APERÇU DU PACKAGE NICEMATRIX

L'origine : les lignes pointillées

Dans ses premières versions (février 2018), l'extension nicematrix [37] se proposait d'améliorer l'esthétique des matrices comportant des points de suspension. De là vient le nom *nicematrix* alors que le nom *nicetabular* serait plus adapté pour les versions actuelles.

Traditionnellement, quand on utilise l'amsmath, voici comment on compose la matrice triangulaire supérieure ne comportant que des 1 au-dessus de la diagonale :

code

```
$\begin{pmatrix}
  1      & \cdots & \cdots & 1      \\
  0      & \ddots &        & \vdots \\
  \vdots & \ddots & \ddots & \vdots \\
  0      & \cdots & 0      & 1      \\
\end{pmatrix}$
```

résultat

$$\begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Les lignes en pointillées sont discontinues, ce qui n'est pas étonnant vu que l'on n'a placé que des caractères isolés comme on le ferait en typographie au plomb. Pour arriver à avoir des lignes pointillées continues, on est obligé d'utiliser une couche de programmation graphique capable de dessiner des objets au-dessus des caractères. L'extension nicematrix utilise pour cela PGF [38], qui est une sous-couche de Ti*k*Z. Avec l'environnement `pNiceMatrix` de nicematrix et les commandes `\Cdots`, `\Vdots` et `\Ddots` qui y sont définies, on obtient le résultat suivant.

code

```
$\begin{pNiceMatrix}
  1      & \Cdots & \Cdots & 1      \\
  0      & \Ddots &        & \Vdots \\
  \Vdots & \Ddots & \Ddots & \Vdots \\
  0      & \Cdots & 0      & 1      \\
\end{pNiceMatrix}$
```

résultat

$$\begin{pNiceMatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pNiceMatrix}$$

Comme on le voit, la syntaxe est très proche de celle de l'environnement `pmatrix` de l'amsmath.

En fait, certaines commandes sont redondantes (il n'y a pas besoin de deux instructions `\Cdots` pour tracer une même ligne pointillée horizontale : une seule suffit).

code

```
$\begin{pNiceMatrix}
  1      &        & \Cdots & 1      \\
  0      & \Ddots &        & \Vdots \\
  \Vdots & \Ddots &        &        \\
  0      & \Cdots & 0      & 1      \\
\end{pNiceMatrix}$
```

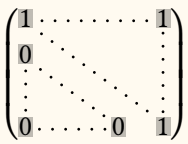
résultat

$$\begin{pNiceMatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pNiceMatrix}$$

Pour arriver à ce résultat, l'extension nicematrix a créé un nœud [39] PGF-Ti*k*Z derrière le contenu de chaque cellule de la matrice, *exception faite des cellules dont le contenu est vide.*

37. L'extension nicematrix est écrite au maximum en expl3, qui est, en quelque sorte, le langage de programmation de LaTeX3 et qui a, d'ores et déjà, été intégré au noyau LaTeX.
38. *Portable Graphics Format* [anglais] : format de graphiques portables.
39. Avec PGF-Ti*k*Z, les nœuds sont créés, par exemple, avec l'opération node dans un chemin.

Sur la figure suivante, on a colorié en gris ces nœuds PGF-Ti*k*Z.



Ces nœuds sont utilisés en interne par nicematrix pour tracer les lignes pointillées continues (entre autres choses), mais l'utilisateur final peut aussi les utiliser directement avec Ti*k*Z. Dans le « `\CodeAfter` » de l'environnement, le nœud correspondant au contenu de la cellule en ligne i et colonne j est simplement noté $i-j$.

code

```
$\begin{pNiceMatrix}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
  \CodeAfter
  \tikz \draw (2-2) circle (2mm) ;
\end{pNiceMatrix}$
```

résultat

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & \textcircled{5} & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

En créant ces nœuds, nicematrix est assez proche de ce que propose la bibliothèque matrix de Ti*k*Z. Il faut néanmoins remarquer que l'espacement entre les rangées d'une matrice de Ti*k*Z n'obéit pas aux mêmes règles que celles qui sont à l'œuvre dans l'environnement `pNiceMatrix` (qui est composé exactement comme l'environnement `pmatrix`, une `pNiceMatrix` étant, en fait, une `pmatrix` en interne).

