

L'extension `nicematrix` *

F. Pantigny
fpantigny@wanadoo.fr

16 juillet 2022

Résumé

L'extension LaTeX `nicematrix` fournit de nouveaux environnements similaires aux environnements classiques `{tabular}`, `{array}` et `{matrix}` de `array` et `amsmath` mais avec des fonctionnalités plus étendues.

$$\begin{array}{c} L_1 \\ L_2 \\ \vdots \\ L_n \end{array} \begin{array}{c} C_1 \\ C_2 \cdots \cdots C_n \end{array} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

Produit	dimensions (cm)			Prix
	L	l	h	
petit	3	5.5	1	30
standard	5.5	8	1.5	50.5
premium	8.5	10.5	2	80
extra	8.5	10	1.5	85.5
spécial	12	12	0.5	70

L'extension `nicematrix` est entièrement contenue dans le fichier `nicematrix.sty`. Ce fichier peut être placé dans le répertoire courant ou dans une arborescence `texmf`. Le mieux reste néanmoins d'installer `nicematrix` avec une distribution TeX comme MiKTeX, TeX Live ou MacTeX.

Remarque : Si vous utilisez un service LaTeX via Internet (ex. : Overleaf) vous pouvez télécharger le fichier `nicematrix.sty` dans le dossier de votre projet pour bénéficier de la dernière version de `nicematrix`.¹

Cette extension peut être utilisée avec `xelatex`, `lualatex` et `pdflatex` mais aussi avec le cheminement classique `latex-dvips-ps2pdf` (ou Adobe Distiller). Néanmoins, le fichier `nicematrix-french.tex` de la présente documentation ne peut être compilé qu'avec XeLaTeX.

Cette extension nécessite et charge les extensions `l3keys2e`, `array`, `amsmath` et `pgfcore` ainsi que le module `shapes` de PGF (l'extension `tikz`, qui est une surcouche de PGF, n'est pas chargée). L'utilisateur final n'a qu'à charger l'extension `nicematrix` avec l'instruction habituelle : `\usepackage{nicematrix}`.

L'idée de `nicematrix` est de créer des nœuds PGF derrière les cases et les positions des filets des tableaux créés par `array` et de les utiliser pour développer de nouvelles fonctionnalités. Comme toujours avec PGF, les coordonnées de ces nœuds sont écrites dans le fichier `aux` pour être utilisées à la compilation suivante. C'est pourquoi l'utilisation de `nicematrix` nécessite **plusieurs compilations successives**.²

La plupart des fonctionnalités de `nicematrix` sont accessibles sans avoir à utiliser explicitement PGF ou Tikz (ce dernier n'est d'ailleurs pas chargé par défaut).

Une commande `\NiceMatrixOptions` est fournie pour régler les options (la portée des options fixées par cette commande est le groupe TeX courant : elles sont semi-globales).

*Ce document correspond à la version 6.11 de `nicematrix`, en date du 2022/07/16.

1. La dernière version de `nicematrix.sty` peut être téléchargée sur le serveur SVN de TeXLive : <https://www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/nicematrix/nicematrix.sty>
2. Si vous utilisez Overleaf, Overleaf effectue automatiquement le nombre de compilations nécessaire.

1 Les environnements de cette extension

L'extension `nicematrix` définit les nouveaux environnements suivants :

<code>{NiceTabular}</code>	<code>{NiceArray}</code>	<code>{NiceMatrix}</code>
<code>{NiceTabular*}</code>	<code>{pNiceArray}</code>	<code>{pNiceMatrix}</code>
<code>{NiceTabularX}</code>	<code>{bNiceArray}</code>	<code>{bNiceMatrix}</code>
	<code>{BNiceArray}</code>	<code>{BNiceMatrix}</code>
	<code>{vNiceArray}</code>	<code>{vNiceMatrix}</code>
	<code>{VNiceArray}</code>	<code>{VNiceMatrix}</code>

Les environnements `{NiceArray}`, `{NiceTabular}` et `{NiceTabular*}` sont similaires aux environnements `{array}`, `{tabular}` et `{tabular*}` de l'extension `array` (qui est chargée par `nicematrix`).

Les environnements `{pNiceArray}`, `{bNiceArray}`, etc. n'ont pas d'équivalents dans `array`.

Les environnements `{NiceMatrix}`, `{pNiceMatrix}`, etc. sont similaires aux environnements correspondants de l'`amsmath` (qui est chargée par `nicematrix`) : `{matrix}`, `{pmatrix}`, etc.

L'environnement `{NiceTabularX}` est similaire à l'environnement `{tabularx}` de l'extension éponyme.³

On conseille d'utiliser prioritairement les environnements classiques et de n'utiliser les environnements de `nicematrix` que lorsqu'on utilise les fonctionnalités supplémentaires offertes par ces environnements (cela permet d'économiser la mémoire).

Tous les environnements de l'extension `nicematrix` acceptent, entre crochets, une liste optionnelle de paires de la forme `clé=valeur`. Il doit n'y avoir aucun espace devant le crochet ouvrant (`[`) de cette liste d'options.

2 L'espace vertical entre les rangées

Il est bien connu que certaines rangées des tableaux créés par défaut avec LaTeX sont trop proches l'une de l'autre. On en donne ci-dessous un exemple classique.

```
\begin{pmatrix}
\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{3} & \frac{1}{4} \\
\end{pmatrix}
```

En s'inspirant de l'extension `cellspace` qui traite de ce problème, l'extension `nicematrix` propose deux clés `cell-space-top-limit` et `cell-space-bottom-limit` qui sont similaires aux deux paramètres `\cellspacetoplimit` et `\cellspacebottomlimit` proposés par `cellspace`.

Il existe aussi une clé `cell-space-limits` pour régler simultanément les deux paramètres.

La valeur initiale de ces paramètres est 0 pt pour que les environnements de `nicematrix` aient par défaut le même comportement que ceux de `array` et de l'`amsmath` mais une valeur de 1 pt serait un bon choix. On conseille de régler leurs valeurs avec la commande `\NiceMatrixOptions`.⁴

```
\NiceMatrixOptions{cell-space-limits = 1pt}

\begin{pNiceMatrix}
\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{3} & \frac{1}{4} \\
\end{pNiceMatrix}
```

3. Néanmoins, on peut aussi utiliser directement les colonnes `X` dans l'environnement `{NiceTabular}`, la largeur souhaitée pour le tableau étant spécifiée par la clé `width` : cf. p. 20.

4. On remarquera que ces paramètres s'appliquent aussi aux colonnes de type `S` de `siunitx` alors que `cellspace` n'est pas utilisable avec ces colonnes.

3 La clé baseline

L'extension `nicematrix` propose une option `baseline` pour la position verticale des tableaux. Cette option `baseline` prend comme valeur un entier qui indique le numéro de rangée dont la ligne de base servira de ligne de base pour le tableau.

```
$A = \begin{pNiceMatrix}[baseline=2]
\frac{1}{\sqrt{1+p^2}} & p & 1-p \\
1 & 1 & 1 \\
1 & p & 1+p
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} \frac{1}{\sqrt{1+p^2}} & p & 1-p \\ 1 & 1 & 1 \\ 1 & p & 1+p \end{pmatrix}$$

L'option `baseline` peut aussi prendre les trois valeurs spéciales `t`, `c` et `b`. Ces trois lettres peuvent aussi être utilisées de manière absolue comme pour l'option des environnements `{tabular}` et `{array}` de `array`. La valeur initiale de `baseline` est `c`.

Dans l'exemple suivant, on utilise l'option `t` (synonyme de `baseline=t`) immédiatement après un `\item` de liste. On remarquera que la présence d'un `\hline` initial n'empêche pas l'alignement sur la ligne de base de la première rangée (avec `{tabular}` ou `{array}` de `array`, il faut utiliser `\firsthline`).

```
\begin{enumerate}
\item un item
\smallskip
\item \renewcommand{\arraystretch}{1.2}
$\begin{NiceArray}[t]{lcccccc}
\hline
n & 0 & 1 & 2 & 3 & 4 & 5 \\
u_n & 1 & 2 & 4 & 8 & 16 & 32
\hline
\end{NiceArray}$
\end{enumerate}
```

1. un item
2.

n	0	1	2	3	4	5
u_n	1	2	4	8	16	32

Il est également possible d'utiliser les outils de `booktabs` : `\toprule`, `\bottomrule`, `\midrule`, etc., à condition, bien entendu, d'avoir chargé `booktabs`.

```
\begin{enumerate}
\item an item
\smallskip
\item
$\begin{NiceArray}[t]{lcccccc}
\toprule
n & 0 & 1 & 2 & 3 & 4 & 5 \\
\midrule
u_n & 1 & 2 & 4 & 8 & 16 & 32
\bottomrule
\end{NiceArray}$
\end{enumerate}
```

1. an item
2.

n	0	1	2	3	4	5
u_n	1	2	4	8	16	32

On peut aussi utiliser la clé `baseline` pour aligner une matrice sur un filet horizontal (tracé par `\hline`). On doit pour cela donner la valeur `line- i` où i est le numéro de la rangée qui *suit* ce filet horizontal.

```
\NiceMatrixOptions{cell-space-limits=1pt}

$A=\begin{pNiceArray}{cc|cc}[baseline=line-3]
\dfrac{1}{A} & \dfrac{1}{B} & 0 & 0 \\
\dfrac{1}{C} & \dfrac{1}{D} & 0 & 0 \\
\hline
0 & 0 & A & B \\
0 & 0 & D & D
\end{pNiceArray}$
```

$$A = \left(\begin{array}{cc|cc} \frac{1}{A} & \frac{1}{B} & 0 & 0 \\ \frac{1}{C} & \frac{1}{D} & 0 & 0 \\ \hline 0 & 0 & A & B \\ 0 & 0 & D & D \end{array} \right)$$

4 Les blocs

4.1 Cas général

Dans les environnements de `nicematrix`, on peut utiliser la commande `\Block` pour placer un élément au centre d'un rectangle de cases fusionnées.⁵

La commande `\Block` doit être utilisée dans la case supérieure gauche du bloc avec deux arguments obligatoires.

- Le premier argument est la taille de ce bloc avec la syntaxe i - j où i est le nombre de rangées et j le nombre de colonnes du bloc.
Si cet argument est laissé blanc, la valeur par défaut est 1-1. Si le nombre de rangées n'est pas indiqué, ou bien est égal à $*$, le bloc s'étend jusqu'à la dernière rangée (idem pour les colonnes).
- Le deuxième argument est le contenu du bloc. On peut utiliser `\\` dans ce contenu pour avoir un contenu sur plusieurs lignes. Dans `{NiceTabular}`, `{NiceTabular*}` et `{NiceTabularX}`, le contenu est composé en mode texte tandis que, dans les autres environnements, il est composé en mode mathématique.

Voici un exemple d'utilisation de la commande `\Block` dans une matrice mathématique.

```
\begin{bNiceArray}{cw{c}{1cm}c|c}[margin]
\Block{3-3}{A} & & 0 \\
& & \Vdots \\
& & 0 \\
\hline
0 & \Cdots & 0 & 0
\end{bNiceArray}
```

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

On peut souhaiter agrandir la taille du « A » placé dans le bloc de l'exemple précédent. Comme il est composé en mode mathématique, on ne peut pas directement utiliser une commande comme `\large`, `\Large` ou `\LARGE`. C'est pourquoi une option à mettre entre chevrons est proposée par `\Block` pour spécifier du code LaTeX qui sera inséré *avant* le début du mode mathématique.⁶

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

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

On peut régler le positionnement horizontal du bloc avec l'une des clés `l`, `c` et `r`.

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

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

La commande `\Block` accepte en fait en premier argument optionnel (entre crochets) une liste de couples *clé=valeur*. Les clés sont les suivantes :

- les clés `l`, `c` et `r` régulent le positionnement horizontal comme dit précédemment ;
- la clé `fill` prend en argument une couleur et remplit le bloc avec cette couleur ;

⁵. Les espaces situés après une commande `\Block` sont supprimés.

⁶. Cet argument entre chevrons peut aussi être utilisé pour insérer une commande de fonte comme `\bfseries`, ce qui peut être utile dans le cas où la commande `\\` apparaît dans le contenu du bloc.

- la clé `draw` prend en argument une couleur et trace le cadre avec cette couleur (la valeur par défaut de cette clé est la couleur courante des filets du tableau) ;
- la clé `color` prend en argument une couleur et l’applique au contenu et trace également le cadre avec cette couleur ;
- les clés `hlines`, `vlines` et `hvlines` tracent les filets correspondants dans le bloc⁷ ;
- la clé `line-width` fixe la largeur utilisée pour tracer les filets (n’a d’intérêt que si `draw`, `hvlines`, `hlines` ou `vlines` est utilisée) ;
- la clé `rounded-corners` impose des coins arrondis (pour le cadre dessiné par `draw` et le fond dessiné par `fill`) avec un rayon égal à la valeur de cette clé (la valeur par défaut est 4 pt⁸) ;
- les clés `t` et `b` régulent la ligne de base qui sera donnée au bloc quand celui-ci est constitué de plusieurs lignes (séparées par la commande `\\`) ;
- quand la clé `tikz` est utilisée, le chemin Tikz correspondant au rectangle délimitant le bloc est exécuté avec Tikz⁹ en utilisant comme options la valeur de cette clé `tikz` (qui doit donc être une liste de clés Tikz applicables à un chemin de Tikz). Pour des exemples d’utilisation de cette clé `tikz`, voir p. 47 ;
- la clé `name` donne un nom au nœud Tikz rectangulaire correspondant au bloc ; on peut utiliser ce nom avec Tikz dans le `\CodeAfter` (cf. p. 28) ;
- la clé `respect-arraystretch` évite la remise à 1 de `\arraystretch` en début de bloc (qui a lieu par défaut) ;
- la clé `borders` permet de ne tracer que certaines des bordures du bloc : cette clé prend comme valeur une liste d’éléments parmi les suivants : `left`, `right`, `top` et `bottom` ; on peut en fait, dans la liste qui est la valeur de la clé `borders` mettre une entrée de la forme `tikz={liste}` où `liste` est une liste de couples `clé=valeur` de Tikz spécifiant les caractéristiques graphiques des traits qui seront dessinés (pour un exemple, voir p. 50).

On doit remarquer que, par défaut, les blocs ne créent pas d’espace. Il n’y a exception que pour les blocs mono-rangée et les blocs mono-colonne comme expliqué plus loin.

Dans l’exemple suivant, on a dû élargir à la main les colonnes 2 et 3 (avec la construction `wc{...}` de `array`).

```
\begin{NiceTabular}{cwc{2cm}wc{3cm}c}
rose      & tulipe & marguerite & dahlia \\
violette
& \Block[draw=red,fill={RGB}{204,204,255},rounded-corners]{2-2}
&          {\LARGE De très jolies fleurs}
& & souci \\
pervenche & & lys \\
arum      & iris  & jacinthe  & muguet
\end{NiceTabular}
```

rose	tulipe	marguerite	dahlia
violette	De très jolies fleurs		souci
pervenche			lys
arum	iris	jacinthe	muguet

4.2 Les blocs mono-colonne

Les blocs mono-colonne ont un comportement spécial.

7. Néanmoins, les filets ne sont pas tracés dans les sous-blocs du bloc, conformément à l’esprit de `nicematrix` : les filets ne sont pas tracés dans les blocs (cf. section 5 p. 7).

8. Cette valeur par défaut est la valeur initiale des *rounded corners* de Tikz.

9. Tikz doit être chargé préalablement (par défaut, `nicematrix` ne charge que PGF), faute de quoi, une erreur sera levée.

- La largeur naturelle du contenu de ces blocs est prise en compte pour la largeur de la colonne courante.
Dans les colonnes à largeur fixée (colonnes `w{...}{...}`, `p{...}`, `b{...}`, `m{...}` et `X`), le contenu du bloc est mis en forme comme un paragraphe de cette largeur.
- La spécification d'alignement horizontal donnée par le type de colonne (`c`, `r` ou `l`) est prise en compte pour le bloc (mais le bloc peut avoir sa propre spécification d'alignement horizontal).
- Les spécifications de fontes imposées à une colonne via la construction `>{...}` dans le préambule du tableau sont prises en compte pour les blocs mono-colonne de cette colonne (ce comportement est assez naturel).

<code>\begin{NiceTabular}{@{>{\color{blue}}lr@{}} \hline</code>	
<code>\Block{2-1}{Pierre} & 12 \\</code>	Pierre 12
<code>& 13 \\ \hline</code>	13
Jacques & 8 \\ \hline	Jacques 8
<code>\Block{3-1}{Stéphanie} & 18 \\</code>	18
<code>& 17 \\</code>	Stéphanie 17
<code>& 15 \\ \hline</code>	15
Amélie & 20 \\ \hline	Amélie 20
Henri & 14 \\ \hline	Henri 14
<code>\Block{2-1}{Estelle} & 15 \\</code>	15
<code>& 19 \\ \hline</code>	Estelle 19
<code>\end{NiceTabular}</code>	

4.3 Les blocs mono-rangée

Pour les blocs mono-rangée, la hauteur (*height*) et la profondeur (*depth*) naturelles sont prises en compte pour la hauteur et la largeur de la rangée en cours (comme le fait la commande standard `\multicolumn` de LaTeX).

4.4 Les blocs mono-case

Les blocs mono-case héritent des caractéristiques des blocs mono-colonne et des blocs mono-rangée.

On pourrait penser que des blocs d'une seule case n'ont aucune utilité mais, en fait, il y a plusieurs situations où leur utilisation peut présenter des avantages.

- Un bloc mono-case permet d'utiliser la commande `\\` pour composer le bloc sur plusieurs lignes.
- On peut utiliser l'option d'alignement horizontal du bloc pour déroger à la consigne générale donnée dans le préambule pour cette colonne.
- On peut tracer un cadre autour du bloc avec la clé `draw` de la commande `\Block` ou colorier le fond avec des bords arrondis avec les clés `fill` et `rounded-corners`.¹⁰
- On peut tracer une ou plusieurs bordures de la case avec la clé `borders`.

<code>\begin{NiceTabular}{cc}</code>	
<code>\toprule</code>	
Écrivain	année de
<code>& \Block[1]{\color{violet}naissance} \\</code>	naissance
<code>\midrule</code>	
Hugo & 1802 \\	Hugo 1802
Balzac & 1799 \\	Balzac 1799
<code>\bottomrule</code>	
<code>\end{NiceTabular}</code>	

On rappelle que si le premier argument obligatoire de `\Block` est laissé blanc, alors le bloc est mono-case¹¹.

10. Pour colorier simplement le fond d'une case, il n'y a pas besoin d'utiliser un bloc mono-case : on peut utiliser la commande `\cellcolor` si la clé `colortbl-like` est utilisée : voir p. 17.

11. On peut considérer que la valeur par défaut de ce premier argument obligatoire est `1-1`.

4.5 Positionnement horizontal du contenu des blocs

Par défaut, le positionnement horizontal des contenus des blocs est calculé sur le *contenu* des colonnes impliquées. De ce fait, dans l'exemple suivant, l'en-tête « Premier groupe » est correctement centré même si un espacement des colonnes a été demandé par une instruction comme `!\qquad` dans le préambule (ce n'est pas le cas avec `\multicolumn`).

