]{12}{3}%
  >{{\nprounddigits{4}}}n{3}{4}%
  >{{\color{blue}}}n{12}{3}}
  \toprule
  123456123456.123e12&
  12.12345&
  123456.23e1
  \\
  12345.123e12&
  12.1&
  14561234.562e12
  \\
  \bottomrule
\end{tabular}
```

qui produit :

avant 123 456 123 456,123 · 10^{12} après	12,124	123 456,23 · 10^{1}
avant 12 345,123 · 10^{12} après	12,100	14 561 234,562 · 10^{12}

Comme on peut le voir dans la deuxième colonne, on peut arrondir les nombres dans certaines colonnes en plaçant un `\nproundigits` dans une spécification « > ». On peut en faire autant avec toutes les commandes qui jouent sur le format des nombres.

`\npunit` Normalement on ne devrait pas placer des unités dans les cellules d'un tableau mais, si nécessaire, on peut le faire avec la commande `\npunit` comme on le montre ici :

```
\begin{tabular}{>{{\npunit{N/mm^2}}}n{5}{3}}
  \toprule
  12345.123\\
  12.12\\
  {\npunit{psi}} 234.4\\
  4.3\\
  \bottomrule
\end{tabular}
```

qui produit :

12 345,123	N/mm^2
12,12	N/mm^2
234,4	psi
4,3	N/mm^2

L'alignement dans les tableaux fournit par l'extension `numprint` a été testé dans les environnements `tabular`, `tabular*`, `array`, `tabularx` [1] et `longtable` [2]. Son fonctionnement avec d'autres environnements ou extensions n'est pas garanti.

6.2 Les anciens types de colonne

`Type de colonne N` Si l'on a passé l'option `oldcolumntype` à l'extension, les types de colonne `n` et

`N` sont définis d'une autre façon que ce qui a été décrit dans la section précédente.

`Type de colonne n` Le type `n` aligne la base sur le signe décimal et le type `N` aligne en plus l'exposant.

On utilise ces commandes comme suit :

```
\tabcolsep0mm
\begin{tabular}{|n{5}{3}|N{5}{3}{3}|}
  \hline
  \numprint{123.45e12}& \numprint{123.45e12}\\
  \numprint{12345.678e123}& \numprint{12345.678e123}\\
  \numprint{123.45e12.3}& \numprint{123.45e12.3}\\
  \numprint{12345.678e123.3}& \numprint{12345.678e123.3}\\
  \hline
\end{tabular}
```

Ce qui produit :

$123{,}45 \cdot 10^{12}$	$123{,}45 \cdot 10^{12}$
$12\,345{,}678 \cdot 10^{123}$	$12\,345{,}678 \cdot 10^{123}$
$123{,}45 \cdot 10^{12{,}3}$	$123{,}45 \cdot 10^{12{,}3}$
$12\,345{,}678 \cdot 10^{123{,}3}$	$12\,345{,}678 \cdot 10^{123{,}3}$

Le premier argument définit le nombre de chiffres avant le signe décimal, le second le nombre de chiffres après. Dans le cas du type `N` le troisième argument définit le nombre de chiffres de l'exposant précédant le signe décimal. On ne peut pas préciser le nombre de chiffres après le signe décimal dans l'exposant, il est fixé à zéro. On notera qu'il faut, quand on utilise les types anciens de colonne, écrire la commande `\numprint` dans chaque contenu de cellule.

6.3 Alignement dans le texte normal

L'alignement des nombres dans les tableaux est obtenu en écrivant les nombres dans des boites de largeur déterminée. On peut utiliser ce comportement également en dehors des environnements de tableaux.

`\npdigits` La macro `\npdigits{`⟨*avant*⟩`}{`⟨*après*⟩`}` déclenche l'alignement des nombres composés par `\numprint`. Le premier argument définit le nombre de chiffres avant le signe décimal tandis que le second définit le nombre de chiffres après ce signe

`\npexponentdigits` dans la mantisse. Comme les exposants sont normalement des entiers la syntaxe de la commande correspondante `\npexponentdigits` est légèrement différente, c'est `\npexponentdigits[`⟨*après*⟩`]{`⟨*avant*⟩`}`. L'argument obligatoire définit le nombre de chiffres avant le signe décimal de l'exposant. Si aucun argument facultatif n'est fournit, le nombre de chiffres après le signe décimal est fixé à zéro. Si l'on en fournit un, c'est le nombre de chiffres après le signe décimal.

Si les commandes `\npdigits` et `\npexponentdigits` sont utilisées dans un groupe, les paramètres reprennent leurs valeurs antérieurs à la fin du groupe.

`\npnodigits` On peut faire cesser l'alignement de la mantisse ou de l'exposant à l'aide de la

`\npnoexponentdigits` commande `\npnodigits` ou `\npnoexponentdigits` selon le cas.

7 Messages d'erreur etc.

Par défaut, `\numprint` produit un message d'erreur si l'argument utilise des caractères invalides ou si le format de nombre est invalide. Quelque personne utilise

\numprint pour réaliser des choses étranges et donc se servent d'arguments invalides exprès. Ces personnes pourront supprimer ces messages d'erreur en passant l'option warning à l'extension.

Si l'on souhaite que des messages de débogage soient écrit dans le fichier de rapport (log), on utilisera l'option debug.

8 Personnalisation avancée

8.1 Changer la sortie

La plupart des choses décrites dans cette section ne doivent pas forcément être faites à la main puisque la fonction de « gestion automatique des langues » – cf. section 5 – les fait automatiquement.

8.1.1 Sans l'option autolanguage

\npthousandsep
\npthousandthpartsep
\npdecimalsign
\npproductsign

On peut, à l'aide des commandes [12],
- \npdecimalsign{⟨*signe*⟩} ;
- \npthousandsep{⟨*séparateur*⟩} ;
- \npthousandthpartsep{⟨*séparateur*⟩} ;
- \npproductsign{⟨*signe*⟩},

changer plusieurs séparateurs et symboles, par exemple