Voici, à titre d'exemple, une matrice composée avec un environnement `NiceMatrix` de nicematrix puis avec une `\matrix` de Ti*k*Z.

code

```
$\begin{NiceMatrix}
  \frac 12 & 2 \\
  3        & 4 \\
\end{NiceMatrix}$
```

résultat

$$\begin{matrix} \frac{1}{2} & 2 \\ 3 & 4 \end{matrix}$$

code

```
$\vcenter{
  \begin{tikzpicture}
    \matrix (A) [matrix of math nodes]
    {
     \frac 12 & 2 \\
     3        & 4 \\
    } ;
\end{tikzpicture}}$
```

résultat

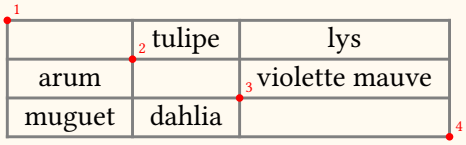
$$\begin{matrix} \frac{1}{2} & 2 \\ 3 & 4 \end{matrix}$$

Même si, à l'origine, l'extension nicematrix avait pour but de composer des matrices mathématiques, elle permet aussi, dans les versions récentes, de composer des tableaux les plus généraux (non mathématiques), comportant, en particulier, des filets.

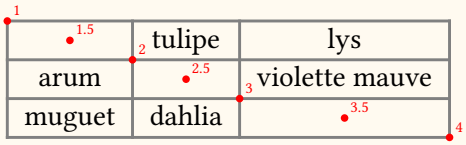
Les nœuds correspondants à la position des filets

Ayant créé des nœuds pour le contenu des cellules, comme dit précédemment, l'auteur de nicematrix (c'est-à-dire moi-même) a eu l'idée de créer des nœuds de type `coordinate` correspondants à la position des filets. Le nœud se trouvant en position i sur la diagonale

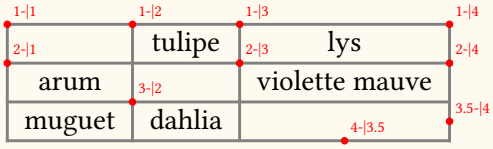
est nommé simplement `i` (PGF autorise pour les nœuds un tel nom composé uniquement de chiffres).



En fait, ce nœud n'est pas tout à fait de type `coordinate`, comme on vient de le dire, car il a une ancre nommée `5` (PGF autorise un tel nom ne comportant que des chiffres pour une ancre). L'ancre `5` du nœud `i`, qui se note `i.5`, se trouve à mi-chemin entre le nœud *i* et et le nœud *i* + 1. On peut donc considérer la figure suivante et imaginer, grâce à cette astuce, qu'il existe des nœuds `1.5`, `2.5` et `3.5`.



Pour avoir l'intersection du filet horizontal numéro *i* et du filet vertical numéro *j*, il suffit d'utiliser le raccourci syntaxique « `-|` » de Ti*k*Z : « *i*`-|`*j* » est le point en ligne *i* et colonne *j*. On donne quelques exemples ci-dessous.



Bien sûr, il ne faut pas confondre une écriture `i-j` qui désigne un nœud rectangulaire (avec ses ancres `north`, `south`, `west`, `east`, etc.) et une écriture `i-|j` qui désigne un point géométrique.

L'utilisation de ces nœuds autorise bien entendu une gamme de possibilités graphiques qui seraient très difficiles, voire impossibles, à obtenir avec les tableaux classiques de LaTeX. En ce qui concerne l'utilisateur final, il peut utiliser ces nœuds avec Ti*k*Z dans le « `\CodeAfter` » des environnements de nicematrix. L'exemple suivant montre comment barrer en croix une case d'un tableau `NiceTabular` (qui est le pendant dans nicematrix de l'environnement classique `tabular`).

code

```
\begin{NiceTabular}{ccc}[hvlines]
  Laurence  & Stéphanie & Noémie     \\
  Alexandra &           & Charlotte  \\
  Cécile    & Eugénie   & Emmanuelle \\
  \CodeAfter
  \tikz
  \draw (2-|2) -- (3-|3) (3-|2) -- (2-|3) ;
\end{NiceTabular}
```

Laurence	Stéphanie	Noémie
Alexandra		Charlotte
Cécile	Eugénie	Emmanuelle

résultat

La clé `hvlines` trace tous les filets horizontaux et verticaux.