```
\begin{NiceTabular}{@{}c!\qquad ccc!\qquad ccc{}}
\toprule
Rang & \Block{1-3}{Premier groupe} & & & \Block{1-3}{Deuxième groupe} & \\\
& 1A & 1B & 1C & 2A & 2B & 2C & \\\
\midrule
1 & 0.657 & 0.913 & 0.733 & 0.830 & 0.387 & 0.893\\
2 & 0.343 & 0.537 & 0.655 & 0.690 & 0.471 & 0.333\\
3 & 0.783 & 0.885 & 0.015 & 0.306 & 0.643 & 0.263\\
4 & 0.161 & 0.708 & 0.386 & 0.257 & 0.074 & 0.336\\
\bottomrule
\end{NiceTabular}
```

Rang	Premier groupe			Deuxième groupe		
	1A	1B	1C	2A	2B	2C
1	0.657	0.913	0.733	0.830	0.387	0.893
2	0.343	0.537	0.655	0.690	0.471	0.333
3	0.783	0.885	0.015	0.306	0.643	0.263
4	0.161	0.708	0.386	0.257	0.074	0.336

Pour avoir un positionnement horizontal du contenu du bloc qui s'appuie sur les limites des colonnes du tableau LaTeX (et non sur le contenu de ces colonnes), il faut utiliser les clés `L`, `R` et `C` de la commande `\Block`.

5 Les filets horizontaux et verticaux

Les techniques habituelles pour tracer des filets peuvent être utilisées dans les environnements de `nicematrix`, à l'exception de `\vline`. Il y a néanmoins quelques petites différences de comportement avec les environnements classiques.

5.1 Quelques différences avec les environnements classiques

5.1.1 Les filets verticaux

Dans les environnements de `nicematrix`, les filets verticaux spécifiés par `|` dans le préambule des environnements ne sont jamais coupés, même en cas de ligne incomplète ou de double filet horizontal spécifié par `\hline\hline` (il n'y a pas besoin d'utiliser `hhline`).

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

Premier	Deuxième
Paul	
Marie	Pauline

En revanche, les filets verticaux ne sont pas tracés à l'intérieur des blocs (créés par `\Block` : cf. p. 4) ni dans les coins (dont la création est demandée par la clé `corners` : cf. p. 10).

Si vous utilisez `booktabs` (qui fournit `\toprule`, `\midrule`, `\bottomrule`, etc.) et que vous tenez absolument à mettre des filets verticaux (ce qui est contraire à l'esprit à `booktabs`), vous constaterez que les filets tracés par `nicematrix` sont compatibles avec `booktabs`.

```

$\begin{NiceArray}{|ccc|} \toprule
a & b & c & d \\ \midrule
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\ \bottomrule
\end{NiceArray}$

```

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
1	2	3	4
1	2	3	4

Il reste néanmoins possible de définir un spécificateur, nommé par exemple `I`, pour tracer des filets verticaux avec le comportement standard de `array` :

```
\newcolumnntype{I}{!{\vrule}}
```

5.1.2 La commande `\cline`

Les traits verticaux et horizontaux que l'on insère avec `\hline` et le spécificateur de colonne « `|` » de `array` rendent le tableau plus large ou plus long d'une quantité égale à la largeur du trait (avec `array` et aussi avec `nicematrix`).

Pour des raisons historiques, il n'en est pas de même pour la commande `\cline`, comme on peut le voir avec l'exemple suivant.

```

\setlength{\arrayrulewidth}{2pt}
\begin{tabular}{cccc} \hline
A&B&C&D \\ \cline{2-2}
A&B&C&D \\ \hline
\end{tabular}

```

A	B	C	D
A	B	C	D

Dans les environnements de `nicematrix`, cette situation est corrigée (il est néanmoins possible de revenir au comportement par défaut de `\cline` avec la clé `standard-cline`).

```

\setlength{\arrayrulewidth}{2pt}
\begin{NiceTabular}{cccc} \hline
A&B&C&D \\ \cline{2}
A&B&C&D \\ \hline
\end{NiceTabular}

```

A	B	C	D
A	B	C	D

Dans les environnements de `nicematrix`, une instruction `\cline{i}` est équivalente à `\cline{i-i}`.

5.2 L'épaisseur et la couleur des filets

Les environnements de `nicematrix` proposent une clé `rules/width` pour fixer la largeur (on devrait plutôt dire l'épaisseur) des filets dans l'environnement. En fait, cette clé ne fait que fixer la valeur du paramètre dimensionnel `\arrayrulewidth`.

On sait que `colortbl` propose la commande `\arrayrulecolor` pour spécifier la couleur de ces filets.

Avec `nicematrix`, il est possible de spécifier une couleur même si `colortbl` n'est pas chargé. Par souci de compatibilité, la commande est nommée également `\arrayrulecolor`. Les environnements de `nicematrix` proposent également une clé `rules/color` qui permet de fixer cette couleur pour l'environnement en question. Cette clé fixe localement la couleur des filets (alors que la commande `\arrayrulecolor` agit globalement).

```

\begin{NiceTabular}{|ccc|}[rules/color=gray]{0.9},rules/width=1pt]
\hline
rose & tulipe & lys \\
arum & iris & violette \\
muguet & dahlia & souci \\
\hline
\end{NiceTabular}

```

rose	tulipe	lys
arum	iris	violette
muguet	dahlia	souci

5.3 Les outils de `nicematrix` pour tracer des filets

Les outils proposés par `nicematrix` pour tracer des filets sont les suivants :

- les clés `hlines`, `vlines`, `hvlines` et `hvlines-except-borders` ;

- le spécificateur « | » dans le préambule (pour les environnements à préambule) ;
- la commande `\Hline`.

Ces outils ont en commun de ne pas tracer les filets dans les blocs ni dans les coins vides (quand la clé `corners` est utilisée).

- Les blocs en question sont :
 - ceux créés par la commande `\Block`¹² de `nicematrix` présentée p. 4 ;
 - ceux délimités implicitement par des lignes en pointillés continues, créées par `\Cdots`, `\Vdots`, etc. : cf. p. 23.
- Les coins sont créés par la clé `corners` détaillée un peu plus loin : cf. p. 10.

En particulier, cette remarque explique la différence entre la commande standard `\hline` et la commande `\Hline` proposée par `nicematrix`.

5.3.1 Les clés `hlines` et `vlines`

Les clés `hlines` et `vlines` (qui, bien sûr, tracent des filets horizontaux et verticaux) prennent comme valeur une liste de numéros qui sont les numéros des filets¹³ à tracer. Si aucune valeur n'est donnée, tous les filets sont tracés.

En fait, pour les environnements avec délimiteurs (comme `{pNiceMatrix}` ou `{bNiceArray}`), la clé `vlines` ne trace pas les filets extérieurs (ce qui est le comportement certainement attendu).

```

$\begin{pNiceMatrix}[vlines,rules/width=0.2pt]
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
\end{pNiceMatrix}$

```

$$\left(\begin{array}{c|c|c|c|c|c} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{array} \right)$$

5.3.2 Les clés `hvlines` et `hvlines-except-borders`

La clé `hvlines`, qui ne prend pas de valeur, est la conjonction des clés `hlines` et `vlines`.

```

\begin{NiceTabular}{cccc}[hvlines,rules/color=blue,rules/width=1pt]
rose      & tulipe & marguerite & dahlia \\
violette  & \Block[draw=red]{2-2}{\LARGE fleurs} & & souci \\
pervenche & & lys \\
arum      & iris & jacinthe & muguet \\
\end{NiceTabular}

```

rose	tulipe	marguerite	dahlia
violette	fleurs		souci
pervenche			lys
arum	iris	jacinthe	muguet

La clé `hvlines-except-borders` est similaire à la clé `hvlines` mais ne trace pas les filets sur les bords horizontaux et verticaux du tableau.

12. Et aussi la commande `\multicolumn` même s'il est recommandé d'utiliser plutôt `\Block` quand on utilise l'extension `nicematrix`.

13. Il est également possible de mettre dans la liste des intervalles de numéros de la forme $i-j$.

5.3.3 Les coins (vides)

Les quatre coins d'un tableau seront notés NW, SW, NE et SE (*north west*, *south west*, *north east* et *south east* en anglais).

Pour chacun de ces coins, on appellera *coin vide* (ou tout simplement *coin*) la réunion de toutes les cases vides situées dans au moins un rectangle entièrement constitué de cases vides partant de ce coin.¹⁴

On peut néanmoins imposer à une case sans contenu d'être considérée comme non vide par `nicematrix` avec la commande `\NotEmpty`.

			A		
		A	A	A	
			A		
		A	A	A	A
A	A	A	A	A	A
A	A	A	A	A	A
	A	A	A		
		B	A		
			A		

Dans l'exemple ci-contre (où B est au centre d'un bloc de taille 2×2), on a colorié en bleu clair les quatre coins (vides) du tableau.

Quand la clé **corners** est utilisée, **nicematrix** calcule les coins vides et ces coins sont alors pris en compte par les outils de tracés de filets (les filets ne seront pas tracés dans ces coins vides).

```
\NiceMatrixOptions{cell-space-top-limit=3pt}
\begin{NiceTabular}{*{6}{c}}[corners,hvlines]
& & & & A & \\
& & A & A & A & \\
& & & A & & \\
& & A & A & A & A & \\
A & A & A & A & A & A & A & \\
A & A & A & A & A & A & A & \\
& A & A & A & & & & \\
& \Block{2-2}{B} & & A & & & & \\
& & & A & & & & \\
\end{NiceTabular}
```

				A	
		A	A	A	
			A		
		A	A	A	A
A	A	A	A	A	A
A	A	A	A	A	A
	A	A	A		
	B		A		
			A		

On peut aussi donner comme valeur à la clé **corners** une liste de coins à prendre en considération (les coins sont notés NW, SW, NE et SE et doivent être séparés par des virgules).

```
\NiceMatrixOptions{cell-space-top-limit=3pt}
\begin{NiceTabular}{*{6}{c}}[corners=NE,hvlines]
1\\
1&1\\
1&2&1\\
1&3&3&1\\
1&4&6&4&1\\
&&&&&1
\end{NiceTabular}
```

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
					1

▷ Les coins sont également pris en compte par les outils de coloriage dans le `\CodeBefore`. Ces outils ne colorient pas les cases qui sont dans les coins (cf. p. 14).

14. Pour être complet, on doit préciser que toute case située dans un bloc (même si elle est vide) n'est pas prise en compte pour la détermination des coins. Ce comportement est naturel. La définition précise de ce qui est considéré comme une « case vide » est donnée plus loin (cf. p. 45).

5.4 La commande `\diagbox`

La commande `\diagbox` (inspirée par l’extension `diagbox`) permet, quand elle est utilisée dans une case, de couper cette case selon une diagonale descendante.

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\diagbox{x}{y} & e & a & b & c \\
e & e & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

$x \backslash y$	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Cette commande `\diagbox` peut aussi être utilisée dans un `\Block`.

5.5 Commandes pour filets personnalisés

Il est possible de définir des commandes et des lettres pour des filets personnalisés avec la clé `custom-line`, utilisable dans `\NiceMatrixOptions` ou bien dans un environnement. Cette clé prend en argument une liste de paires de la forme `clé=valeur`. Il y a d’abord deux clés pour spécifier les outils qui permettront d’utiliser ce nouveau type de filet.

- la clé `command` indique le nom (sans la contre-oblique) d’une commande qui sera créée par `nicematrix` et que l’utilisateur pourra utiliser pour tracer des filets horizontaux (de manière similaire à `\hline`);
- **Nouveau 6.11** la clé `ccommand` indique le nom (sans la contre-oblique) d’une commande qui sera créée par `nicematrix` et que l’utilisateur pourra utiliser pour tracer des filets horizontaux partiels (de manière similaire à `\cline`, d’où le nom `ccommand`) : l’argument de cette commande est une liste d’intervalles de colonnes spécifiés par la syntaxe i ou $i-j$; ¹⁵
- la clé `letter` prend en argument une lettre ¹⁶ qui pourra être utilisée par l’utilisateur dans le préambule d’un environnement à préambule (comme `\NiceTabular`) pour spécifier un filet vertical.

Pour la description du filet elle-même, il y a trois possibilités.

- *Première possibilité*

Il est possible de spécifier des filets multiples, colorés avec une couleur entre les filets (comme on peut le faire avec `colortbl` par exemple).

- la clé `multiplicity` indique le nombre de traits successifs qui seront tracés : par exemple, une valeur de 2 va créer des filets doubles comme créés en standard par `\hline\hline` ou bien `||` dans le préambule d’un environnement;
- la clé `color` fixe la couleur des filets;
- la clé `sep-color` fixe la couleur entre deux filets consécutifs (n’a d’intérêt que dans le cas où la clé `multiplicity` est utilisée).

Ce système permet en particulier de définir des commandes pour tracer des filets avec une couleur spécifique (et ces filets respecteront les blocs et les coins comme les autres filets de `nicematrix`).

```
\begin{NiceTabular}{lclcl}[custom-line = {letter=I, color=blue}]
\hline
& \Block{1-3}{dimensions} \\
& L & l & h \\
\hline
```

15. Il est recommandé de n’utiliser ces commandes qu’une seule fois par ligne car chaque utilisation crée un espace vertical entre les rangées correspondant à la largeur totale du trait qui sera tracé.

16. Les lettres suivantes ne sont pas autorisées : `lcrpmbVX|()[]!@<>`

```

Produit A & 3 & 1 & 2 \\
Produit B & 1 & 3 & 4 \\
Produit C & 5 & 4 & 1 \\
\hline
\end{NiceTabular}

```

	dimensions		
	L	l	H
Produit A	3	1	2
Produit B	1	3	4
Produit C	5	4	1

— *Deuxième possibilité*

On peut utiliser la clé `tikz` (si Tikz est chargé, `nicematrix` ne chargeant par défaut que PGF). Dans ce cas-là, le filet est tracé directement avec Tikz en utilisant comme paramètres la valeur de la clé `tikz` qui doit être une liste de couples *clé=valeur* applicables à un chemin Tikz.

Par défaut aucune réservation de place n'est faite pour le filet qui sera tracé avec Tikz. On peut demander une réservation (horizontale pour un filet vertical et verticale pour un filet horizontal) avec la clé `total-width` qui est donc en quelque sorte la largeur du filet qui sera tracé (cette largeur n'est *pas* calculée à partir des caractéristiques fournies par la clé `tikz`).

Voici ce que l'on obtient avec la clé `dotted` de Tikz.

```

\NiceMatrixOptions
{
  custom-line =
  {
    letter = I ,
    tikz = dotted ,
    total-width = \pgflinewidth
  }
}

\begin{NiceTabular}{cIcIc}
un & deux & trois \\
quatre & cinq & six \\
sept & huit & neuf
\end{NiceTabular}

```

un	deux	trois
quatre	cinq	six
sept	huit	neuf

— *Troisième possibilité : la clé `dotted`*

Comme on le voit dans l'exemple précédent, les pointillés tracés par la clé `dotted` de Tikz ne sont pas ronds. C'est pourquoi l'extension `nicematrix` propose dans la clé `custom-line` une clé `dotted` qui va tracer des pointillés ronds. La valeur initiale de la clé `total-width` est, dans ce cas-là, égale au diamètre des points (l'utilisateur peut quand même utiliser la clé `total-width` pour en changer la valeur). Ces pointillés ronds sont aussi utilisés par `nicematrix` pour des lignes en pointillés continues créées entre deux composantes de la matrice par `\Cdots`, `\Vdots`, etc. (voir p. 23).

L'extension `nicematrix` prédéfinit en fait les commandes `\hdottedline` et `\cdottedline` et la lettre « : » pour ces filets en pointillés.¹⁷

17. Néanmoins, l'utilisateur peut écraser ces définitions de `\hdottedline`, `\cdottedline` et de « : » avec `custom-line` s'il le souhaite (par exemple pour les remplacer par des lignes en tirets).

```
% présent dans nicematrix.sty
\NiceMatrixOptions
{
  custom-line =
  {
    letter = : ,
    command = hdottedline ,
    ccommand = cdottedline ,
    dotted
  }
}
```

Dans les environnements de `nicematrix`, il est donc possible d'utiliser les commandes `\hdottedline` et `\cdottedline` pour tracer des filets horizontaux en pointillés.