```
\npdecimalsign{\ensuremath{\cdot}}\npthousandsep{,}\npproductsign{*}%
\numprint{-123456}; \numprint{3,1415927e-3.2}
```

produit « $-123{,}456$; $3{\cdot}141{,}592{,}7 * 10^{-3{\cdot}2}$ ».

Avec \npthousandsep on modifie les séparateurs de milliers avant et après le signe décimal. Si l'on veut utiliser des séparateurs différents, on appellera \npthousandthpartsep *après* \npthousandsep.

Les séparateurs, comme le signe décimal, sont composés dans le mode dans lequel est composé le nombre lui-même (mode texte ou mathématique). Si l'on veut garantir un mode spécial, on devra utiliser \ensuremath pour le mode mathématique ou soit \mbox, soit \textrm soit encore \text [13] pour le mode texte.

Le signe de produit est, par contre, toujours composé en mode mathématique. Aussi, on n'a pas besoin d'ajouter un \ensuremath pour utiliser des commandes du mode mathématiques – comme \times p. ex.

\global

En utilisant ces commandes dans un groupe ({...},\begingroup...\endgroup, ou un environnement), on ne change le comportement de \numprint que localement – à l'intérieur du groupe en question. En les préfixant par \global on peut, tout en étant dans un groupe, rendre les modifications globales. Par exemple :

```
Local:
\numprint{123e4},
{\npproductsign{\cdot}\numprint{123e4}},
\numprint{123e4}.
Global:
\numprint{123e4},
```

12. Les noms de ces commandes n'avaient pas de « np » dans les versions antérieures. On a dû changer cela pour éviter des incompatibilités avec l'option frenchb de babel.

13. Que fournit l'extension amsmath.

```
{\global\npproductsign{\cdot}\numprint{123e4}},
\numprint{123e4}
```

produit ce qui suit :
Local : $123 \cdot 10^4$, $123 \cdot 10^4$, $123 \cdot 10^4$. Global : $123 \cdot 10^4$, $123 \cdot 10^4$, $123 \cdot 10^4$

La version actuelle a pour valeurs par défaut ce qui suit :

```
\npthousandsep{\,}
\npdecimalsign{,}
\npproductsign{\cdot}
\npunitseparator{\,}
```

`\npunitseparator` L'espace entre le nombre et l'unité est « `\,` » par défaut. On peut le modifier à l'aide de `\npunitseparator{`⟨*séparateur*⟩`}`, p. ex.