L'extension nicematrix utilise en interne ces nœuds pour tracer les filets. Cela permet, par exemple, d'avoir des filets verticaux qui traversent les double-filets horizontaux tracés par `\hline\hline`, sans qu'il y ait besoin d'utiliser l'extension hhline.

code

```
\begin{NiceTabular}{|c|c|} \hline
  Premier & Deuxième \\ \hline\hline
  Paul               \\ \hline
  Marie   & Pauline  \\ \hline
\end{NiceTabular}
```

résultat

Premier	Deuxième
Paul	
Marie	Pauline

Bien entendu, l'auteur est conscient que, dans bien des situations, une présentation de tableau avec uniquement des filets horizontaux telle que proposée par l'extension booktabs de Simon Fear est préférable. Nous présentons ici des outils sans préjuger des choix graphiques des utilisateurs.

Le problème du coloriage des tableaux

Fort de tous ces nœuds, nicematrix peut essayer de faciliter la composition des tableaux.

L'un des problèmes subtils de la composition des tableaux est celui du coloriage. L'outil le plus classique utilisé pour cela est l'extension colortbl de David Carlisle. Beaucoup de personnes chargent cette extension *via* l'extension xcolor avec son option `table`.

L'extension colortbl suit la construction des tableaux faite par LaTeX. Le tableau est donc construit ligne par ligne, les panneaux de couleur étant insérés au moment adéquat, entre les filets et les composantes des cellules. Pour une ligne colorée qui suit un filet horizontal, les panneaux de couleur sont insérés après le filet horizontal. La taille et la position de ces objets sont calculés, comme toujours avec TeX, à la précision du point d'échelle (sp) et le résultat, dans le pdf produit, est mathématiquement irréprochable.

Néanmoins, les lecteurs de pdf qui ont à effectuer la rastérisation des pages (c'est-à-dire la conversion des contenus vectoriels en images pixellisées pour l'affichage sur écran) ont tendance à raisonner selon le « modèle du peintre » (*painting model* en anglais) qui n'est pas celui de la typographique.

En typographie, les différents types (c'est-à-dire les petits blocs de plomb correspondants aux caractères) sont juxtaposés les uns à côté des autres sans pouvoir se chevaucher. TeX procède essentiellement de la même manière pour la construction de ses « boîtes ».

En revanche, le PostScript (dont le pdf est l'héritier) décrit une page selon le « modèle du peintre ». La page est constituée de « marques » qui se superposent opaquement comme le font les coups de pinceau d'un peintre. Cela permet, bien entendu, la réalisation aisée de figures et d'illustrations.

C'est pourquoi certains lecteurs de pdf ont tendance à donner la priorité aux éléments tracés postérieurement aux autres. Un rectangle coloré tracé après un filet horizontal, les deux étant mathématiquement juxtaposés sans aucun recouvrement, aura quand même tendance à masquer le filet à certains niveaux de zoom. C'est le cas en particulier avec `Adobe Reader`.

À l'heure où de nombreux pdf ne sont jamais imprimés mais uniquement visualisés sur écran, on peut légitimement vouloir résoudre ce problème. L'extension nicematrix apporte une solution car, chose que nous n'avions pas encore mentionnée, les nœuds de la forme « `i-|j` » sont en fait accessibles *avant* la construction du tableau (ce n'est pas encore le cas pour les nœuds de la forme « `i-j` ») [40]. Pour y parvenir, nicematrix utilise un mécanisme proposé par pgf (c'est le même mécanisme que celui utilisé par l'extension tikzmark) : les coordonnées des nœuds sont écrites dans le fichier `.aux` pour pouvoir être accessibles, à la

40. En fait, dans les dernières versions de nicematrix, ces nœuds-là sont également accessibles (quand on a utilisé la clé `create-cell-nodes` de la commande `\CodeBefore`).

compilation suivante, avant le début de l'environnement (le fichier aux est toujours, avec LaTeX, exécuté au début de la compilation).