```
\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
\hdottedline
6 & 7 & 8 & 9 & 10 \\
  emphase\cdottedline{1,4-5}
11 & 12 & 13 & 14 & 15
\end{pNiceMatrix}
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ \hdottedline 6 & 7 & 8 & 9 & 10 \\ \cdottedline 11 & 12 & 13 & 14 & 15 \end{pmatrix}$$

Dans les environnements avec un préambule explicite (comme `{NiceTabular}`, `{NiceArray}`, etc.), il est possible de dessiner un trait vertical en pointillés avec le spécificateur « : ».

```
\begin{pNiceArray}{cccc:c}
1 & 2 & 3 & 4 & 5 \\
6 & 7 & 8 & 9 & 10 \\
11 & 12 & 13 & 14 & 15
\end{pNiceArray}
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & : 5 \\ 6 & 7 & 8 & 9 & : 10 \\ 11 & 12 & 13 & 14 & : 15 \end{pmatrix}$$

6 Les couleurs des rangées et des colonnes

6.1 Utilisation de `colortbl`

Rappelons que l'extension `colortbl` peut être chargée directement par `\usepackage{colortbl}` ou en chargeant l'extension `xcolor` avec l'option `table` : `\usepackage[table]{xcolor}`.

Comme l'extension `nicematrix` est fondée sur `array`, il est possible d'utiliser `colortbl` avec `nicematrix`.

Il y a néanmoins deux inconvénients :

- L'extension `colortbl` patche `array`, ce qui entraîne des incompatibilités (par exemple avec la commande `\hdotsfor`).
- L'extension `colortbl` construit le tableau ligne par ligne, en alternant rectangles colorés, filets et contenu des cases. Le PDF résultant déroute certains lecteurs de PDF et on a parfois des artefacts d'affichage.
 - Certains filets semblent disparaître. Ce phénomène est dû au fait que les lecteurs de PDF donnent souvent la priorité aux éléments graphiques qui ont été tracés postérieurement (conformément à l'esprit du « modèle du peintre » de PostScript et PDF). De ce point de vue, MuPDF (qui est utilisé par exemple par SumatraPDF) donne de meilleurs résultats que Adobe Reader.
 - Une fine ligne blanche semble apparaître entre deux cases de même couleur. Ce phénomène se produit quand chaque case est coloriée avec sa propre instruction `fill` (opérateur `fill` de PostScript noté `f` en PDF). C'est le cas avec `colortbl` avec lequel chaque case est coloriée individuellement, même si on utilise `\columncolor` ou `\rowcolor`.

Concernant ce phénomène, Adobe Reader donne de meilleurs résultats que MuPDF.

L'extension `nicematrix` propose des outils qui permettent d'éviter ces inconvénients.

6.2 Les outils de nicematrix dans le `\CodeBefore`

L'extension `nicematrix` propose des outils (indépendants de `colortbl`) pour tracer d'abord les rectangles colorés, puis le contenu des cases et les filets. Cette manière de faire est plus dans l'esprit du « modèle du peintre » des formats PostScript et PDF et convient donc mieux aux lecteurs de PDF. L'inconvénient est qu'elle nécessite plusieurs compilations successives.¹⁸

L'extension `nicematrix` fournit une clé `code-before` pour du code qui sera exécuté avant le tracé du tableau.

Une syntaxe alternative est proposée : on peut placer le contenu de ce `code-before` entre les mots-clés `\CodeBefore` et `\Body` juste au début de l'environnement.

```
\begin{pNiceArray}{preamble}
\CodeBefore
  instructions du code-before
\Body
  contenu de l'environnement
\end{pNiceArray}
```

De nouvelles commandes sont disponibles dans ce `\CodeBefore` : `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors`, `\rowlistcolors`, `\chessboardcolors` et `\arraycolor`.¹⁹

Toutes ces commandes acceptent un argument optionnel (entre crochets et en première position) qui est le modèle colorimétrique pour la spécification des couleurs.

Ces commandes ne colorient pas les cases qui se trouvent dans les « coins » si la clé `corners` a été utilisée. La description de cette clé a été faite p. 10.

- La commande `\cellcolor` tient son nom de la commande `\cellcolor` de `colortbl`. Elle prend en arguments obligatoires une couleur et une liste de cases sous le format $i-j$ où i est le numéro de ligne et j le numéro de colonne.

```
\begin{NiceTabular}{|c|c|c|}
\CodeBefore
  \cellcolor[HTML]{FFFF88}{3-1,2-2,1-3}
\Body
\hline
a & b & c \\ \hline
e & f & g \\ \hline
h & i & j \\ \hline
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

- La commande `\rectanglecolor` prend trois arguments obligatoires. Le premier est la couleur, les deux suivants fournissent la case en haut à gauche et la case en bas à droite du rectangle.

```
\begin{NiceTabular}{|c|c|c|}
\CodeBefore
  \rectanglecolor{blue!15}{2-2}{3-3}
\Body
\hline
a & b & c \\ \hline
e & f & g \\ \hline
h & i & j \\ \hline
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

- La commande `\arraycolor` prend en argument obligatoire une couleur et colorie tout le tableau (sauf les éventuelles rangées et colonnes extérieures : cf. p. 21) avec cette couleur. Ce n'est qu'un cas particulier de la commande `\rectanglecolor`.

¹⁸. Si vous utilisez Overleaf, Overleaf effectue automatiquement le nombre de compilations nécessaire.

¹⁹. On pourra remarquer que, dans le `\CodeBefore`, des nœuds PGF/Tikz de la forme $(i-j)$ correspondant à la position des filets éventuels sont également accessibles : cf. p. 42.

- La commande `\chessboardcolors` prend en arguments obligatoires deux couleurs et colorie les cases en quinconces avec les deux couleurs.

```

 $\begin{pNiceMatrix}[r,margin]
\CodeBefore
\chessboardcolors{red!15}{blue!15}
\Body
1 & -1 & 1 \\
-1 & 1 & -1 \\
1 & -1 & 1
\end{pNiceMatrix}$ 

```

$$\begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}$$

On a utilisé la clé `r` qui impose que toutes les colonnes soient alignées à droite (cf. p. 36).

- La commande `\rowcolor` doit son nom à la commande `\rowcolor` de `colortbl`. Son premier argument obligatoire est la couleur et le deuxième est une liste de numéros de rangées ou bien d'intervalles de rangées sous la forme *a-b* (un intervalle de la forme *a-* représente toutes les rangées à partir de la rangée *a*).

```

 $\begin{NiceArray}{lll}[hvlines]
\CodeBefore
\rowcolor{red!15}{1,3-5,8-}
\Body
a_1 & b_1 & c_1 \\
a_2 & b_2 & c_2 \\
a_3 & b_3 & c_3 \\
a_4 & b_4 & c_4 \\
a_5 & b_5 & c_5 \\
a_6 & b_6 & c_6 \\
a_7 & b_7 & c_7 \\
a_8 & b_8 & c_8 \\
a_9 & b_9 & c_9 \\
a_{10} & b_{10} & c_{10}
\end{NiceArray}$ 

```

a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_4	c_4
a_5	b_5	c_5
a_6	b_6	c_6
a_7	b_7	c_7
a_8	b_8	c_8
a_9	b_9	c_9
a_{10}	b_{10}	c_{10}

- La commande `\columncolor` doit son nom à la commande `\columncolor` de `colortbl`. Sa syntaxe est similaire à celle de `\rowcolor`.
- La commande `\rowcolors` (avec un *s*) doit son nom à la commande `\rowcolors` de `xcolor`²⁰. Le *s* rappelle qu'il y a deux couleurs. Elle colorie alternativement les rangées avec les deux couleurs à partir de la rangée dont le numéro est donné en premier argument (obligatoire), comme le fait la commande `\rowcolors` de `xcolor`.

En fait, le premier argument (obligatoire) peut, plus généralement, contenir une liste d'intervalles correspondant à l'ensemble des rangées sur lesquelles portera l'effet de `\rowcolors` (un intervalle de la forme *i* désigne en fait l'intervalle constitué de toutes les rangées du tableau à partir de la rangée *i*).

La commande `\rowcolors` accepte une liste de couples *clé=valeur* comme argument optionnel en dernière position (l'argument optionnel en première position correspond à l'espace colorimétrique). Les clés disponibles sont `cols`, `restart` et `respect-blocks`.

- La clé `cols` décrit un ensemble de colonnes sur lesquelles portera l'effet de `\rowcolors`. Cet ensemble de colonnes est une liste d'intervalles de la forme *i-j* (où *i* et *j* peuvent être remplacés par `*`).
- Avec la clé `restart`, chacun des intervalles de rangées spécifié par le premier argument de `\rowcolors` recommence avec la même couleur.²¹

20. La commande `\rowcolors` de `xcolor` est disponible quand `xcolor` est chargé avec l'option `table`. Cette option charge également l'extension `colortbl`.

21. Autrement, la couleur d'une rangée ne dépend que de la parité de son numéro absolu.

- Avec la clé **respect-blocks**, qui est de type booléen, les « rangées » colorées alternativement peuvent s'étendre sur plusieurs rangées réelles du tableau pour englober les blocs (créés par la commande `\Block` : cf. p. 4).

```
\begin{NiceTabular}{clr}[hvlines]
\CodeBefore
  \rowcolors[gray]{2}{0.8}{}[cols=2-3,restart]
\Body
\Block{1-*}{Résultats} \\
\Block{2-1}{A}& Pierre & 12 \\
& Jacques & 8 \\
\Block{4-1}{B}& Stéphanie & 18 \\
& Amélie & 20 \\
& Henri & 14 \\
& Estelle & 15
\end{NiceTabular}
```

Résultats		
A	Pierre	12
	Jacques	8
B	Stéphanie	18
	Amélie	20
	Henri	14
	Estelle	15

```
\begin{NiceTabular}{lr}[hvlines]
\CodeBefore
  \rowcolors{1}{blue!10}{}[respect-blocks]
\Body
\Block{2-1}{Pierre} & 12 \\
& 13 \\
Jacques & 8 \\
\Block{3-1}{Stéphanie} & 18 \\
& 17 \\
& 15 \\
Amélie & 20 \\
Henri & 14 \\
\Block{2-1}{Estelle} & 15 \\
& 19
\end{NiceTabular}
```

Pierre	12
	13
Jacques	8
Stéphanie	18
	17
	15
Amélie	20
Henri	14
Estelle	15
	19

- L'extension `nicematrix` propose aussi une commande `\rowlistcolors`. Cette commande généralise la commande `\rowcolors` : au lieu de prendre deux arguments successifs pour les couleurs, elle prend un seul argument qui est une liste de couleurs séparées par des virgules. Dans cette liste, le symbole `=` représente une couleur identique à la précédente.

```
\begin{NiceTabular}{c}
\CodeBefore
  \rowlistcolors{1}{red!15,blue!15,green!15}
\Body
Mathilde \\
Pierre \\
Paul \\
Amélie \\
Jacques \\
Antoine \\
Stéphanie \\
\end{NiceTabular}
```

Mathilde
Pierre
Paul
Amélie
Jacques
Antoine
Stéphanie

On rappelle que toutes les commandes de coloriage que l'on vient de décrire ne colorient pas les cases qui sont dans les « coins ». Dans l'exemple suivant, on utilise la clé **corners** pour demander de considérer le coin *north east* (NE).


```

\begin{NiceTabular}{cccccc}[corners=NE,margin,hvlines,first-row,first-col]
\CodeBefore
  \rowlistcolors{1}{blue!15, }
\Body
  & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 1 & & & & & & \\
1 & 1 & 1 & & & & & \\
2 & 1 & 2 & 1 & & & & \\
3 & 1 & 3 & 3 & 1 & & & \\
4 & 1 & 4 & 6 & 4 & 1 & & \\
5 & 1 & 5 & 10 & 10 & 5 & 1 & \\
6 & 1 & 6 & 15 & 20 & 15 & 6 & 1 \\
\end{NiceTabular}

```

	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

On remarquera que ces commandes sont compatibles avec les commandes de `booktabs` (`\toprule`, `\midrule`, `\bottomrule`, etc). Néanmoins, l'extension `booktabs` n'est *pas* chargée par `nicematrix`.

```

\begin{NiceTabular}{lSSSS}
\CodeBefore
  \rowcolor{red!15}{1-2}
  \rowcolors{3}{blue!15}{}
\Body
\toprule
\Block{2-1}{Produit} &
\Block{1-3}{dimensions (cm)} & & &
\Block{2-1}{\rotate{Prix}} \\
\cmidrule{rl}{2-4}
& L & l & h & \\
\midrule
petit & 3 & 5.5 & 1 & 30 \\
moyen & 5.5 & 8 & 1.5 & 50.5 \\
premium & 8.5 & 10.5 & 2 & 80 \\
extra & 8.5 & 10 & 1.5 & 85.5 \\
spécial & 12 & 12 & 0.5 & 70 \\
\bottomrule
\end{NiceTabular}

```

Produit	dimensions (cm)			Prix
	L	l	h	
petit	3	5.5	1	30
moyen	5.5	8	1.5	50.5
premium	8.5	10.5	2	80
extra	8.5	10	1.5	85.5
spécial	12	12	0.5	70

On a utilisé le type de colonne `S` de `siunitx`.

6.3 Outils de coloriage avec la syntaxe de `colortbl`

On peut accéder aux outils de coloriage précédents avec une syntaxe proche de celle proposée par `colortbl` (même si `colortbl` n'est pas chargé). On doit pour cela utiliser la clé `colortbl-like` dans l'environnement concerné.²²

On a alors accès aux trois commandes suivantes :

- `\cellcolor` qui colorie la case courante²³ ;
- `\rowcolor` à utiliser dans une case et qui colorie le reste de la rangée ;
- `\columncolor` à utiliser dans le préambule du tableau de la même manière que la commande éponyme de `colortbl` (néanmoins, contrairement à la commande `\columncolor` de `colortbl`, celle de `nicematrix` peut apparaître à l'intérieur d'une autre commande, elle-même utilisée dans le préambule).

```

\NewDocumentCommand { \Blue } { } { \columncolor{blue!15}}
\begin{NiceTabular}[colortbl-like]{>{\Blue}c>{\Blue}cc}
\toprule

```

²². Pour le moment, cette clé n'est *pas* disponible dans `\NiceMatrixOptions`.

²³. Néanmoins, cette commande `\cellcolor` supprimera les espaces qui la suivent, ce que ne fait pas la commande `\cellcolor` de `colortbl`.

```

\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}

```

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

7 La commande `\RowStyle`

La commande `\RowStyle` prend en argument des instructions de mise en forme qui seront appliquées à chacune des cases restantes sur la rangée en cours.

Elle prend aussi en premier argument optionnel, entre crochets, une liste de couples *clé=valeur*.

- La clé `nb-rows` indique le nombre de rangées consécutives concernées par les spécifications de cette commande (une valeur `*` signifie que toutes les rangées restantes seront concernées).
- Les clés `cell-space-top-limit`, `cell-space-bottom-limit` et `cell-space-limits` sont disponibles avec le même effet que les clés globales de même nom (cf. p. 2).
- La clé `rowcolor` fixe la couleur de fond et la clé `color` fixe la couleur du texte.²⁴
- La clé `bold` impose des caractères gras aux éléments de la rangée, qu'ils soient en mode texte ou bien en mode mathématique.

```

\begin{NiceTabular}{cccc}
\hline
\RowStyle[cell-space-limits=3pt]{\rotate}
premier & deuxième & troisième & quatrième \\
\RowStyle[nb-rows=2,color=white,rowcolor=blue!50]{\sffamily}
1 & 2 & 3 & 4 \\
I & II & III & IV
\end{NiceTabular}

```

premier	deuxième	troisième	quatrième
1	2	3	4
I	II	III	IV

La commande `\rotate` est présentée p. 36.

8 La largeur des colonnes

8.1 Techniques de base

Dans les environnements avec un préambule explicite (comme `{NiceTabular}`, `{NiceArray}`, etc.), il est possible de fixer la largeur d'une colonne avec les lettres classiques `w`, `W`, `p`, `b` et `m` de l'extension `array`.

²⁴ La clé `color` utilise la commande `\color` mais insère aussi une instruction `\leavevmode` devant. Cela évite un espace vertical parasite dans les cellules qui correspondent à des colonnes de type `p`, `b`, `m` et `X` (qui débutent en mode vertical).

```
\begin{NiceTabular}{W{c}{2cm}cc}[hvlines]
Paris & New York & Madrid \\
Berlin & London & Roma \\
Rio & Tokyo & Oslo
\end{NiceTabular}
```

Paris	New York	Madrid
Berlin	London	Roma
Rio	Tokyo	Oslo

Dans les environnements de `nicematrix`, il est aussi possible de fixer la largeur *minimale* de toutes les colonnes (à l'exception des éventuelles colonnes extérieures : cf. p. 21) directement avec l'option `columns-width`.

```
$\begin{pNiceMatrix}[columns-width = 1cm]
1 & 12 & -123 \\
12 & 0 & 0 \\
4 & 1 & 2
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 12 & -123 \\ 12 & 0 & 0 \\ 4 & 1 & 2 \end{pmatrix}$$

Notez que l'espace inséré entre deux colonnes (égal à 2 `\tabcolsep` dans `{NiceTabular}` et à 2 `\arraycolsep` dans les autres environnements) n'est pas supprimé (il est évidemment possible de le supprimer en mettant `\tabcolsep` ou `\arraycolsep` à 0 avant).

Il est possible de donner la valeur spéciale `auto` à l'option `columns-width` : toutes les colonnes du tableau auront alors une largeur égale à la largeur de la case la plus large du tableau.²⁵

```
$\begin{pNiceMatrix}[columns-width = auto]
1 & 12 & -123 \\
12 & 0 & 0 \\
4 & 1 & 2
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 12 & -123 \\ 12 & 0 & 0 \\ 4 & 1 & 2 \end{pmatrix}$$

Sans surprise, il est possible de fixer la largeur minimale de toutes les colonnes de tous les tableaux dans une certaine portion de document avec la commande `\NiceMatrixOptions`.