```
\npunitseparator{~}
```

`\npdegreeseparator` Par défaut, on n'ajoute pas d'espace entre le nombre et le symbole du degré. On peut spécifier un séparateur à l'aide de `\npdegreeseparator{`⟨*séparateur*⟩`}`.

`\npcelsiusseparator` Par défaut, l'espace ajouté entre un nombre et un °C est le même que celui utilisé pour une unité normale. On peut spécifier un séparateur différent avec `\npcelsiusseparator{`⟨*séparateur*⟩`}`. On notera que `numprint` ne connait pas `\tccentigrade`.

`\nppercentseparator` Par défaut, l'espace ajouté entre un nombre et le signe « pourcent » est l'espace pour une unité normale. On peut spécifier un séparateur différent avec `\nppercentseparator{`⟨*séparateur*⟩`}`.

8.1.2 Avec l'option `autolanguage`

Si l'on utilise l'option `autolanguage` les modifications, faites par les commandes décrites dans la section précédente, sont perdues au changement de langue suivant ou au `\begin{document}`. Aussi, on ne peut, avec cette option, les utiliser comme on vient de le décrire.

Aussi, on doit redéfinir les commandes qui fixent les paramètres de `numprint` dépendants de la langue. Pour chacune des langues connues, une commande `\npstyle`⟨*langue*⟩ est définie qui produit les modifications, p. ex. `\npstyleenglish` `\npstyleenglish` pour l'anglais. Cette commande est définie comme suit :

```
\newcommand*\npstyleenglish{%
  \npthousandsep{,}%
  \npdecimalsign{.}%
  \npproductsign{\times}%
  \npunitseparator{\,}%
  \npdegreeseparator{}%
  \npcelsiusseparator{\nprt@unitsep}%
  \nppercentseparator{\nprt@unitsep}%
}
```

Si l'on veut des réglages différents pour cette langue, on a deux possibilités.

1. Copier la définition de `\npstyle`⟨*language*⟩ dans le fichier de style et la modifier comme on l'entend. Par exemple [14] :

   ```
   \renewcommand*\npstyleenglish{%
   ```

14. N.B. : on doit utiliser `\renewcommand*` et non pas `\newcommand*`.

```
  \npthousandsep{,}%
  \npdecimalsign{{\cdot}}%
  \npproductsign{\times}%
  \npunitseparator{\,}%
  \npdegreeseparator{}%
  \npcelsiusseparator{\nprt@unitsep}%
  \nppercentseparator{\nprt@unitsep}%
}
```

2. Ajouter des réglages différents à la langue à l'aide de `\g@addto@macro` ce qui permet d'ajouter des commandes à une commande `\npstyle`⟨*langue*⟩ existante, p. ex.

```
\makeatletter
\g@addto@macro\npstyleenglish{%
  \nppercentseparator{}%
}%
\makeatother
```

   Cela présente l'avantage de ne pas perdre les modifications de la commande originelle lors d'une copie. L'inconvénient est que certaines commandes peuvent être appelées deux fois ce qui provoque un léger ralentissement.

Les modifications prennent effet la première fois où la commande de style est appelée ensuite, c.-à-d. quand on change de langue ou au `\begin{document}`.

Si l'on utilise une langue que `numprint` ne gère pas encore, on peut en ajouter la gestion dans le préambule du document.

`\npaddtolanguage` Le cas le plus simple est celui d'une langue qui utilise les mêmes réglages qu'une langue déjà gérée. Si, par exemple, on veut utiliser le danois avec les mêmes réglages que l'allemand, il suffit d'ajouter

```
\npaddtolanguage{danish}{german}
```

au préambule de son document.

Si l'on veut, au contraire, utiliser des réglages différents, on définira une commande de style adéquate. Prenons une fois de plus le danois comme exemple. Définissons une commande `\npstyledanish` qui règle tout ce que nous voulons changer des valeurs par défaut – je choisis des valeurs étranges pour raison de clarté – :

```
\newcommand*\npstyledanish{%
  \npthousandsep{.}%
  \npdecimalsign{\ensuremath{\cdot}}%
  \npproductsign{*}%
  \npunitseparator{~}%
  \npdegreeseparator{}%
  \npcelsiusseparator{\nprt@unitsep}%
  \nppercentseparator{\nprt@unitsep}%
}
```

De plus, ajoutons l'appel de cette commande à la commande de bascule pour le danois :

```
\npaddtolanguage{danish}{danish}
```

8.2 Changer l'analyse de l'argument

On a dit plus haut que les séparateurs de milliers n'étaient pas autorisés dans l'argument de `\numprint`. L'utilisateur peut modifier ce comportement.

`\nprt@dotlist` La commande `\numprint` utilise, pour la plupart des éléments de l'entrée, des listes qui contiennent les caractères correspondant. La macro `\nprt@dotlist` contient les caractères interprétés comme des signes décimaux. Elle est définie comme suit :