L'utilisateur final peut d'ailleurs, s'il le souhaite, utiliser ces nœuds avant la construction du tableau proprement dit. Une syntaxe spéciale est prévue pour cela par nicematrix, utilisant les mots-clés `\CodeBefore` et `\Body`. On en donne maintenant un exemple.

code

```
\begin{NiceTabular}{ccc}[hvlines,rules/width=0.2pt,baseline=1]
  \CodeBefore
    \tikz
    \fill [red!15] (2-|2.5) -- (2.5-|2) -- (3-|2.5) -- (2.5-|3) -- cycle ;
  \Body
    dalhia  & narcisse & primevère \\
    tulipe  & rose     & muguet    \\
    camélia & iris     & violette
\end{NiceTabular}
```

résultat

dalhia	narcisse	primevère
tulipe	rose	muguet
camélia	iris	violette

L'extension nicematrix utilise en interne ce `\CodeBefore` (si on peut dire) pour colorier les cellules, les rangées et les colonnes avec une syntaxe similaire à celle de colortbl (à condition d'utiliser la clé `colortbl-like`).

code

```
\NewDocumentCommand { \Blue } { } {\columncolor{blue!15}} % syntaxe de xparse
\begin{NiceTabular}[colortbl-like]{>{\Blue}c>{\Blue}cc}
  \toprule \rowcolor{red!15}
  Nom      & Prénom    & Année de naissance \\
  \midrule
  Achard   & Jacques   & 5 juin 1962        \\
  Lefebvre & Mathilde  & 23 mai 1988        \\
  Vanesse  & Stéphanie & 30 octobre 1994    \\
  Dupont   & Chantal   & 15 janvier 1998    \\
  \bottomrule
\end{NiceTabular}
```

résultat

Nom	Prénom	Année de naissance
Achard	Jacques	5 juin 1962
Lefebvre	Mathilde	23 mai 1988
Vanesse	Stéphanie	30 octobre 1994
Dupont	Chantal	15 janvier 1998

Les commandes `\toprule`, `\midrule` et `\bottomrule` sont des commandes de l'extension booktabs. On remarquera en passant que les commandes de coloriage de nicematrix sont compatibles avec ces commandes de booktabs.

Formatage à l'intérieur ou à l'extérieur du tableau

Il y a toujours, pour les langages de description de tableaux, l'alternative entre insérer les instructions de formatage au fur et à mesure des composantes du tableau ou bien les avoir de manière centralisée, avant ou après les données proprement dites.

Sur bien des points, l'extension nicematrix offre à l'utilisateur les deux possibilités.

Par exemple, concernant les coloriages de rangées, de colonnes ou de cellules, on a vu que des commandes à la syntaxe similaire à celles de colortbl sont proposées (sans compter qu'il est possible d'utiliser tout simplement les commandes de colortbl elles-mêmes dans un tableau `NiceTabular`). Pour autant, des commandes de coloriage sont également disponibles dans l'argument de la commande `\CodeBefore` pour spécifier le coloriage de manière centralisée.

Le résultat est bien entendu le même que précédemment.

code

```
\begin{NiceTabular}{ccc}
\CodeBefore
  \columncolor{blue!15}{1,2}
  \rowcolor{red!15}{1}
\Body
\toprule
Nom      & Prénom    & Année de naissance \\
\midrule
Achard   & Jacques   & 5 juin 1962        \\
Lefebvre & Mathilde  & 23 mai 1988        \\
Vanesse  & Stéphanie & 30 octobre 1994    \\
Dupont   & Chantal   & 15 janvier 1998    \\
\bottomrule
\end{NiceTabular}
```

résultat

Nom	Prénom	Année de naissance
Achard	Jacques	5 juin 1962
Lefebvre	Mathilde	23 mai 1988
Vanesse	Stéphanie	30 octobre 1994
Dupont	Chantal	15 janvier 1998

Concernant les filets, il est possible de les spécifier de la manière classique (avec le spécificateur « | » dans le préambule et la commande `\hline`). Il existe néanmoins aussi une clé `hvlines` qui trace tous les filets. Sans compter qu'on peut toujours tracer n'importe quel trait avec Ti*k*Z en utilisant les nœuds créés par nicematrix.

La commande `\Block`

La commande `\Block` est l'outil le plus versatile disponible dans les environnements de nicematrix. Cette commande s'utilise dans une cellule du tableau avec la syntaxe suivante où les deux premiers arguments, entre crochets et chevrons, sont optionnels :

`\Block`[⟨*options*⟩]<⟨*tokens*⟩>{⟨*n*⟩-⟨*p*⟩}{⟨*contenu*⟩}

Les nombres n et p désignent les nombres de lignes et de colonnes du bloc à créer. Si cet argument est laissé vide, c'est un bloc de taille `1-1` qui est créé.