```
\NiceMatrixOptions{columns-width=10mm}
$\begin{pNiceMatrix}
a & b \\ c & d
\end{pNiceMatrix}
=
\begin{pNiceMatrix}
1 & 1245 \\ 345 & 2
\end{pNiceMatrix}$
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 1245 \\ 345 & 2 \end{pmatrix}$$

Mais il est aussi possible de fixer une zone dans laquelle toutes les matrices auront leurs colonnes de la même largeur, égale à la largeur de la case la plus large de toutes les matrices de la zone. Cette construction utilise l'environnement `{NiceMatrixBlock}` avec l'option `auto-columns-width`²⁶. L'environnement `{NiceMatrixBlock}` n'a pas de rapport direct avec la commande `\Block` présentée précédemment dans ce document (cf. p. 4).

```
\begin{NiceMatrixBlock}[auto-columns-width]
$\begin{array}{c}
\begin{bNiceMatrix}
9 & 17 \\ -2 & 5
\end{bNiceMatrix} \\
\begin{bNiceMatrix}
1 & 1245345 \\ 345 & 2
\end{bNiceMatrix}
\end{array}$
\end{NiceMatrixBlock}
```

$$\begin{bmatrix} 9 & 17 \\ -2 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1245345 \\ 345 & 2 \end{bmatrix}$$

25. Le résultat est atteint dès la première compilation (mais PGF/Tikz écrivant des informations dans le fichier `aux`, un message demandant une deuxième compilation apparaîtra).

26. Pour le moment, c'est le seul usage de l'environnement `{NiceMatrixBlock}` mais il pourrait y en avoir davantage dans le futur.

8.2 Les colonnes V de varwidth

Rappelons d'abord le fonctionnement d'un environnement `{varwidth}` de l'extension éponyme `varwidth`. Un tel environnement est similaire à l'environnement classique `{minipage}` mais la largeur indiquée (en argument) n'est que la largeur *maximale* de la boîte créée. Dans le cas général, la largeur d'une boîte `{varwidth}` est la largeur naturelle de son contenu.

Cela est illustré avec les exemples suivants :

```
\fbox{%
\begin{varwidth}{8cm}
\begin{itemize}
\item premier item
\item deuxième item
\end{itemize}
\end{varwidth}}
```

— premier item
— deuxième item

```
\fbox{%
\begin{minipage}{8cm}
\begin{itemize}
\item premier item
\item deuxième item
\end{itemize}
\end{minipage}}
```

— premier item
— deuxième item

L'extension `varwidth` définit également le type de colonne `V`. Une colonne `V{<dim>}` encapsule toutes ses cases dans une `{varwidth}` d'argument `<dim>` (et effectue quelques réglages supplémentaires).

Lorsque l'extension `varwidth` est chargée, ces colonnes `V` de `varwidth` sont prises en charge par `nicematrix`. Dans le cadre de `nicematrix`, l'un des intérêts des colonnes de type `V` par rapport aux colonnes de type `p`, `m` ou `b` est que, pour les cases d'une telle colonne, le nœud PGF/Tikz créé pour le contenu d'une telle case a une largeur ajustée au contenu de la cellule en question : cf. p. 40. Si le contenu est vide, la cellule sera considérée comme vide par `nicematrix` pour le tracé des lignes pointillées et la détermination des « coins vides » (ce n'est pas le cas avec une case d'une colonne `p`, `m` ou `b`).

```
\begin{NiceTabular}[corners=NW,hvlines]{V{3cm}V{3cm}V{3cm}}
& un très très très très long texte & un très très très très long texte \\
un très très très très long texte & \\
un très très très très long texte & \\
\end{NiceTabular}
```

	un très très très très long texte	un très très très très long texte
un très très très très long texte		
un très très très très long texte		

Remarquons que l'extension `varwidth` a quelques problèmes (au moins dans sa version 0.92). Par exemple, avec LuaLaTeX, elle en fonctionne pas si le contenu commence par une instruction `\color`.

8.3 Les colonnes X

L'environnement `{NiceTabular}` propose aussi des colonnes `X` similaires à celles proposées par l'environnement `{tabularx}` de l'extension éponyme.

La valeur requise par la largeur du tableau peut être passée en argument de la clé `width` (dans `{NiceTabular}` ou dans `\NiceMatrixOptions`). La valeur initiale de ce paramètre est `\linewidth` (et non `\textwidth`).

Pour se rapprocher davantage de l’environnement `{tabularx}`, `nicematrix` propose aussi un environnement `{NiceTabularX}` qui prend en premier argument (obligatoire) la largeur voulue pour le tableau.²⁷

Comme avec les extensions `tabu`²⁸ et `tabularray`, le spécificateur `X` accepte entre crochets un argument optionnel qui est une liste de clés.

- On peut spécifier un poids pour la colonne en mettant directement un entier positif comme argument du spécificateur `X`. Par exemple, une colonne `X[2]` aura une largeur double de celle d’une colonne `X` (qui a un poids de 1).²⁹
- On peut spécifier l’alignement horizontal avec une des lettres `l`, `c` et `r` (qui insèrent respectivement `\raggedright`, `\centering` et `\raggedleft` suivi de `\arraybackslash`).
- On peut spécifier l’alignement vertical avec l’une des lettres `t` (alias `p`), `m` et `b` (qui construisent respectivement des colonnes de types `p`, `m` et `b`). La valeur par défaut est `t`.

```
\begin{NiceTabular}[width=9cm]{X[2,l]X[l]}[hvlines]
Un texte relativement long qui tient sur plusieurs lignes. &
Un texte relativement long qui tient sur plusieurs lignes. \\
Un texte plus court. & Un texte plus court.
\end{NiceTabular}
```

Un texte relativement long qui tient sur plusieurs lignes.	Un texte relativement long qui tient sur plusieurs lignes.
Un texte plus court.	Un texte plus court.

9 Les rangées et colonnes extérieures

Les environnements de `nicematrix` permettent de composer des rangées et des colonnes « extérieures » grâce aux options `first-row`, `last-row`, `first-col` et `last-col`. C’est particulièrement intéressant pour les matrices (mathématiques).

Si elle est présente, la « première rangée » (extérieure) est numérotée par 0 (et non 1). Il en est de même pour la « première colonne ».

```
$\begin{pNiceMatrix}[first-row,last-row,first-col,last-col,nullify-dots]
& C_1 & & \Cdots & & C_4 & & \\
L_1 & & a_{11} & & a_{12} & & a_{13} & & a_{14} & & L_1 & \\
\Vdots & & a_{21} & & a_{22} & & a_{23} & & a_{24} & & \Vdots & \\
& & a_{31} & & a_{32} & & a_{33} & & a_{34} & & & \\
L_4 & & a_{41} & & a_{42} & & a_{43} & & a_{44} & & L_4 & \\
& & C_1 & & \Cdots & & C_4 & & & & \\
\end{pNiceMatrix}$
```

$$\begin{array}{c}
C_1 \dots\dots\dots C_4 \\
L_1 \left(\begin{array}{cccc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right) L_1 \\
\vdots \quad \quad \quad \vdots \\
L_4 \quad \quad \quad L_4 \\
C_1 \dots\dots\dots C_4
\end{array}$$

27. Si `tabularx` est chargé, on doit obligatoirement utiliser `{NiceTabularX}` pour utiliser les colonnes `X` (cela est dû à un problème de conflit de définition des spécificateurs de colonne `X`).

28. L’extension `tabu` est maintenant considérée comme obsolète.

29. Les valeurs négatives pour les poids, comme proposées par `tabu` (maintenant obsolète), ne sont *pas* prises en charge par `nicematrix`. Si une telle valeur est utilisée, une erreur sera levée.

Les lignes pointillées ont été tracées avec les outils qui seront présentés p. 23.

Il y a plusieurs remarques à formuler.

- Si on utilise un environnement avec préambule explicite (`{NiceTabular}`, `{NiceArray}` ou l'une de ses variantes), on ne doit pas mettre dans ce préambule de spécification de colonne pour les éventuelles première et dernière colonne : ce sera automatiquement (et nécessairement) une colonne `r` pour la première colonne et une colonne `l` pour la dernière.³⁰
- On peut se demander comment `nicematrix` détermine le nombre de rangées et de colonnes nécessaires à la composition de la « dernière rangée » et de la « dernière colonne ».
 - Dans le cas d'un environnement avec préambule, comme `{NiceTabular}` ou `{pNiceArray}`, le nombre de colonnes se déduit évidemment du préambule.
 - Dans le cas où l'option `light-syntax` (cf. p. 38) est utilisée, `nicematrix` profite du fait que cette option nécessite de toutes manières le chargement complet du contenu de l'environnement (d'où l'impossibilité de mettre du verbatim dans ce cas-là) avant composition du tableau. L'analyse du contenu de l'environnement donne le nombre de rangées et de colonnes.
 - Dans les autres cas, `nicematrix` détermine le nombre de rangées et de colonnes à la première compilation et l'écrit dans le fichier `aux` pour pouvoir l'utiliser à la compilation suivante. *Néanmoins, il est possible de donner le numéro de la dernière rangée et le numéro de la dernière colonne en arguments des options `last-row` et `last-col`, ce qui permettra d'accélérer le processus complet de compilation. C'est ce que nous ferons dans la suite.*

On peut contrôler l'apparence de ces rangées et colonnes avec les options `code-for-first-row`, `code-for-last-row`, `code-for-first-col` et `code-for-last-col`. Ces options sont des listes de tokens qui seront insérées au début de chaque case de la rangée ou de la colonne considérée.

```
\NiceMatrixOptions{code-for-first-row = \color{red},
                  code-for-first-col = \color{blue},
                  code-for-last-row = \color{green},
                  code-for-last-col = \color{magenta}}
$\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col,nullify-dots]
    & C_1 & & \Cdots & & C_4 & & \\
L_1 & & a_{11} & & a_{12} & & a_{13} & & a_{14} & & L_1 & \\
\vdots & & a_{21} & & a_{22} & & a_{23} & & a_{24} & & \vdots & \\
\hline
    & & a_{31} & & a_{32} & & a_{33} & & a_{34} & & & \\
L_4 & & a_{41} & & a_{42} & & a_{43} & & a_{44} & & L_4 & \\
    & & C_1 & & \Cdots & & C_4 & & & & & \\
\end{pNiceArray}$
```

$$\begin{array}{c}
 \textcolor{red}{C_1} \cdots \cdots \cdots \textcolor{red}{C_4} \\
 \textcolor{blue}{L_1} \left(\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right) \textcolor{magenta}{L_1} \\
 \vdots \quad \quad \quad \vdots \\
 \textcolor{blue}{L_4} \quad \quad \quad \textcolor{magenta}{L_4} \\
 \textcolor{green}{C_1} \cdots \cdots \cdots \textcolor{green}{C_4}
 \end{array}$$

Remarques

- Comme on peut le voir dans l'exemple précédent, les filets horizontaux et verticaux ne s'étendent pas dans les rangées et colonnes extérieures. Cette remarque s'applique aussi aux filets définis par les outils de personnalisation de `nicematrix` (cf. la clé `custom-line` p. 11).
- Une spécification de couleur présente dans `code-for-first-row` s'applique à une ligne pointillée tracée dans cette « première rangée » (sauf si une valeur a été donnée à `xdots/color`). Idem pour les autres.

³⁰. Si on souhaite une colonne extérieure avec un autre type d'alignement, on aura intérêt à considérer la commande `\SubMatrix` disponible dans le `\CodeAfter` (cf. p. 29).

- Sans surprise, une éventuelle option `columns-width` (décrite p. 18) ne s’applique pas à la « première colonne » ni à la « dernière colonne ».
- Pour des raisons techniques, il n’est pas possible d’utiliser l’option de la commande `\` après la « première rangée » ou avant la « dernière rangée ». Le placement des délimiteurs serait erroné. Pour contourner cette restriction, on pourra envisager d’utiliser la commande `\SubMatrix` dans le `\CodeAfter` (cf. p. 29).

10 Les lignes en pointillés continues

À l’intérieur des environnements de l’extension `nicematrix`, de nouvelles commandes sont définies : `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` et `\Iddots`. Ces commandes sont conçues pour être utilisées à la place de `\dots`, `\cdots`, `\vdots`, `\ddots` et `\iddots`.³¹

Chacune de ces commandes doit être utilisée seule dans la case du tableau et elle trace une ligne en pointillés entre les premières cases non vides³² situées de part et d’autre de la case courante. Bien entendu, pour `\Ldots` et `\Cdots`, c’est une ligne horizontale ; pour `\Vdots`, c’est une ligne verticale et pour `\Ddots` et `\Iddots`, ce sont des lignes diagonales. On peut changer la couleur d’une ligne avec l’option `color`.³³

```
\begin{bNiceMatrix}
a_1      & \Cdots &      & & a_1      & \\
\Vdots   & a_2    & \Cdots & & a_2      & \\
        & & \Vdots & \Ddots[color=red] & & \\
\\
a_1      & a_2    &      & & a_n      & \\
\end{bNiceMatrix}
```

$$\begin{bmatrix} a_1 & \cdots & a_1 \\ \vdots & & \vdots \\ & a_2 & \cdots & a_2 \\ \vdots & & & \vdots \\ a_1 & a_2 & & a_n \end{bmatrix}$$

Pour représenter la matrice nulle, on peut choisir d’utiliser le codage suivant :

```
\begin{bNiceMatrix}
0      & \Cdots & 0      & \\
\Vdots & & \Vdots & \\
0      & \Cdots & 0      & \\
\end{bNiceMatrix}
```

$$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

On peut néanmoins souhaiter une matrice plus grande. Habituellement, dans un tel cas, les utilisateurs de LaTeX ajoutent une nouvelle ligne et une nouvelle colonne. Il est possible d’utiliser la même méthode avec `nicematrix` :

```
\begin{bNiceMatrix}
0      & \Cdots & \Cdots & 0      & \\
\Vdots & & & \Vdots & \\
\Vdots & & & \Vdots & \\
0      & \Cdots & \Cdots & 0      & \\
\end{bNiceMatrix}
```

$$\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$$

Dans la première colonne de cet exemple, il y a deux instructions `\Vdots` mais, bien entendu, une seule ligne en pointillés sera tracée.

En fait, dans cet exemple, il aurait été possible de tracer la même matrice plus rapidement avec le

31. La commande `\iddots`, définie par `nicematrix`, est une variante de `\ddots` avec les points allant vers le haut. Si `mathdots` est chargée, la version de `mathdots` est utilisée. Elle correspond à la commande `\adots` de `unicode-math`.

32. La définition précise de ce qui est considéré comme une « case vide » est donnée plus loin (cf. p. 45).

33. Il est aussi possible de changer la couleur de toutes ces lignes pointillées avec l’option `xdots/color` (`xdots` pour rappeler que cela s’applique à `\Cdots`, `\Ldots`, `\Vdots`, etc.) : cf. p. 27.

codage suivant (parmi d'autres) :

```
\begin{bNiceMatrix}
0      & \Cdots &      & 0      & \\
\Vdots &      &      &      & \\
      &      &      & \Vdots & \\
0      &      & \Cdots & 0      & \\
\end{bNiceMatrix}
```

$$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ & & \\ 0 & \cdots & 0 \end{bmatrix}$$

Il y a aussi d'autres moyens de changer la taille d'une matrice. On pourrait vouloir utiliser l'argument optionnel de la commande `\hspace*` pour l'espacement vertical et la commande `\hspace*` dans une case pour l'espacement horizontal.³⁴

Toutefois, une commande `\hspace*` pourrait interférer dans la construction des lignes en pointillés. C'est pourquoi l'extension `nicematrix` fournit une commande `\Hspace` qui est une variante de `\hspace` transparente pour la construction des lignes en pointillés de `nicematrix`.

```
\begin{bNiceMatrix}
0      & \Cdots & \Hspace*{1cm} & 0      & \\
\Vdots &      &      & \Vdots & \\
0      & \Cdots &      & 0      & \\
\end{bNiceMatrix}
```

$$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ & & \\ 0 & \cdots & 0 \end{bmatrix}$$

10.1 L'option `nullify-dots`

Considérons la matrice suivante qui a été composée classiquement avec l'environnement `{pmatrix}` de `amsmath`.

```
$A = \begin{pmatrix}
h & i & j & k & l & m \\
x & & & & & x
\end{pmatrix}
```

$$A = \begin{pmatrix} h & i & j & k & l & m \\ x & & & & & x \end{pmatrix}$$

Si nous ajoutons des instructions `\ldots` dans la seconde rangée, la géométrie de la matrice est modifiée.

```
$B = \begin{pmatrix}
h & i & j & k & l & m \\
x & \ldots & \ldots & \ldots & \ldots & x
\end{pmatrix}
```

$$B = \begin{pmatrix} h & i & j & k & l & m \\ x & \dots & \dots & \dots & \dots & x \end{pmatrix}$$

Par défaut, avec `nicematrix`, si nous remplaçons `{pmatrix}` par `{pNiceMatrix}` et `\ldots` par `\Ldots`, la géométrie de la matrice n'est pas changée.

```
$C = \begin{pNiceMatrix}
h & i & j & k & l & m \\
x & \Ldots & \Ldots & \Ldots & \Ldots & x
\end{pNiceMatrix}
```

$$C = \begin{pmatrix} h & i & j & k & l & m \\ x & \dots & \dots & \dots & \dots & x \end{pmatrix}$$

On pourrait toutefois préférer la géométrie de la première matrice A et vouloir avoir la même géométrie avec une ligne en pointillés continue dans la seconde rangée. C'est possible en utilisant l'option `nullify-dots` (et une seule instruction `\Ldots` suffit).

```
$D = \begin{pNiceMatrix}[nullify-dots]
h & i & j & k & l & m \\
x & \Ldots & & & & x
\end{pNiceMatrix}
```

$$D = \begin{pmatrix} h & i & j & k & l & m \\ x & \dots & & & & x \end{pmatrix}$$

L'option `nullify-dots` « smashe » les instructions `\Ldots` (et ses variantes) horizontalement mais aussi verticalement.