```
\newcommand*\nprt@dotlist{.,}
```

Si l'on désire, par exemple, n'autoriser que le point comme signe décimal, on redéfinira la liste :

```
\renewcommand*\nprt@dotlist{.}
```

Si l'on veut faire cette modification dans son document plutôt que dans le fichier de configuration `numprint.cfg` – cf. section 9.3 – on doit entourer tout cela par `\makeatletter` et `\makeatother`.

`\nprt@explist` La macro `\nprt@explist` contient les caractères interprétés comme délimiteurs entre la mantisse et l'exposant. Par défaut, elle contient « `eEdD` ». On la redéfinit comme on le fait pour `\nprt@dotlist`.

`\nprt@ignorelist` La macro `\nprt@ignorelist` contient la liste des caractères d'entrée ignorés – en plus des espaces, `\,` et `~`. Elle est vide par défaut. Si l'on désire, par exemple, n'autoriser que le point comme signé décimal et permettre la virgule comme séparateur de milleirs dans l'entrée, on peut utiliser la redéfinition suivante :

```
\renewcommand*\nprt@dotlist{.}
\renewcommand*\nprt@ignorelist{,}
```

Alors, « `,` » est ignoré dans l'entrée et « `.` » est interprété comme le signe décimal. Par exemple :

```
\makeatletter
{\renewcommand*\nprt@dotlist{.}%
\renewcommand*\nprt@ignorelist{,}%
\numprint{12,234.123,45e1,2,3.0}}
\makeatother
```

produit « 12 234,123 $45 \cdot 10^{123,0}$ ».

`\nprt@signlist` La macro `\nprt@signlist` contient la liste des signes connus. Par défaut, elle est fixée comme « `+-\pm` ». On peut modifier cette liste de signes reconnus en redéfinissant `\nprt@signlist`. Si, par exemple, le caractère « `*` » doit être connu comme un signe, on écrira :

```
\renewcommand*\nprt@signlist{+-\pm *}
```

Ce caractère sera imprimé quand on l'utilisera comme signe. Cela, toutefois, peut poser un problème : le signe peut être différent suivant le mode dans lequel il est composé comme le montre l'exemple suivant. En mode texte, `\numprint{*1234}` apparait comme « $^*{}_1$ 234 » alors qu'il est composé comme « $*1$ 234 » en mode mathématique. Pour éviter ce comportement, on doit définir une commande qui imprime le signe. Cette macro doit avoir un nom conforme à `\nprt@list@`⟨*signe*⟩.

`\nprt@sign@*` Dans le cas qui nous occupe, on définira `\nprt@list@*`. Comme le signe peut être un caractère arbitraire, on définira la commande comme suit :

```
\expandafter\newcommand\csname nprt@sign@*\endcsname{\ensuremath{*}}
```

Avec cette macro, `\numprint{*1234}` est imprimé comme « ∗1 234 » en mode texte et comme « ∗1 234 » en mode mathématique.

À cause de ce mécanisme d'impression du signe, on ne peut pas utiliser d'autres noms de macros que `\pm`[15] pour le signe. On doit utiliser un caractère unique comme le « * » traité ci-dessus.

9 Quelques trucs

9.1 Signe s'adaptant au mode

`\nprt@sign@+` `\nprt@sign@-` `\nprt@sign@+-`

Les signes par défaut sont composés en mode mathématiques, indépendamment du mode dans lequel sont composés les nombres. Lorsque l'on utilise une police dans laquelle les signes sont très différents en mode texte et en mode mathématique cela peut être gênant. Aussi, on peut utiliser des signes différents suivant le mode. Pour cela, les signes par défaut utilisent des macros du même genre que celles associées aux signes définis par l'utilisateur comme décrit dans la section 8.2. Ces macros sont définies comme suit :

```
\expandafter\newcommand\csname nprt@sign@+\endcsname{\ensuremath{+}}
\expandafter\newcommand\csname nprt@sign@-\endcsname{\ensuremath{-}}
\expandafter\newcommand\csname nprt@sign@+-\endcsname{\ensuremath{\pm}}
```

Si, par exemple, on ne veut pas utiliser le signe moins « mathématique » en mode texte mais le remplacer par un autre caractère, on peut redéfinir `\nprt@sign@-` :

```
\expandafter\renewcommand\csname nprt@sign@-\endcsname{%
  \ifmmode -\else ---\fi}
```

Avec cette définition, `\numprint{-1234}` produit « —1 234 » en mode texte et « −1 234 » en mode mathématique.

9.2 Composer les nombres négatifs en rouge

Si l'on veut imprimer les nombres négatifs en rouge, on peut utiliser la commande `\nprt@sign@-` [16]. L'exemple suivant montre de quelle façon procéder.