La commande `\Block` sert en premier lieu à fusionner un certain nombre de cellules du tableau en plaçant son contenu au centre du rectangle de cases fusionnées. De ce fait, elle peut avantageusement remplacer les commandes classiques `\multicolumn` et `\multirow` (cette dernière étant proposée par l'extension éponyme multirow).

code

```
$\begin{bNiceArray}{ccc|c}[margin]
  \Block{3-3}<\LARGE>{A} &                &   & 0      \\
                         & \hspace*{1cm} &   & \Vdots \\
                         &                &   & 0      \\ \hline
  0                      & \Cdots         & 0 & 0
\end{bNiceArray}$
```

$$\left[\begin{array}{ccc|c} & & & 0 \\ & A & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array}\right]$$

résultat

Les tokens placés entre chevrons sont insérés avant le passage en mode mathématique (ici, la commande \LARGE n'aurait pas pu être utilisée en mode mathématique).

Le contenu du bloc est placé *après* la composition du tableau en calculant sa position grâce aux nœuds PGF-Ti*k*Z créés par nicematrix. De la sorte, le placement est correct dès lors que l'on a bien spécifié le nombre de rangées logiques du tableau (et le nombre de colonnes, bien entendu).

Cela est à comparer avec le fonctionnement de la commande \multirow qui place son contenu au moment de composer la ligne où apparaît la commande \multirow et qui a donc besoin en argument du nombre de *lignes physiques* à prendre en compte.

Voici d'ailleurs ce que l'on écrirait sans doute pour composer la même matrice avec \multirow. On a été obligé d'ajuster « à la main » le premier argument de \multirow (3.5).

code

```
$\left[
  \begin{array}{ccc|c}
    \multicolumn{3}{c|}{\multirow{3.5}{*}{\LARGE $A$}} & 0 \\
      & \hspace*{1cm} &   & \vdots \\
      &                &   & 0      \\ \hline
    0 & \cdots         & 0 & 0
  \end{array}
\right]$
```

$$\left[\begin{array}{ccc|c} & & & 0 \\ & A & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array}\right]$$

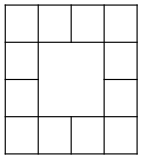
résultat

À l'origine, la commande \Block avait seulement pour but de composer des matrices par blocs comme dans l'exemple précédent. Mais ses usages se sont bien étendus au fur et à mesure des versions successives de nicematrix. On peut même avoir à l'utiliser avec un contenu vide. En effet, les positions des blocs sont gardées en mémoire par nicematrix et influencent le comportement de plusieurs autres commandes. Par exemple, la clé hvlines dont nous avons déjà parlé trace en fait tous les filets, mais exception faite de ceux à l'intérieur des blocs. En un sens, on peut dire qu'elle « respecte » les blocs.

code

```
\begin{NiceTabular}{cccc}[hvlines]
 &  &  &          \\
 & \Block{2-2}{} \\
 &                \\
 &                \\
\end{NiceTabular}
```

résultat



On peut par exemple aussi utiliser la commande `\Block` pour mettre en évidence une partie d'une matrice.

code

```
$\begin{pNiceMatrix}[right-margin]
  1 & 2 & 3 \\
  4 & \Block[draw,rounded-corners]{2-2}{}
  0 & 0     \\
  7 & 0 & 0
\end{pNiceMatrix}$
```

résultat

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 0 & 0 \\ 7 & 0 & 0 \end{pmatrix}$$

L'avenir de nicematrix

Bien qu'elle facilite la composition des tableaux et des matrices, l'extension nicematrix n'est pas l'extension universelle de construction de tableaux dont rêvent sans doute les utilisateurs de LaTeX.

Pour le moment (version 5.17a) les principaux défauts de l'extension nicematrix sont les suivants :

- il n'est pas possible d'utiliser un environnement de nicematrix à l'intérieur d'un autre environnement de nicematrix (mais on peut utiliser un environnement de nicematrix à l'intérieur d'un `tabular` et vice versa, ce qui fait que ce point n'est sans doute pas très gênant) ;
- les tableaux construits par nicematrix ne peuvent pas être coupés par des sauts de page.
- il n'y a pas d'environnement « `NiceTabularX` », c'est-à-dire qu'il n'y a pas de système de calcul automatique de largeur de colonne comme celui proposé par l'extension tabularx de David Carlisle avec ses colonnes de type X.

Ce dernier point sera sans doute corrigé dans une nouvelle version de nicematrix qui proposera un type de colonne X. Arriver à proposer des environnements sécables par saut de page (comme le propose par exemple l'extension longtable) est en revanche un objectif nettement plus difficile à atteindre.

François Pantigny