34. Dans `nicematrix`, il faut utiliser `\hspace*` et non `\hspace` car `nicematrix` utilise `array`. Remarquons aussi que l'on peut également régler la largeur des colonnes en utilisant l'environnement `{NiceArray}` (ou une de ses variantes) avec une colonne de type `w` ou `W` : cf. p. 18

10.2 Les commandes `\Hdotsfor` et `\Vdotsfor`

Certaines personnes utilisent habituellement la commande `\hdotsfor` de l'extension `amsmath` pour tracer des lignes en pointillés horizontales dans une matrice. Dans les environnements de `nicematrix`, il convient d'utiliser `\Hdotsfor` à la place pour avoir les lignes en pointillés similaires à toutes celles tracées par l'extension `nicematrix`.

Comme avec les autres commandes de `nicematrix` (comme `\Cdots`, `\Ldots`, `\Vdots`, etc.), la ligne en pointillés tracée par `\Hdotsfor` s'étend jusqu'au contenu des cases de part et d'autre.

```


$$\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
1 & \Hdotsfor{3} & & 5 \\
1 & 2 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5
\end{pNiceMatrix}$$


```

Néanmoins, si ces cases sont vides, la ligne en pointillés s'étend seulement dans les cases spécifiées par l'argument de `\Hdotsfor` (par conception).

```


$$\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
& \Hdotsfor{3} \\
1 & 2 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5
\end{pNiceMatrix}$$


```

Remarque : Contrairement à la commande `\hdotsfor` de `amsmath`, la commande `\Hdotsfor` est utilisable même lorsque l'extension `colortbl`³⁵ est chargée (mais vous risquez d'avoir des problèmes si vous utilisez `\rowcolor` sur la même rangée que `\Hdotsfor`).

L'extension `nicematrix` propose aussi une commande `\Vdotsfor` similaire à `\Hdotsfor` mais traçant des lignes verticales.

L'exemple suivant utilise à la fois `\Hdotsfor` et `\Vdotsfor` :

```

\begin{bNiceMatrix}
C[a_1,a_1] & \Cdots & C[a_1,a_n] \\
& \hspace*{20mm} & C[a_1,a_1^{(p)}] & \Cdots & C[a_1,a_n^{(p)}] \\
\Vdots & \Ddots & \Vdots \\
& \Hdotsfor{1} & \Vdots & \Ddots & \Vdots \\
C[a_n,a_1] & \Cdots & C[a_n,a_n] \\
& & C[a_n,a_1^{(p)}] & \Cdots & C[a_n,a_n^{(p)}] \\
\rule{0pt}{15mm} & \Vdotsfor{1} & & \Ddots & & \Vdotsfor{1} \\
C[a_1^{(p)},a_1] & \Cdots & C[a_1^{(p)},a_n] \\
& & C[a_1^{(p)},a_1^{(p)}] & \Cdots & C[a_1^{(p)},a_n^{(p)}] \\
\Vdots & \Ddots & \Vdots \\
& \Hdotsfor{1} & \Vdots & \Ddots & \Vdots \\
C[a_n^{(p)},a_1] & \Cdots & C[a_n^{(p)},a_n] \\
& & C[a_n^{(p)},a_1^{(p)}] & \Cdots & C[a_n^{(p)},a_n^{(p)}]
\end{bNiceMatrix}

```

35. On rappelle que lorsque l'extension `xcolor` est chargée avec l'option `table`, l'extension `colortbl` est chargée.

$$\left[\begin{array}{ccc} C[a_1, a_1] \cdots \cdots C[a_1, a_n] & & C[a_1, a_1^{(p)}] \cdots \cdots C[a_1, a_n^{(p)}] \\ \vdots & \ddots & \vdots \\ C[a_n, a_1] \cdots \cdots C[a_n, a_n] & & C[a_n, a_1^{(p)}] \cdots \cdots C[a_n, a_n^{(p)}] \\ & \ddots & \\ C[a_1^{(p)}, a_1] \cdots \cdots C[a_1^{(p)}, a_n] & & C[a_1^{(p)}, a_1^{(p)}] \cdots \cdots C[a_1^{(p)}, a_n^{(p)}] \\ \vdots & \ddots & \vdots \\ C[a_n^{(p)}, a_1] \cdots \cdots C[a_n^{(p)}, a_n] & & C[a_n^{(p)}, a_1^{(p)}] \cdots \cdots C[a_n^{(p)}, a_n^{(p)}] \end{array} \right]$$

10.3 Comment créer les lignes en pointillés de manière transparente

Si on a un document déjà tapé qui contient un grand nombre de matrices avec des points de suspension, on peut souhaiter utiliser les lignes pointillées de `nicematrix` sans avoir à modifier chaque matrice. Pour cela, `nicematrix` propose deux options `renew-dots` et `renew-matrix`.³⁶

— L’option `renew-dots`

Avec cette option, les commandes `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`³¹ et `\hdotsfor` sont redéfinies dans les environnements de `nicematrix` et agissent alors comme `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` et `\Hdotsfor` ; la commande `\dots` (points de suspension « automatiques » de `amsmath`) est aussi redéfinie et se comporte comme `\Ldots`.

— L’option `renew-matrix`

Avec cette option, l’environnement `{matrix}` est redéfini et se comporte comme `{NiceMatrix}` et il en est de même pour les cinq variantes.

Par conséquent, avec les options `renew-dots` et `renew-matrix`, un code classique donne directement le résultat fourni par `nicematrix`.

```
\NiceMatrixOptions{renew-dots,renew-matrix}
\begin{pmatrix}
1 & & \cdots & & \cdots & & 1 & \\
0 & & \ddots & & & & & \vdots \\
\vdots & & \ddots & & \ddots & & \vdots & \\
0 & & \cdots & & 0 & & & 1
\end{pmatrix}
\end{pmatrix}
```

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

10.4 Les labels des lignes en pointillés

Les commandes `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` et `\Hdotsfor` (ainsi que la commande `\line` dans le `\CodeAfter` décrite p. 28) peuvent en fait prendre deux arguments optionnels spécifiés par les caractères `_` et `^` pour des labels situés au-dessous et au-dessus de la ligne. Les arguments sont composés en mode mathématique avec `\scriptstyle`.

```
$\begin{bNiceMatrix}
1 & \hspace*{1cm} & & 0 \\\[8mm]
& \Ddots^{n \text{ fois}} & & \\
0 & & & 1
\end{bNiceMatrix}
```

$$\begin{bmatrix} 1 & & & 0 \\ & \ddots & & \\ & & n \text{ fois} & \\ 0 & & & 1 \end{bmatrix}$$

³⁶. Comme toutes les autres options, les options `renew-dots` et `renew-matrix` peuvent être fixées avec la commande `\NiceMatrixOptions`, mais ces deux options-là peuvent aussi être passées en option du `\usepackage`.

10.5 Personnalisation des lignes en pointillés

Les lignes pointillées tracées par `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` et `\Vdotsfor` (ainsi que par la commande `\line` dans le `\CodeAfter` décrite p. 28) peuvent être paramétrées par les options suivantes (que l'on met entre crochets après la commande) :

- `color` ;
- `radius` ;
- `shorten-start`, `shorten-end` et `shorten` ;
- `inter` ;
- `line-style`.

Ces options peuvent aussi être fixées avec `\NiceMatrixOptions`, comme options de `\CodeAfter`, ou bien au niveau d'un environnement mais elles doivent alors être préfixées par `xdots` (`xdots` pour rappeler que cela s'applique à `\Cdots`, `\Ldots`, `\Vdots`, etc.), ce qui fait que leurs noms deviennent :

- `xdots/color` ;
- `xdots/radius` ;
- `xdots/shorten-start`, `xdots/shorten-end` et `xdots/shorten` ;
- `xdots/inter` ;
- `xdots/line-style`.

Pour la clarté, dans la suite, on utilisera ces noms-là.

L'option `xdots/color`

L'option `xdots/color` indique bien entendu la couleur de la ligne tracée. On remarquera néanmoins que les lignes tracées dans les rangées et colonnes extérieures (décrites plus loin) bénéficient d'un régime spécial : cf. p. 21.

Nouveau 6.9 L'option `xdots/radius`

L'option `radius` correspond au rayon des points circulaires qui sont tracés. La valeur initiale est 0.53 pt.

L'option `xdots/shorten`

Les clés `xdots/shorten-start` et `xdots/shorten-end` indiquent la marge qui est laissée aux deux extrémités de la ligne. La clé `xdots/shorten` fixe les deux clés simultanément. La valeur initiale de 0.3 em (il est conseillé d'utiliser une unité de mesure dépendante de la fonte courante).

Nouveau 6.10 Les clés `xdots/shorten-start` et `xdots/shorten-end` ont été introduites dans la version 6.10. Les versions précédentes ne proposaient que `xdots/shorten`.

Nouveau 6.9 L'option `xdots/inter`

L'option `xdots/inter` indique la distance entre deux points. La valeur initiale est 0.45 em (il est conseillé d'utiliser une unité de mesure dépendante de la fonte courante).

L'option `xdots/line-style`

Il faut savoir que, par défaut, les lignes de Tikz tracées avec le paramètre `dotted` sont composées de points carrés et non pas ronds.³⁷

```
\tikz \draw [dotted] (0,0) -- (5,0) ;
```

Voulant proposer des lignes avec des points ronds dans le style de celui de `\ldots` (au moins celui des fontes *Computer Modern*), l'extension `nicematrix` contient en interne son propre système de ligne en pointillés (qui, au passage, n'utilise que `pgf` et non `tikz`). Ce style est appelé le style **standard**. Cette valeur est la valeur initiale du paramètre `xdots/line-style`.

³⁷. La raison de départ est que le format PDF comporte un système de description de lignes en tirets, qui, puisqu'il est incorporé dans le PDF, est affiché très rapidement par les lecteurs de PDF. Il est facile à partir de ce type de ligne de créer des lignes de points carrés alors qu'une ligne de points ronds doit être construite explicitement point par point. Voir néanmoins à l'adresse suivante pour un moyen d'avoir un style de pointillés ronds avec Tikz : <https://tex.stackexchange.com/questions/52848/tikz-line-with-large-dots>

Néanmoins (quand Tikz est chargé), on peut utiliser pour `xdots/line-style` n'importe quel style proposé par Tikz, c'est-à-dire n'importe quelle suite d'options Tikz applicables à un chemin (à l'exception de « `color` », « `shorten >` » et « `shorten <` »).

Voici par exemple une matrice tridiagonale avec le style `loosely dotted` :

```
\begin{pNiceMatrix}[nullify-dots,xdots/line-style=loosely dotted]
a      & b      & 0      & & & \Cdots & 0      & \\
b      & a      & b      & & \Ddots & & & \Vdots \\
0      & b      & a      & & \Ddots & & & \\
      & & \Ddots & & \Ddots & & \Ddots & \\
\Vdots & & & & & & & 0      \\
0      & \Cdots & & & 0      & & b      & a
\end{pNiceMatrix}
```

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

10.6 Les lignes pointillées et les filets

Les lignes pointillées délimitent des blocs virtuels qui ont le même comportement vis à vis des filets que les blocs créés par `\Block` (les filets spécifiés par le spécificateur `|` dans le préambule, la commande `\Hline`, les clés `vlines`, `hlines`, `hvlines` et `hvlines-except-borders` et les outils créés par `custom-line` ne sont pas tracés dans les blocs).³⁸

```
\begin{bNiceMatrix}[margin,hvlines]
\Block{3-3}<\LARGE>{A} & & 0 \\
& \hspace*{1cm} & & \Vdots \\
& & 0 \\
0 & \Cdots & 0 & 0
\end{bNiceMatrix}
```

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

11 Le `\CodeAfter`

On a présenté p. 14 la clé `code-before`. Il existe en fait une clé similaire `code-after` qui peut être utilisée pour indiquer du code qui sera exécuté *après* la construction du tableau.

Pour améliorer la lisibilité du code, une syntaxe alternative est proposée : on peut spécifier les instructions du `code-after` à la fin de l'environnement, après le mot-clé `\CodeAfter`. Bien que ce soit un mot-clé, `\CodeAfter` accepte quand même un argument optionnel (entre crochets). Les clés autorisées dans cet argument optionnel forment un sous-ensemble des clés acceptées par `\NiceMatrixOptions`.

Les utilisateurs expérimentés peuvent, en particulier, utiliser les nœuds PGF/Tikz créés par `nicematrix` dans le `\CodeAfter`. Ces nœuds sont décrits à partir de la page 39.

Par ailleurs, plusieurs commandes spéciales sont disponibles dans le `\CodeAfter` : `\line`, `\SubMatrix`, `\OverBrace` et `\UnderBrace`. On va maintenant détailler ces commandes.

11.1 La commande `\line` dans le `\CodeAfter`

La commande `\line` permet de tracer directement des lignes en pointillés entre les cases. Elle prend deux arguments correspondant aux cases ou blocs à relier. Chacun de ces deux arguments peut être :

- une spécification de case de la forme $i-j$ où i est le numéro de ligne et j est le numéro de colonne ;

³⁸. En revanche, la commande `\line` dans le `\CodeAfter` (cf. p. 28) ne crée pas de bloc.

- **Nouveau 6.10** le nom d'un bloc (créé avec la commande `\Block` en utilisant la clé `name` de cette commande).

Les options disponibles pour personnaliser les lignes pointillées créées par `\Cdots`, `\Vdots`, etc. peuvent aussi être passées à cette commande (cf. p. 27).

Cette commande peut par exemple être utilisée pour tracer une ligne entre deux cases adjacentes.

```
\NiceMatrixOptions{xdots/shorten = 0.6 em}
```

```
\begin{pNiceMatrix}
```

```
I      & 0      & \Cdots & 0      & \\\
```

```
0      & I      & \Ddots & \Vdots & \\\
```

```
\Vdots & \Ddots & I      & 0      & \\\
```

```
0      & \Cdots & 0      & I      & \\\
```

```
\CodeAfter \line{2-2}{3-3}
```

```
\end{pNiceMatrix}
```

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

Elle peut aussi être utilisée pour tracer une ligne diagonale non parallèle aux autres lignes diagonales (par défaut, les lignes tracées par `\Ddots` sont « parallélisées » : cf. p. 45).

```
\begin{bNiceMatrix}
```

```
1      & \Cdots & & 1      & 2      & \Cdots & & 2      & \\\
```

```
0      & \Ddots & & \Vdots & \Vdots & \hspace*{2.5cm} & & \Vdots & \\\
```

```
\Vdots & \Ddots & & & & & & & \\\
```

```
0      & \Cdots & 0 & 1      & 2      & \Cdots & & 2      & \\\
```

```
\CodeAfter \line[shorten=6pt]{1-5}{4-7}
```

```
\end{bNiceMatrix}
```

$$\left[\begin{array}{cc|cc} 1 & \cdots & 1 & 2 \\ 0 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \end{array} \right] \begin{array}{cc|cc} 2 & \cdots & 2 \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ 2 & \cdots & 2 \end{array}$$

11.2 La commande `\SubMatrix` dans le `\CodeAfter`

La commande `\SubMatrix` permet de positionner des délimiteurs sur une partie du tableau, partie qui est considérée comme une sous-matrice. La commande `\SubMatrix` prend cinq arguments :

- le premier argument est le délimiteur gauche qui peut être n'importe quel délimiteur extensible de LaTeX : `(`, `[`, `\{`, `\langle`, `\group`, `\lfloor`, etc. mais aussi le délimiteur nul `.` ;
- le deuxième argument est le coin supérieur gauche de la sous-matrice avec la syntaxe $i-j$ où i est le numéro de rangée et j le numéro de colonne ;
- le troisième argument est le coin inférieur droit avec la même syntaxe ;
- la quatrième argument est le délimiteur droit ;
- le cinquième argument, optionnel, entre crochets, est une liste de couples *clé=valeur*.³⁹

On remarquera que la commande `\SubMatrix` trace les délimiteurs après la construction de la matrice : aucun espace n'est inséré par la commande `\SubMatrix`. C'est pourquoi, dans l'exemple suivant, on a utilisé la clé `margin` et on a inséré à la main de l'espace entre la troisième et la quatrième colonne avec `@{\hspace{1.5em}}` dans le préambule du tableau.

```
\[ \begin{NiceArray}{ccc@{\hspace{1.5em}}c}[cell-space-limits=2pt,margin]
```

```
1      & 1      & 1      & x \\\
```

```
\dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\\
```

```
1      & 2      & 3      & z
```

```
\CodeAfter
```

```
\SubMatrix({1-1}{3-3})
```

```
\SubMatrix({1-4}{3-4})
```

```
\end{NiceArray} \]
```

$$\begin{pmatrix} 1 & 1 & 1 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

³⁹ Il n'y a pas d'argument optionnel entre crochets en première position car un crochet ouvrant juste après `\SubMatrix` doit pouvoir être interprété comme le premier argument (obligatoire) de `\SubMatrix` : ce crochet est alors le délimiteur gauche de la sous-matrice (ex. : `\SubMatrix[{2-2}{4-7}]`).