```
\usepackage{color}
\makeatletter
\expandafter\renewcommand\csname nprt@sign@-\endcsname{%
  \color{red}\ensuremath{-}}
\makeatother
```

Avec cette définition,

```
\numprint{1234}, \numprint{-1234},
\numprint{1234e-123}, \numprint{-1234e123}.
```

produit « 1 234, −1 234, $1\,234 \cdot 10^{-123}$, $-1\,234 \cdot 10^{123}$ ». [17] Pour éviter qu'un exposant négatif soit imprimé en rouge lorsque la mantisse est positive on a recourt à un bidouillage : la commande `\color` est redéfinit comme inopérante à l'intérieur du code qui compose l'exposant.

15. `\pm` est traité à part.
16. Voir l'annexe C [TdS].
17. L'apparence de la sortie dépend du visualisateur. En PostScript et en PDF, on voit la couleur rouge mais dans beaucoup de visualisateur de dvi, on ne la voit pas.

9.3 Fichier de configuration

Si l'installation LaTeX contient un fichier `numprint.cfg` dans le chemin de recherche de TeX, `numprint` le charge en dernier avant de rendre la main. Ainsi, on peut ajouter dans ce fichier toutes les modifications que l'on souhaite, comme de nouvelles langues par exemple.

A Compatibilité avec les versions antérieures

Dans la plupart des cas, les macros pour l'utilisateur de cette extension (celles dont le nom ne contient pas de `@`) devraient être compatibles avec les anciennes versions. L'analyse lexicale (*parsing*) de l'argument a été améliorée et certains arguments de `\numprint` pourraient être acceptés qui ne l'étaient pas avec une version antérieure, ou l'inverse.

L'espacement des nombres alignés a été corrigé également. Aussi, il y aura des incompatibilités avec les versions antérieurs si l'on utilise l'alignement des exposants ou dans des environnement mathématiques autres que `\textstyle`.

B Listes des options et commandes

Cette section contient les listes des options de l'extension et les commandes qu'elle fournit. Les options ou commandes mutuellement exclusives sont regroupées.

B.1 Options de l'extension

Les valeurs par défaut sont signalées par *.

`warning`	Produit des avertissements au lieu de messages d'erreur.
`error`*	Produit des messages d'erreur au lieu d'avertissements.
`autolanguage`	Enclenche l'adaptation automatique à la langue.
`noautolanguage`*	Réglages fixes.
`sepfour`*	Séparateur pour les nombres de quatre chiffres.
`nosepfour`	Pas de séparateur pour les nombres de quatre chiffres.
`addmissingzero`*	Ajoute les zéros manquants avant ou après le signe décimal.
`noaddmissingzero`	Ne le fait pas.
`addplus`	Ajoute un plus aux nombres sans signe.
`noaddplus`*	Ne le fait pas.
`addplusexponent`	Ajoute un plus à l'exposant quand il n'a pas de signe.
`noaddplusexponent`*	Ne le fait pas.
`oldcolumntypes`	Définit les types anciens de colonne qui demandent l'utilisation de `\numprint` dans les tableaux.
`newcolumntypes`*	Utilise les nouveaux types de colonne.
`boldmath`	Définit la version `npbold` mathématique.

`np`	Définit l'abréviation `\np` pour `\numprint`.
`debug`	Produit des information de débogage dans le fichier de rapport (`log`).

B.2 Commandes

`\npfourdigitsep`	Enclenche la séparation des nombres de quatre chiffres.
`\npfourdigitnosep`	Arrête la séparation des nombres de quatre chiffres.
`\npaddmissingzero`	Enclenche l'ajout des zéros manquants avant ou après le signe décimal.
`\npnoaddmissingzero`	Arrête l'ajout des zéros manquants avant ou après le signe décimal.
`\npaddplus`	Ajoute un plus aux nombres sans signe.
`\npnoaddplus`	Ne le fait pas.
`\npaddplusexponent`	Ajoute un plus à l'exposant quand il n'a pas de signe.
`\npnoaddplusexponent`	Ne le fait pas.
`\np`	Abréviation de `\numprint` (disponible uniquement si l'on a choisi l'option `np`).