La commande `\SubMatrix` accepte en fait également deux arguments optionnels spécifiés par les symboles traditionnels `^` et `_` pour des éléments en exposant et en indice.

```
$\begin{bNiceMatrix}[right-margin=1em]
1 & 1 & 1 \\
1 & a & b \\
1 & c & d
\CodeAfter
\SubMatrix[{{2-2}}{{3-3}}]^T
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & b \\ 1 & c & d \end{bmatrix}^T$$

Les clés disponibles pour la commande `\SubMatrix` sont les suivantes :

- `left-xshift` et `right-xshift` déplacent horizontalement les délimiteurs (il existe aussi la clé `xshift` qui permet de régler simultanément ces deux clés) ;
- `extra-height` ajoute une quantité à la hauteur totale des délimiteurs (hauteur `\ht` + profondeur `\dp`) ;
- `delimiters/color` permet de fixer la couleur des délimiteurs (cette clé est également disponible dans `\NiceMatrixOptions` et au niveau des environnements à délimiteurs ou comme option de `\CodeAfter`) ;
- `slim` qui est une clé booléenne : lorsqu'elle est utilisée la position horizontale des délimiteurs est calculée uniquement sur le contenu des cases de la sous-matrice alors, que, dans le cas général, elle est calculée sur le contenu des cases des colonnes mises en jeu (voir exemple ci-dessous) ;
- `vlines` contient une liste de numéros de filets verticaux à tracer dans la sous-matrice (si cette clé est utilisée sans valeur, tous les filets verticaux sont tracés) ;
- `hlines` est similaire à `vlines` mais pour les filets horizontaux ;
- `hvlines`, qui s'utilise sans valeur, trace tous les filets dans la sous-matrice.

On remarquera que tous ces filets sont dessinés après la construction du tableau principal : les colonnes et les rangées ne sont pas écartées.

Ces clés sont aussi accessibles dans `\NiceMatrixOptions`, au niveau des environnements de `nicematrix` ou comme option de `\CodeAfter` avec le préfixe `sub-matrix`, c'est-à-dire qu'elles sont alors nommées `sub-matrix/left-xshift`, `sub-matrix/right-xshift`, `sub-matrix/xshift`, etc.

```
$\begin{NiceArray}{cc@{\hspace{5mm}}l}[cell-space-limits=2pt]
& & \frac{1}{2} \\
& & \frac{1}{4} \\
a & b & \frac{1}{2}a + \frac{1}{4}b \\
c & d & \frac{1}{2}c + \frac{1}{4}d
\CodeAfter
\SubMatrix({{1-3}}{{2-3}})
\SubMatrix({{3-1}}{{4-2}})
\SubMatrix({{3-3}}{{4-3}})
\end{NiceArray}$
```

$$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{4} \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \frac{1}{2}a + \frac{1}{4}b \\ \frac{1}{2}c + \frac{1}{4}d \end{pmatrix}$$

Voici le même exemple avec la clé `slim` pour l'une des sous-matrices.

```
$\begin{NiceArray}{cc@{\hspace{5mm}}l}[cell-space-limits=2pt]
& & \frac{1}{2} \\
& & \frac{1}{4} \\
a & b & \frac{1}{2}a + \frac{1}{4}b \\
c & d & \frac{1}{2}c + \frac{1}{4}d
\CodeAfter
\SubMatrix({{1-3}}{{2-3}})[slim]
\SubMatrix({{3-1}}{{4-2}})
\SubMatrix({{3-3}}{{4-3}})
\end{NiceArray}$
```

$$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{4} \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \frac{1}{2}a + \frac{1}{4}b \\ \frac{1}{2}c + \frac{1}{4}d \end{pmatrix}$$

Il existe aussi une clé `name` qui permet de donner un nom à une sous-matrice créée par une commande `\SubMatrix`. Ce nom est utilisé pour créer des nœuds PGF/Tikz : voir p. 43.

Il est également possible de spécifier des délimiteurs⁴⁰ en les plaçant dans le préambule de l'environnement (pour les environnements à préambule : `{NiceArray}`, `{pNiceArray}`, etc.). Cette syntaxe est inspirée de l'extension `blkarray`.

Dans le cas de deux délimiteurs successifs (nécessairement un fermant suivi d'un ouvrant pour une autre sous-matrice) un espace égal à `\enskip` est inséré automatiquement.

```
$\begin{pNiceArray}{(c)(c)(c)}
a_{11} & a_{12} & & a_{13} \\
a_{21} & \displaystyle \int_0^1 \frac{1}{x^2+1} dx & & a_{23} \\
a_{31} & a_{32} & & a_{33} \\
\end{pNiceArray}$
```

$$\begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} \begin{pmatrix} a_{12} & \int_0^1 \frac{1}{x^2+1} dx & a_{32} \end{pmatrix} \begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}$$

11.3 Les commandes `\OverBrace` et `\UnderBrace` dans le `\CodeAfter`

Les commandes `\OverBrace` and `\UnderBrace` permettent de placer des accolades horizontales sur une partie du tableau. Ces commandes prennent trois arguments :

- le premier argument est le coin supérieur gauche du rectangle de cellules impliquées dans l'accolade avec la syntaxe habituelle $i-j$ où i est le numéro de rangée et j le numéro de colonne ;
- le deuxième argument est le coin inférieur droit avec la même syntaxe ;
- le troisième argument est le « label » de l'accolade qui sera placé par `nicematrix` (avec PGF) au-dessus de l'accolade (pour la commande `\OverBrace`) ou au-dessous (pour `\UnderBrace`).

```
\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
11 & 12 & 13 & 14 & 15 & 16 \\
\CodeAfter
\OverBrace{1-1}{2-3}{A}
\OverBrace{1-4}{2-6}{B}
\end{pNiceMatrix}
```

$$\begin{pmatrix} \overbrace{1 \quad 2 \quad 3}^A & \overbrace{4 \quad 5 \quad 6}^B \\ 11 & 12 & 13 & 14 & 15 & 16 \end{pmatrix}$$

Les commandes `\OverBrace` et `\UnderBrace` acceptent en fait un premier argument optionnel (entre crochets) pour une liste de couples *clé=valeur*. Les clés disponibles sont les suivantes :

- `left-shorten` et `right-shorten` qui ne prennent pas de valeur ; quand `left-shorten` est utilisée, l'abscisse de l'extrémité de gauche de l'accolade est calculée à partir du contenu du sous-tableau concerné alors que, sinon, c'est la position du filet vertical éventuel qui est utilisée (de même pour `right-shorten`) ;
- `shorten`, qui est la conjonction des clés `left-shorten` et `right-shorten` ;
- `yshift`, qui déplace verticalement l'accolade (et son label) ;
- `color` qui fixe la couleur de l'accolade et du label.

```
\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
11 & 12 & 13 & 14 & 15 & 16 \\
\CodeAfter
\OverBrace[shorten,yshift=3pt]{1-1}{2-3}{A}
\OverBrace[shorten,yshift=3pt]{1-4}{2-6}{B}
\end{pNiceMatrix}
```

$$\begin{pmatrix} \overbrace{1 \quad 2 \quad 3}^A & \overbrace{4 \quad 5 \quad 6}^B \\ 11 & 12 & 13 & 14 & 15 & 16 \end{pmatrix}$$

⁴⁰. Les délimiteurs en question sont `(`, `[` et `\{` et les fermants correspondants. On peut bien sûr aussi mettre des `|` et des `||` dans le préambule.

12 Les notes dans les tableaux

12.1 Les notes de pied de page

L’extension `nicematrix` permet, en utilisant `footnote` ou bien `footnotehyper`, d’extraire les notes insérées avec `\footnote` dans un environnement de `nicematrix` pour les reporter en pied de page avec les autres notes du document.

Si `nicematrix` est chargée avec l’option `footnote` (avec `\usepackage[footnote]{nicematrix}` ou avec `\PassOptionsToPackage`), l’extension `footnote` est chargée (si elle ne l’est pas déjà) et elle est utilisée pour extraire les notes de pied de page.

Si `nicematrix` est chargée avec l’option `footnotehyper`, l’extension `footnotehyper` est chargée (si elle ne l’est pas déjà) et elle est utilisée pour extraire les notes de pied de page.

Attention : Les extensions `footnote` et `footnotehyper` sont incompatibles. L’extension `footnotehyper` est le successeur de l’extension `footnote` et devrait être utilisée préférentiellement. L’extension `footnote` a quelques défauts ; en particulier, elle doit être chargée après l’extension `xcolor` et elle n’est pas parfaitement compatible avec `hyperref`.

12.2 Les notes de tableaux

L’extension `nicematrix` propose aussi une commande `\tabularnote` qui permet de spécifier des notes qui seront composées à la fin du tableau avec une longueur de ligne égale à la largeur du tableau (hors éventuelles colonnes extérieures spécifiées par `first-col` et `last-col`). Sans surprise, cette commande n’est disponible que dans les environnements sans délimiteurs, c’est-à-dire `{NiceTabular}`, `{NiceArray}` et `{NiceMatrix}`.

En fait, cette commande n’est disponible que si l’extension `enumitem` a été chargée (avant ou après `nicematrix`). Les notes sont en effet composées en fin de tableau selon un type de liste défini par l’extension `enumitem`.

```
\begin{NiceTabular}{@{}llr@{}}
\toprule \RowStyle{\bfseries}
Nom & Prénom & Date de naissance \\
\midrule
Achard & \tabularnote{La famille Achard est une très ancienne famille du Poitou.} & \\
& Jacques & 5 juin 1962 \\
Lefebvre & \tabularnote{Le patronyme Lefebvre est une altération de Lefébure.} & \\
& Mathilde & 23 mai 1988 \\
Vanesse & Stéphanie & 30 octobre 1994 \\
Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```

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

^a La famille Achard est une très ancienne famille du Poitou.

^b Le patronyme Lefebvre est une altération de Lefébure.

— La commande `\tabularnote` est en fait utilisable avant l’environnement de `nicematrix`, le but étant de pouvoir l’utiliser sur le titre inséré par `\caption` dans un environnement `{table}` de LaTeX.

- Si plusieurs commandes `\tabularnote{...}` se suivent *sans aucun espace entre elles*, les appels de notes correspondants sont composés ensemble, séparés par une virgule (comme avec l’option `multiple` de `footmisc` pour les notes de pied de page).
- Si une commande `\tabularnote{...}` se trouve exactement à la fin d’une case (sans aucun espace après), l’appel de note est composé en débordement vers la droite (cela peut permettre de mieux conserver l’alignement des contenus d’une colonne).
- Si la clé `notes/para` est utilisée, les notes sont composées à la fin du tableau en un seul paragraphe.
- Il existe une clé `tabularnote` qui permet d’insérer du texte dans la zone des notes avant les notes numérotées.
- Si l’extension `booktabs` a été chargée (avant ou après `nicematrix`), la clé `notes/bottomrule` permet de faire tracer un `\bottomrule` de `booktabs` *après* les notes.
- **Nouveau 6.8** Lorsque plusieurs commandes `\tabularnote` sont utilisées avec le même argument, une seule note est insérée en fin de tableau (mais tous les labels sont marqués). Il est possible de désactiver cette fonctionnalité avec la clé `notes/detect-duplicates`
- Il est possible de référencer une note de tableau (avec la commande `\label` placée après le `\tabularnote`).

Voir sur la table 1, p. 34, certaines de ces remarques illustrées. Cette table a été composée avec le code suivant.

```
\begin{table}
\setlength{\belowcaptionskip}{1ex}
\centering
\caption{Utilisation de \texttt{\textbackslash
    tabularnote}\tabularnote{On peut mettre une note dans le titre.}}
\label{t:tabularnote}
\begin{NiceTabular}{@{}llc@{}}%
    [notes/bottomrule, tabularnote = Un peu de texte avant les notes.]
\toprule
Nom & Prénom & Durée de vie \\
\midrule
Barrère & Bertrand & 86\\
Nightingale\tabularnote{Souvent considérée comme la première
    infirmière.}\tabularnote{Surnommée «la Dame à la Lampe».}
& Florence\tabularnote{Cette note est commune à deux appels de notes.} & 90 \\
Schœlcher & Victor & 89\tabularnote{L'appel de note déborde à droite.}\\
Touchet & Marie\tabularnote{Cette note est commune à deux appels de notes.} & 89 \\
Wallis & John & 87 \\
\bottomrule
\end{NiceTabular}
\end{table}
```

12.3 Personnalisation des notes de tableau

Les notes de tableau peuvent être personnalisées grâce à un ensemble de clés disponibles dans `\NiceMatrixOptions`. Ces clés ont un nom préfixé par `notes` :

- `notes/para`
- `notes/bottomrule`
- `notes/style`
- `notes/label-in-tabular`
- `notes/label-in-list`
- `notes/enumitem-keys`
- `notes/enumitem-keys-para`
- `notes/code-before`

— `notes/detect-duplicates`

Pour la commodité, il est aussi possible de fixer ces clés dans `\NiceMatrixOptions` via une clé `notes` qui prend en argument une liste de paires *clé=valeur* où le nom des clés n'a plus à être préfixé par `notes` :

```
\NiceMatrixOptions
{
  notes =
  {
    bottomrule ,
    style = ... ,
    label-in-tabular = ... ,
    enumitem-keys =
    {
      labelsep = ... ,
      align = ... ,
      ...
    }
  }
}
```

On détaille maintenant ces clés.

- La clé `notes/para` demande la composition des notes en fin de tableau en un seul paragraphe.
Valeur initiale : `false`
Cette clé est également accessible dans un environnement individuel.
- La clé `notes/bottomrule` permet de faire tracer un `\bottomrule` de `booktabs` *après* les notes. Ce trait n'est tracé que s'il y a effectivement des notes dans le tableau. L'extension `booktabs` doit avoir été chargée (avant ou après l'extension `nicematrix`). Dans le cas contraire, une erreur est générée.
Valeur initiale : `false`
Cette clé est également accessible dans un environnement individuel.
- La clé `notes/style` est une commande dont l'argument est spécifié par `#1` et qui indique le style de numérotation des notes. C'est ce style qui est utilisé par `\ref` pour faire référence à une note de tableau pour laquelle on a utilisé un `\label`. Ce sont les labels mis en forme avec ce style qui sont séparés par des virgules quand on utilise plusieurs commandes `\tabularnote` successivement. Le marqueur `#1` est censé correspondre à un nom de compteur LaTeX.
Valeur initiale : `\textit{\alph{#1}}`
Une autre valeur possible pourrait être tout simplement `\arabic{#1}`

TABLE 1 – Utilisation de `\tabularnote`^a

Nom	Prénom	Durée de vie
Barrère	Bertrand	86
Nightingale ^{b,c}	Florence ^d	90
Schoelcher	Victor	89 ^e
Touchet	Marie ^d	89
Wallis	John	87

Un peu de texte avant les notes.

^a On peut mettre une note dans le titre.

^b Souvent considérée comme la première infirmière.

^c Surnommée « la Dame à la Lampe ».

^d Cette note est commune à deux appels de notes.

^e L'appel de note déborde à droite.

- La clé `notes/label-in-tabular` est une commande dont l'argument est spécifié par `#1` et qui sert au formatage de l'appel de note dans le tableau. En interne, le numéro de note a déjà été formaté par `notes/style` avant d'être passé en argument à cette commande.

Valeur initiale : `#1`

Pour la composition du français, il est de tradition de mettre un petit espace avant l'appel de note. On peut faire ce réglage de la manière suivante :

```
\NiceMatrixOptions{notes/label-in-tabular = \,\textsuperscript{#1}}
```

- La clé `notes/label-in-list` est une commande dont l'argument est spécifié par `#1` et qui sert au formatage du numéro de note dans la liste des notes en fin de tableau. En interne, le numéro de note a déjà été formaté par `notes/style` avant d'être passé en argument à cette commande.

Valeur initiale : `#1`

Pour la composition du français, on ne compose pas les labels des notes en lettres supérieures dans la liste des notes. On pourra donc prendre le réglage suivant :

```
\NiceMatrixOptions{notes/label-in-list = #1.\nobreak\hspace{0.25em}}
```

La commande `\nobreak` est pour le cas où l'option `para` est utilisée.

- Les notes sont composées en fin de tableau en utilisant en interne un style de liste de `enumitem`. Ce style de liste est défini de la manière suivante (avec, bien sûr, des clés de `enumitem`) :

```
noitemsep , leftmargin = * , align = left , labelsep = Opt
```

La spécification `align = left` de ce style demande que le label de la note soit composé à gauche dans la boîte qui lui est dévolue. Ce réglage a l'avantage d'avoir les notes calées à gauche, ce qui est plaisant si on compose des tableaux dans l'esprit de `booktabs` (voir par exemple la table 1, p. 34).

La clé `notes/enumitem-keys` fournie par `nicematrix` permet modifier ce type de liste de `enumitem` (en utilisant en interne la commande `\setlist*` de `enumitem`).

- La clé `notes/enumitem-keys-para` est similaire à la précédente mais elle est utilisée pour le type de liste qui sera utilisé quand l'option `para` est choisie. Bien entendu, quand cette option `para` est active, c'est une liste de type `inline` (suivant le vocabulaire de `enumitem`) qui est utilisée et les paires `clé=valeur` doivent donc correspondre à une telle liste de type `inline`.

Initialement, le style de liste utilisé est défini par :

```
afterlabel = \nobreak, itemjoin = \quad
```

- La clé `notes/code-before` est une liste de tokens qui seront insérés avant la composition de la liste de notes.

Valeur initiale : *vide*

Si on souhaite, par exemple, que les notes soient composées en gris et en `\footnotesize`, c'est cette clé qu'il faut utiliser.

```
\NiceMatrixOptions{notes/code-before = \footnotesize \color{gray}}
```

On peut aussi mettre dans cette clé `\raggedright` ou `\RaggedRight` (cette dernière est une commande de `ragged2e`).

- La clé `notes/detect-duplicates` active la détection des commandes `\tabularnote` avec le même argument.

Valeur initiale : `true`

Pour un exemple de personnalisation des notes de tableau, voir p. 47.

12.4 Utilisation de `{NiceTabular}` avec `threeparttable`

Si vous souhaitez utiliser les environnements `{NiceTabular}`, `{NiceTabular*}` ou `{NiceTabularX}` dans un environnement `{threeparttable}` de l'extension éponyme, vous devez patcher l'environnement `{threeparttable}` avec le code suivant (avec une version de LaTeX postérieure à 2020/10/01).

```

\makeatletter
\AddToHook{env/threeparttable/begin}
  {\TPT@hookin{NiceTabular}\TPT@hookin{NiceTabular*}\TPT@hookin{NiceTabularX}}
\makeatother

```

13 Autres fonctionnalités

13.1 Commande `\ShowCellNames`

Nouveau 6.9 La commande `\ShowCellNames`, utilisable dans le `\CodeBefore` et le `\CodeAfter` affiche le nom (sous la forme $i-j$) de chaque case.

```

\begin{NiceTabular}{ccc}[hvlines,cell-space-limits=3pt]
\CodeBefore
  \ShowCellNames
\Body
  \Block{2-2}{} & & test \\
  & & blabla \\
  & some text & nothing
\end{NiceTabular}

```

1-1	1-2	test
2-1	2-2	blabla
3-1	some text	nothing

13.2 Utilisation du type de colonne S de siunitx

Si l'extension `siunitx` est chargée (avant ou après `nicematrix`), il est possible d'utiliser les colonnes de type S de `siunitx` dans les environnements de `nicematrix`. L'implémentation n'utilise explicitement aucune macro privée de `siunitx`.

```

$\begin{pNiceArray}{ScW{c}{1cm}c}[nullify-dots,first-row]
{C_1} & \Cdots & & C_n \\
2.3 & 0 & \Cdots & 0 \\
12.4 & \Vdots & & \Vdots \\
1.45 & \\
7.2 & 0 & \Cdots & 0
\end{pNiceArray}$

```

$$\begin{pmatrix} C_1 & \dots & C_n \\ 2.3 & 0 & \dots & 0 \\ 12.4 & \vdots & & \vdots \\ 1.45 & \vdots & & \vdots \\ 7.2 & 0 & \dots & 0 \end{pmatrix}$$

En revanche, les colonnes d de l'extension `dcolumn` ne sont pas prises en charge par `nicematrix`.

13.3 Colonne par défaut dans `{NiceMatrix}`

Nouveau 6.11 Les environnements sans préambule (`{NiceMatrix}`, `{pNiceMatrix}`, etc.) ainsi que la commande `\pAutoNiceMatrix` et ses variantes, acceptent la clé `columns-type` qui indique le type des colonne qui sera utilisé.

Les clés `l` et `r` sont des raccourcis pour `columns-type=l` et `columns-type=r`.

```

$\begin{bNiceMatrix}[r]
\cos x & - \sin x \\
\sin x & \cos x
\end{bNiceMatrix}$

```

$$\begin{bmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{bmatrix}$$

La clé `columns-type` peut être utilisée dans `\NiceMatrixOptions` à condition de la préfixer par `matrix`, ce qui fait que son nom devient `matrix/columns-type`.

13.4 La commande `\rotate`

Utilisée au début d'une case, la commande `\rotate` (fournie par `nicematrix`) compose le contenu après une rotation de 90° dans le sens direct.

Dans l'exemple suivant, on l'utilise dans le `code-for-first-row`.⁴¹

```
\NiceMatrixOptions%
{code-for-first-row = \scriptstyle \rotate \text{image de },
code-for-last-col = \scriptstyle }
$A = \begin{pNiceMatrix}[first-row,last-col=4]
e_1 & e_2 & e_3 & \\
1 & 2 & 3 & e_1 \\
4 & 5 & 6 & e_2 \\
7 & 8 & 9 & e_3
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{matrix} e_1 \\ e_2 \\ e_3 \end{matrix}$$

Si la commande `\rotate` est utilisée dans la « dernière rangée » (extérieure à la matrice), les éléments qui subissent cette rotation sont alignés vers le haut.

```
\NiceMatrixOptions%
{code-for-last-row = \scriptstyle \rotate ,
code-for-last-col = \scriptstyle }
$A = \begin{pNiceMatrix}[last-row,last-col=4]
1 & 2 & 3 & e_1 \\
4 & 5 & 6 & e_2 \\
7 & 8 & 9 & e_3 \\
\text{image de } e_1 & e_2 & e_3
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{matrix} e_1 \\ e_2 \\ e_3 \end{matrix}$$

13.5 L'option `small`

Avec l'option `small`, les environnements de l'extension `nicematrix` sont composés d'une manière proche de ce que propose l'environnement `{smallmatrix}` de l'`amsmath` (et les environnements `{psmallmatrix}`, `{bsmallmatrix}`, etc. de `mathtools`).

```
$\begin{bNiceArray}{cccc|c}[small,
last-col,
code-for-last-col = \scriptscriptstyle,
columns-width = 3mm ]
1 & -2 & 3 & 4 & 5 \\
0 & 3 & 2 & 1 & 2 & L_2 - L_1 \\
0 & 1 & 1 & 2 & 3 & L_1 + L_3
\end{bNiceArray}$
```

$$\left[\begin{array}{cccc|c} 1 & -2 & 3 & 4 & 5 \\ 0 & 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 2 & 3 \end{array} \right] \begin{matrix} \\ L_2 \leftarrow 2L_1 - L_2 \\ L_3 \leftarrow L_1 + L_3 \end{matrix}$$

On remarquera néanmoins que l'environnement `{NiceMatrix}` avec l'option `small` ne prétend pas être composé exactement comme l'environnement `{smallmatrix}`. C'est que les environnements de `nicematrix` sont tous fondés sur `{array}` (de `array`) alors que ce n'est pas le cas de `{smallmatrix}` (fondé directement sur un `\halign` de TeX).

En fait, l'option `small` correspond aux réglages suivants :

- les composantes du tableau sont composées en `\scriptstyle` ;
- `\arraystretch` est fixé à 0.47 ;
- `\arraycolsep` est fixé à 1.45 pt ;
- les caractéristiques des lignes en pointillés sont également modifiées.

41. On peut aussi l'utiliser dans `\RowStyle` (cf. p. 18).

13.6 Les compteurs iRow et jCol

Dans les cases du tableau, il est possible d'utiliser les compteurs LaTeX `iRow` et `jCol` qui représentent le numéro de la rangée courante et le numéro de la colonne courante⁴². Bien entendu, l'utilisateur ne doit pas modifier les valeurs de ces compteurs qui sont utilisés en interne par `nicematrix`.

Dans le `\CodeBefore` (cf. p. 14) et dans le `\CodeAfter` (cf. p. 28), `iRow` représente le nombre total de rangées (hors éventuelles rangées extérieures) et `jCol` le nombre total de colonnes (hors potentielles colonnes extérieures).

```

 $\begin{pNiceMatrix}%
  [first-row,
  first-col,
  code-for-first-row = \mathbf{\alpha{jCol}} ,
  code-for-first-col = \mathbf{\arabic{iRow}} ]
& & & & \\
& 1 & 2 & 3 & 4 \\
& 5 & 6 & 7 & 8 \\
& 9 & 10 & 11 & 12
\end{pNiceMatrix}$ 

```

$$\begin{matrix} & \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ \mathbf{1} & 1 & 2 & 3 & 4 \\ \mathbf{2} & 5 & 6 & 7 & 8 \\ \mathbf{3} & 9 & 10 & 11 & 12 \end{matrix}$$

Si des compteurs LaTeX nommés `iRow` ou `jCol` sont créés dans le document par d'autres extensions que `nicematrix` (ou tout simplement par l'utilisateur final), ces compteurs sont masqués dans les environnements de `nicematrix`.

L'extension `nicematrix` propose aussi des commandes pour composer automatiquement des matrices à partir d'un motif général. Ces commandes sont nommées `\AutoNiceMatrix`, `\pAutoNiceMatrix`, `\bAutoNiceMatrix`, `\vAutoNiceMatrix`, `\VAutoNiceMatrix` et `\BAutoNiceMatrix`.

Chacune de ces commandes prend deux arguments obligatoires : le premier est la taille de la matrice, sous la forme $n \times p$, où n est le nombre de rangées et p est le nombre de colonnes et le deuxième est le motif (c'est-à-dire simplement des tokens qui seront insérés dans chaque case de la matrice).

```

 $C = \pAutoNiceMatrix{3-3}{C_{\arabic{iRow},\arabic{jCol}}}$ 

```

$$C = \begin{pmatrix} C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

13.7 L'option light-syntax

L'option `light-syntax` (inspirée de l'extension `spalign`) permet d'alléger la saisie des matrices, ainsi que leur lisibilité dans le source TeX. Lorsque cette option est activée, on doit utiliser le point-virgule comme marqueur de fin de rangée et séparer les colonnes par des espaces ou des tabulations. On remarquera toutefois que, comme souvent dans le monde TeX, les espaces après les séquences de contrôle ne sont pas comptées et que les éléments entre accolades sont considérés comme un tout.

```

 $\begin{bNiceMatrix}[light-syntax,first-row,first-col]
{} a & b ;
a \ 2\cos a & {\cos a + \cos b} ;
b \cos a + \cos b & { 2 \cos b }
\end{bNiceMatrix}$ 

```

$$\begin{matrix} & a & b \\ a & 2 \cos a & \cos a + \cos b \\ b & \cos a + \cos b & 2 \cos b \end{matrix}$$

On peut changer le caractère utilisé pour indiquer les fins de rangées avec l'option `end-of-row`. Comme dit précédemment, la valeur initiale de ce paramètre est un point-virgule.

Lorsque l'option `light-syntax` est utilisée, il n'est pas possible de mettre d'éléments en verbatim (avec par exemple la commande `\verb`) dans les cases du tableau.⁴³

42. On rappelle que le numéro de la « première rangée » (si elle existe) est 0 et que le numéro de la « première colonne » (si elle existe) est 0 également.

43. La raison en est que lorsque l'option `light-syntax` est utilisée, le contenu complet de l'environnement est chargé comme un argument de commande TeX. L'environnement ne se comporte plus comme un « vrai » environnement de LaTeX qui se contente d'insérer des commandes avant et après.

13.8 Couleur des délimiteurs

Pour les environnements avec délimiteurs (`{pNiceArray}`, `{pNiceMatrix}`, etc.), il est possible de changer la couleur des délimiteurs avec la clé `delimiters/color`.

```

\begin{bNiceMatrix}[delimiters/color=red]
1 & 2 & \\
3 & 4 & 
\end{bNiceMatrix}
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Cette couleur s'applique aussi aux délimiteurs tracés par `\SubMatrix` (cf. p. 29).

13.9 L'environnement `{NiceArrayWithDelims}`

En fait, l'environnement `{pNiceArray}` et ses variantes sont fondés sur un environnement plus général, appelé `{NiceArrayWithDelims}`. Les deux premiers arguments obligatoires de cet environnement sont les délimiteurs gauche et droit qui seront utilisés dans la construction de la matrice. Il est possible d'utiliser `{NiceArrayWithDelims}` si on a besoin de délimiteurs atypiques ou asymétriques.

```

\begin{NiceArrayWithDelims}
{\downarrow}{\uparrow}{ccc}[margin]
1 & 2 & 3 & \\
4 & 5 & 6 & \\
7 & 8 & 9 & 
\end{NiceArrayWithDelims}
```

$$\begin{array}{|ccc|} \downarrow & 1 & 2 & 3 & \uparrow \\ & 4 & 5 & 6 & \\ & 7 & 8 & 9 & \end{array}$$

13.10 La commande `\OnlyMainNiceMatrix`

La commande `\OnlyMainNiceMatrix` n'exécute son argument que si on se trouve dans le tableau principal, c'est-à-dire que l'on n'est pas dans les rangées extérieures. Si elle est utilisée hors d'un environnement de `nicematrix`, elle est sans effet.

Pour un exemple d'utilisation, voir tex.stackexchange.com/questions/488566

14 Utilisation de Tikz avec `nicematrix`

14.1 Les nœuds correspondant aux contenus des cases

L'extension `nicematrix` crée un nœud PGF-Tikz pour chaque case non vide du tableau considéré. Ces nœuds sont utilisés, entre autres, pour tracer les lignes en pointillés entre les cases du tableau.

Attention : Par défaut, aucun nœud n'est créé dans une case vide.

Néanmoins, on peut forcer la création d'un nœud avec la commande `\NotEmpty`.⁴⁴

Tous les nœuds du document doivent avoir des noms deux à deux distincts et le nom de ces nœuds doit donc faire intervenir le numéro de l'environnement courant. Les environnements créés par `nicematrix` sont en effet numérotés par un compteur global interne.

Si l'environnement concerné a le numéro n , alors le nœud de la rangée i et de la colonne j a pour nom `nm-n-i-j`.

La commande `\NiceMatrixLastEnv` donne le numéro du dernier de ces environnements (pour LaTeX, il s'agit d'une commande — complètement développable — et non d'un compteur).

Il est néanmoins recommandé de passer plutôt par la clé `name`. Celle-ci permet de donner un nom à l'environnement. Une fois l'environnement nommé, les nœuds sont accessibles à travers les noms « `nom-i-j` » où `nom` est le nom donné au tableau et i et j les numéros de rangée et de colonne de la case considérée. On peut les utiliser avec PGF mais l'utilisateur final préférera sans doute utiliser Tikz

⁴⁴ Il faut toutefois remarquer qu'avec cette commande, la case est considérée comme non vide, ce qui a des conséquences sur le tracé des lignes pointillées (cf. p. 23) et la détermination des « coins » (cf. p. 10).

(qui est une sur-couche de PGF). Il faut néanmoins se souvenir que `nicematrix` ne charge pas Tikz par défaut. Dans les exemples qui suivent, on suppose que Tikz a été chargé par l'utilisateur.

```


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


```

Ne pas oublier les options `remember picture` et `overlay`.

Dans le `\CodeAfter`, et si Tikz est chargé, les choses sont plus simples. On peut (et on doit) désigner les nœuds sous la forme $i-j$: il n'y a pas à préciser l'environnement qui est évidemment l'environnement courant.

```


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


```

Dans l'exemple suivant, nous avons surligné tous les nœuds de la matrice (on explique plus loin la technique utilisée : cf. p. 54).

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix}$$

Les nœuds de la dernière colonne (hors éventuelle « colonne extérieure » spécifiée par `last-col`) peuvent aussi être désignés par $i\text{-last}$. De même, les nœuds de la dernière ligne peuvent être désignés par $\text{last-}j$.

14.1.1 Les colonnes V de `varwidth`

Quand l'extension `varwidth` est chargée, les colonnes de type V définies par `varwidth` sont prises en charge par `nicematrix`. Il peut être intéressant de préciser que, pour une cellule située dans une colonne de type V, le nœud PGF/Tikz créé par `nicematrix` pour le contenu de cette cellule a une largeur ajustée au contenu de cette cellule. Cela est en contraste avec le cas des colonnes de type p, m ou b dans lesquelles les nœuds ont toujours une largeur égale à la largeur de la colonne. Dans l'exemple suivant, la commande `\lipsum` est fournie par l'extension éponyme.

```


```

Titre

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

On a utilisé les nœuds indiquant la position des filets, qui sont présentés un peu plus loin, p. 42.

14.2 Les « nœuds moyens » et les « nœuds larges »

En fait, l'extension `nicematrix` peut créer deux séries de nœuds supplémentaires (*extra nodes* en anglais) : les « nœuds moyens » (*medium nodes* en anglais) et les « nœuds larges » (*large nodes* en anglais). Les premiers sont créés avec l'option `create-medium-nodes` et les seconds avec l'option `create-large-nodes`.⁴⁵

Ces nœuds ne sont pas utilisés par défaut par `nicematrix`.

Les noms des « nœuds moyens » s'obtiennent en ajoutant le suffixe « `-medium` » au nom des nœuds normaux. Dans l'exemple suivant, on a surligné tous les « nœuds moyens ». Nous considérons que cet exemple se suffit à lui-même comme définition de ces nœuds.

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix}$$

Les noms des « nœuds larges » s'obtiennent en ajoutant le suffixe « `-large` » au nom des nœuds normaux. Dans l'exemple suivant, on a surligné tous les « nœuds larges ». Nous considérons que cet exemple se suffit à lui-même comme définition de ces nœuds.⁴⁶

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix}$$

Les « nœuds larges » de la première colonne et de la dernière colonne peuvent apparaître trop petits pour certains usages. C'est pourquoi il est possible d'utiliser les options `left-margin` et `right-margin` pour ajouter de l'espace des deux côtés du tableau et aussi de l'espace dans les « nœuds larges » de la première colonne et de la dernière colonne. Dans l'exemple suivant, nous avons utilisé les options `left-margin` et `right-margin`.⁴⁷

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix}$$

Il est aussi possible d'ajouter de l'espace sur les côtés du tableau avec les options `extra-left-margin` et `extra-right-margin`. Ces marges ne sont pas incorporées dans les « nœuds larges ». Dans l'exemple suivant, nous avons utilisé `extra-left-margin` et `extra-right-margin` avec la valeur 3 pt.

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix}$$

Attention : Ces nœuds sont reconstruits à partir des contenus des cases et ne correspondent donc pas nécessairement aux cases délimitées par des filets.

Voici un tableau qui a été composé de la manière suivante :

```
\large
\begin{NiceTabular}{wl{2cm}ll}[hvlines]
fraise & amande & abricot \\
prune & pêche & poire \\
noix & noisette & brugnon
\end{NiceTabular}
```

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

45. Il existe aussi l'option `create-extra-nodes` qui est un alias pour la conjonction de `create-medium-nodes` et `create-large-nodes`.

46. Il n'y a pas de « nœuds larges » créés dans les rangées et colonnes extérieures (pour ces rangées et colonnes, voir p. 21).

47. Les options `left-margin` et `right-margin` prennent des dimensions comme valeurs mais, si aucune valeur n'est donnée, c'est la valeur par défaut qui est utilisée et elle est égale à `\arraycolsep` (valeur initiale : 5 pt). Il existe aussi une option `margin` pour fixer à la fois `left-margin` et `right-margin`.

Ci-contre, on a colorié toutes les cases de ce tableau avec `\chessboardcolors` (cf. p. 15).

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Voici maintenant tous les « nœuds larges » de ce tableau (sans utilisation de `margin` ni de `extra-margin`).

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Les nœuds que l'on vient de décrire ne sont pas accessibles par défaut dans le `\CodeBefore` (décrit p. 14).

On peut rendre ces nœuds accessibles dans le `\CodeBefore` en utilisant la clé `create-cell-nodes` du mot-clé `\CodeBefore` (dans ce cas-là, les nœuds sont créés une première fois avant la construction du tableau en utilisant des informations écrites dans le fichier `aux` puis recréés lors de la composition du tableau proprement dit).

Voici un exemple d'utilisation de ces nœuds dans le `\CodeAfter`.