`\numprint`	Compose un nombre (commande principale de l'extension).
`\npdecimalsign`	Change le signe décimal.
`\npthousandsep`	Change le séparateur de milliers (devant et derrière le signe décimal).
`\npthousandspartsep`	Change le séparateur de milliers (seulement après le signe décimal).
`\npproductsign`	Change le signe du produit.
`\npunitseparator`	Change le séparateur entre un nombre et une unité.
`\npdegreeseparator`	Change le séparateur entre un nombre et un symbole de degré.
`\npcelsiusseparator`	Change le séparateur entre un nombre et ℃.
`\nppercentseparator`	Change le séparateur entre un nombre et un signe de pourcentage.
`\nprounddigits`	Déclare combien de chiffres sont imprimés après le signe décimal.
`\npnoround`	Arrête l'arrondissement des nombres qui sont imprimés comme ils sont donnés.
`\nproundexpdigits`	Déclare combien de chiffres sont imprimés après le signe décimal de l'exposant.
`\npnoroundexp`	Arrête l'arrondissement de l'exposant.
`\nplpadding`	Déclare le nombre de chiffres de remplissage à gauche.
`\npnolpadding`	Arrête le remplissage à gauche.
`\npreplacenull`	Remplace la partie qui suit le signe décimal par un autre texte si elle est nulle.

`\npprintnull`	Imprime des zéros après le signe décimal.
`\npdigits`	Enclenche l'alignement des nombres avec un nombre de chiffres donnés avant et après le signe décimal.
`\npnodigits`	Compose les nombres dans une boite qui est aussi large que nécessaire.
`\npexponentdigits`	Enclenche l'alignement des exposants.
`\npnoexponentdigits`	Arrête l'alignement des exposants.
`\npaddtolanguage`	Ajoute des définitions à la section extra d'une langue de `babel`.
`\npstyledefault`	Définit les réglages dans le format normal.
`\npstylegerman`	Définit le format allemand des nombres.
`\npstyleenglish`	Définit le format anglais des nombres.
`\npmakebox`	Commande semblable à `\makebox` mais qui prend du texte au lieu d'une longueur comme premier argument facultatif.
`\npboldmath`	Version du gras mathématiques avec les chiffres de largeurs identiques aux maigres.
`\npafternum`	Place du texte après le nombre dans un tableau.
`\npunit`	Place les unités dans un tableau.

C Bogues connus

- Dans l'alignement des exposants à l'intérieur d'un tableau, la distance entre « 10 » et l'exposant est trop petite.
- L'impression en rouge des nombres négatifs ne fonctionne pas. Seul le signe moins est en rouge. Quand ce bogue fut-il introduit ?

D À faire

- Ajouter plus de langues pour la gestion automatique de l'internationalisation.
- Ajouter la gestion du « < » dans les définitions de tableaux.
- Éviter plusieurs des variables temporaires.
- Ajouter la gestion de l'alignement à droite et centré des nombres dans les tableaux.

E Remerciements

- Tilman Finke, tfinke@it-and-law.de, eut l'idée de l'arrondissement des nombres.
- Stephan Helma, s.p.helma@gmx.net, a implanté le remplissage à gauche des nombres. J'ai changé légèrement cette fonction.
- Gestion du portuguais par Vilar Camara Neto et Luis.
- Gestion du néerlandais par Ralph Hendriks.
- Gestion du français par Daniel Flipo.

Références

[1] Carlisle, David : *The* ***tabularx*** *package*, version 2.07, 1999. CTAN:macros/latex/contrib/tabularx/.

[2] Carlisle, David : *The* ***longtable*** *package*, version 4.10, 2000. CTAN:macros/latex/contrib/longtable/.

[3] Guthöhrlein, Eckhart : *The* ***rccol*** *package*, version 1.1a, 2000. CTAN:macros/latex/contrib/rccol/.

Historique des modifications

Comme la version 1.00 est une implantation totalement nouvelle, l'historique des changements antérieurs à la version 1.00 a été retiré de ce document. Vous pouvez consulter `numprint032.dtx` ou `README` pour la voir.