```
\begin{NiceArray}{c@{\;}c@{\;}c@{\;}c@{\;}c}[create-medium-nodes]
  u_1 & - & u_0 & = & r & \\
  u_2 & - & u_1 & = & r & \\
  u_3 & - & u_2 & = & r & \\
  u_4 & - & u_3 & = & r & \\
  \phantom{u_5} & & \phantom{u_4} & & \smash{\vdots} & \\
  u_n & - & u_{n-1} & = & r & \\
\hline
  u_n & - & u_0 & = & nr & \\
\CodeAfter
  \tikz[very thick, red, opacity=0.4,name suffix = -medium]
  \draw (1-1.north west) -- (2-3.south east)
  (2-1.north west) -- (3-3.south east)
  (3-1.north west) -- (4-3.south east)
  (4-1.north west) -- (5-3.south east)
  (5-1.north west) -- (6-3.south east) ;
\end{NiceArray}
```

$$\begin{array}{rcl}
 u_1 - u_0 & = & r \\
 u_2 - u_1 & = & r \\
 u_3 - u_2 & = & r \\
 u_4 - u_3 & = & r \\
 & \vdots & \\
 u_n - u_{n-1} & = & r \\
 \hline
 u_n - u_0 & = & nr
 \end{array}$$

14.3 Les nœuds indiquant la position des filets

L'extension `nicematrix` crée un nœud PGF-Tikz nommé simplement i (précédé du préfixe habituel) à l'intersection du filet horizontal de numéro i et du filet vertical de numéro i (ou plutôt la position potentielle de ces filets car ils ne sont peut-être pas tracés). Le dernier nœud a aussi un alias nommé simplement `last`. Il existe aussi un nœud nommé $i.5$ à mi-chemin entre le nœud i et le nœud $i + 1$. Ces nœuds sont accessibles dans le `\CodeAfter` mais aussi dans le `\CodeBefore`.

$\bullet^{1.5}$	\bullet^2 tulipe	lys
arum	$\bullet^{2.5}$	\bullet^3 violette mauve
muguet	dahlia	$\bullet^{3.5}$

Si on utilise Tikz (on rappelle que `nicematrix` ne charge pas Tikz mais uniquement PGF qui est une sous-couche de Tikz), on peut donc accéder (dans le `\CodeAfter` mais aussi dans le `\CodeBefore`) à l'intersection du filet horizontal i et du filet vertical j avec la syntaxe $(i-|j)$.

```
\begin{NiceMatrix}
\CodeBefore
\tikz \draw [fill=red!15] (7-|4) |- (8-|5) |- (9-|6) |- cycle ;
\Body
1 \\\
1 & 1 \\\
1 & 2 & 1 \\\
1 & 3 & 3 & 1 \\\
1 & 4 & 6 & 4 & 1 \\\
1 & 5 & 10 & 10 & 5 & 1 \\\
1 & 6 & 15 & 20 & 15 & 6 & 1 \\\
1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 \\\
1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1
\end{NiceMatrix}
```

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1

```

Les nœuds de la forme $i.5$ peuvent être utilisés par exemple pour barrer une ligne (si on a chargé Tikz).

```
\begin{pNiceArray}{ccc|c}
2 & 1 & 3 & 0 \\\
3 & 3 & 1 & 0 \\\
3 & 3 & 1 & 0
\CodeAfter
\tikz \draw [red] (3.5-|1) -- (3.5-|last) ;
\end{pNiceArray}
```

$$\begin{pmatrix} 2 & 1 & 3 & | & 0 \\ 3 & 3 & 1 & | & 0 \\ \hline 3 & 3 & 1 & | & 0 \end{pmatrix}$$

14.4 Les nœuds correspondants aux commandes `\SubMatrix`

La commande `\SubMatrix` disponible dans le `\CodeAfter` a été présentée p. 29.

Si une commande `\SubMatrix` est utilisée avec la clé `name` sous la forme `name=MonNom`, trois nœuds PGF/Tikz sont créés avec les noms `MonNom-left`, `MonNom` et `MonNom-right`.

Les nœuds `MonNom-left` et `MonNom-right` correspondent aux délimiteurs gauche et droit et le nœud `MonNom` correspond à la sous-matrice elle-même.

Dans l'exemple suivant, on a surligné ces trois nœuds (la sous-matrice elle-même a été créée avec `\SubMatrix{\{2-2\}{3-3}\}`).

$$\begin{pmatrix} 121 & 23 & 345 & 345 \\ 45 & \left\{ \begin{array}{cc} 346 & 863 \\ 38458 & 34 \end{array} \right\} & 444 \\ 3462 & 7 & 78 & 309 \end{pmatrix}$$

15 API pour les développeurs

L’extension `nicematrix` fournit deux variables internes mais publiques⁴⁸ :

- `\g_nicematrix_code_before_tl` ;
- `\g_nicematrix_code_after_tl`.

Ces variables constituent le code du « `code-before` » (que l’on rentre souvent avec la syntaxe utilisant `\CodeBefore` et `\Body` en début d’environnement) et du « `code-after` » (que l’on rentre souvent en fin d’environnement après le mot-clé `\CodeAfter`). Le développeur peut donc les utiliser pour y ajouter du code à partir d’une case du tableau (l’affectation devra être globale, ce qui permettra de sortir de la case, qui est un groupe au sens de TeX).

On remarquera que l’utilisation de `\g_nicematrix_code_before_tl` nécessite une compilation supplémentaire (car les instructions sont écrites dans le fichier `aux` pour être utilisées à la compilation suivante).

Exemple : On souhaite écrire une commande `\crossbox` qui barre en croix la cellule courante. Cette commande prendra en argument optionnel une liste de couples *clé-valeur* qui sera passée à Tikz avant que la croix ne soit tracée.


On peut alors programmer cette commande `\crossbox` de la manière suivante, qui utilise explicitement la variable publique `\g_nicematrix_code_after_tl`.

```
\ExplSyntaxOn
\cs_new_protected:Nn \__pantigny_crossbox:nnn
{
  \tikz \draw [ #3 ]
    ( #1 -| \int_eval:n { #2 + 1 } ) -- ( \int_eval:n { #1 + 1 } -| #2 )
    ( #1 -| #2 ) -- ( \int_eval:n { #1 + 1 } -| \int_eval:n { #2 + 1 } ) ;
}

\NewDocumentCommand \crossbox { ! 0 { } }
{
  \tl_gput_right:Nx \g_nicematrix_code_after_tl
  {
    \__pantigny_crossbox:nnn
    { \int_use:c { c@iRow } }
    { \int_use:c { c@jCol } }
    { \exp_not:n { #1 } }
  }
}
\ExplSyntaxOff
```

Voici un exemple d’utilisation :

```
\begin{NiceTabular}{ccc}[hvlines]
merlan & requin & cabillaud \\
baleine & \crossbox[red] & morue \\
mante & raie & poule
\end{NiceTabular}
```

merlan	requin	cabillaud
baleine		morue
mante	raie	poule

16 Remarques techniques

Première remarque : l’extension `nicematrix` doit être chargée après l’extension `underscore`. Si ce n’est pas le cas, une erreur est levée.

48. Conformément aux conventions de LaTeX3, toute variable dont le nom commence par `\g_nicematrix` ou `\l_nicematrix` est publique alors que toute variable dont le nom débute par `\g__nicematrix` ou par `\l__nicematrix` est privée.

16.1 Lignes diagonales

Par défaut, toutes les lignes diagonales⁴⁹ d'un même tableau sont « parallélisées ». Cela signifie que la première diagonale est tracée et que, ensuite, les autres lignes sont tracées parallèlement à la première (par rotation autour de l'extrémité la plus à gauche de la ligne). C'est pourquoi la position des instructions `\Ddots` dans un tableau peut avoir un effet marqué sur le résultat final.

Dans les exemples suivants, la première instruction `\Ddots` est marquée en couleur :

Exemple avec parallélisation (comportement par défaut) :

```
$A = \begin{pNiceMatrix}
1      & \Cdots & & & 1      & \\
a+b    & \Ddots & & & \Vdots & \\
\Vdots & \Ddots & & & & \\
a+b    & \Cdots & a+b & & 1      & \\
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ a+b & \ddots & & & \vdots \\ \vdots & \ddots & & & \vdots \\ a+b & \cdots & a+b & & 1 \end{pmatrix}$$

```
$A = \begin{pNiceMatrix}
1      & \Cdots & & & 1      & \\
a+b    & & & & \Vdots & \\
\Vdots & \Ddots & \Ddots & & & \\
a+b    & \Cdots & a+b & & 1      & \\
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ a+b & & & & \vdots \\ \vdots & \ddots & \ddots & & \vdots \\ a+b & \cdots & a+b & & 1 \end{pmatrix}$$

Il est possible de désactiver la parallélisation avec l'option `parallelize-diags` mise à `false` :

Le même exemple sans parallélisation :

$$A = \begin{pmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ a+b & \ddots & & & \vdots \\ \vdots & \ddots & & & \vdots \\ a+b & \cdots & a+b & & 1 \end{pmatrix}$$

On peut choisir l'instruction `\Ddots` qui sera tracée en premier (et qui servira pour tracer les suivantes quand la parallélisation est activée) avec la clé `draw-first` : `\Ddots[draw-first]`.

16.2 Les cases « vides »

Une instruction comme `\Ldots`, `\Cdots`, etc. essaye de déterminer la première case vide de part et d'autre de la case considérée. Quand la clé `corners` (cf. p. 10) est utilisée, les coins qui sont déterminés sont composés de cases vides. Néanmoins, une case « vide » n'est pas nécessairement sans contenu dans le codage TeX (c'est-à-dire sans aucun token entre les deux esperluettes `&`). Les règles précises sont les suivantes :

- Une case implicite est vide. Par exemple, dans la matrice suivante

```
\begin{pmatrix}
a & b \\
c & \\
\end{pmatrix}
```

la dernière case (deuxième rangée et deuxième colonne) est vide.

- Chaque case avec un rendu par TeX de largeur nulle est vide.
- Une case qui contient la commande `\NotEmpty` est non vide (et un nœud PGF/Tikz est créé pour cette case).

49. On parle des lignes créées par `\Ddots` et non des lignes créées par une commande `\line` dans le `\CodeAfter`.

- Une case avec une commande `\Hspace` (ou `\Hspace*`) est vide. Cette commande `\Hspace` est une commande définie par l’extension `nicematrix` avec la même signification que `\hspace` excepté que la case où cette commande est utilisée est considérée comme vide. Cette commande peut être utilisée pour fixer la largeur des colonnes sans interférer avec le tracé des lignes en pointillés par `nicematrix`.
- Une case d’une colonne de type `p`, `m` ou `t` est toujours non vide (*attention* : ne pas se fier à ce point car il est susceptible de changer). En revanche, une case d’une colonne de type `V` de `varwidth` (cf. p. 20) est vide quand son contenu TeX est de largeur nulle.

16.3 L’option `exterior-arraycolsep`

L’environnement `{array}` insère un espace horizontal égal à `\arraycolsep` avant et après chaque colonne. En particulier, il y a un espace égal à `\arraycolsep` avant et après le tableau. Cette caractéristique de l’environnement `{array}` n’était probablement pas une bonne idée⁵⁰. L’environnement `{matrix}` et ses variantes (`{pmatrix}`, `{vmatrix}`, etc.) de `amsmath` préfèrent supprimer ces espaces avec des instructions explicites `\hskip -\arraycolsep`⁵¹. L’extension `nicematrix` fait de même dans tous ses environnements y compris l’environnement `{NiceArray}`. Néanmoins, si l’utilisateur souhaite que l’environnement `{NiceArray}` se comporte par défaut comme l’environnement `{array}` (par exemple pour faciliter l’adaptation d’un document existant), il peut contrôler ce comportement avec l’option `exterior-arraycolsep` accessible via la commande `\NiceMatrixOptions`. Avec cette option, des espaces extérieurs de longueur `\arraycolsep` seront insérés dans les environnements `{NiceArray}` (les autres environnements de l’extension `nicematrix` ne sont pas affectés).

16.4 Incompatibilités

L’extension `nicematrix` n’est pas compatible avec la classe `ieeeaccess` (car cette classe n’est pas compatible avec PGF/Tikz).⁵²

Pour pouvoir utiliser `nicematrix` avec la classe `aastex631`, on doit ajouter dans le préambule du fichier les lignes suivantes :

```
\BeforeBegin{NiceTabular}{\let\begin\BeginEnvironment\let\end\EndEnvironment}
\BeforeBegin{NiceArray}{\let\begin\BeginEnvironment}
\BeforeBegin{NiceMatrix}{\let\begin\BeginEnvironment}
```

Pour pouvoir utiliser `nicematrix` avec la classe `sn-jnl`, pgf doit être chargé avant le `\documentclass` :

```
\RequirePackage{pgf}
\documentclass{sn-jnl}
```

L’extension `nicematrix` n’est pas parfaitement compatible avec l’extension `arydshln` (parce que cette extension redéfinit de nombreuses commandes internes de `array`). De toutes manières, dans le cadre de `nicematrix`, il est conseillé de tracer les filets en tiretés avec les commandes de `nicematrix` en créant un style de ligne personnalisée avec `custom-line` : cf. p. 11.

50. Dans la documentation de `{amsmath}`, on peut lire : *The extra space of `\arraycolsep` that `array` adds on each side is a waste so we remove it [in `{matrix}`] (perhaps we should instead remove it from `array` in general, but that’s a harder task).*

51. Et non en insérant `@{}` de part et d’autre du préambule, ce qui fait que la longueur des `\hline` n’est pas modifiée et elle peut paraître trop longue, surtout avec des crochets.

52. Voir <https://tex.stackexchange.com/questions/528975/error-loading-tikz-in-ieeeaccess-class>

17 Exemples

17.1 Utilisation de la clé « tikz » de la commande \Block

La clé `tikz` de la commande `\Block` n'est disponible que lorsque Tikz est chargé.⁵³
 Pour l'exemple suivant, la bibliothèque `patterns` de Tikz doit aussi être chargée.

```
\usetikzlibrary{patterns}

\ttfamily \small
\begin{NiceTabular}{X[m]X[m]X[m]}[hvlines,cell-space-limits=3pt]
  \Block[tikz={pattern=grid,pattern color=lightgray}]{ }
    {pattern = grid,\ \ pattern color = lightgray}
  & \Block[tikz={pattern = north west lines,pattern color=blue}]{ }
    {pattern = north west lines,\ \ pattern color = blue}
  & \Block[tikz={outer color = red!50, inner color=white }]{2-1}
    {outer color = red!50,\ \ inner color = white} \ \
    \Block[tikz={pattern = sixpointed stars, pattern color = blue!15}]{ }
    {pattern = sixpointed stars,\ \ pattern color = blue!15}
  & \Block[tikz={left color = blue!50}]{ }
    {left color = blue!50} \ \
\end{NiceTabular}
```

<pre>pattern = grid, pattern color = lightgray</pre>	<pre>pattern = north west lines, pattern color = blue</pre>	<pre>outer color = red!50, inner color = white</pre>
<pre>pattern = sixpointed stars, pattern color = blue!15</pre>	<pre>left color = blue!50</pre>	

17.2 Notes dans les tableaux

Les outils de `nicematrix` pour les notes dans les tableaux ont été présentés à la partie 12 p. 32.

Imaginons que l'on souhaite numéroter les notes de tableau (celles construites avec `\tabularnote`) avec des astérisques.⁵⁴

On commence par écrire une commande `\stars` similaire aux commandes classiques `\arabic`, `\alph`, `\Alpha`, etc. mais qui produit un nombre d'astérisques égal à son argument.⁵⁵

```
\ExplSyntaxOn
\NewDocumentCommand \stars { m }
{ \prg_replicate:nn { \value { #1 } } { $ \star $ } }
\ExplSyntaxOff
```

Bien entendu, on change le style des notes avec la clé `notes/style`. Mais, il serait bon aussi de changer certains paramètres du type de liste (au sens de `enumitem`) utilisé pour composer les notes après le tableau. On demande de composer les labels avec une largeur égale à celle du plus grand des labels. Or, le label le plus large est bien entendu celui avec le maximum d'astérisques. On connaît ce nombre : il est égal à `\value{tabularnote}` (car `tabularnote` est le compteur LaTeX utilisé par `\tabularnote` et il est donc égal à la fin au nombre total de notes dans le tableau). On utilise alors la clé `widest*` de `enumitem` pour demander une largeur de label correspondante : `widest*=\value{tabularnote}`.

```
\NiceMatrixOptions
{
  notes =
```

53. Par défaut, `nicematrix` ne charge que PGF, qui est une sous-couche de Tikz.

54. Bien entendu, il faut qu'il y en ait très peu : trois paraît un maximum.

55. Ou plutôt : à la valeur de son argument.

Un exemple avec un système linéaire :

```
\begin{pNiceArray}{*6c|c}[nullify-dots,last-col,code-for-last-col=\scriptstyle]
1      & 1 & 1 & \Cdots & 1 & 0 & \\
0      & 1 & 0 & \Cdots & 0 & & L_2 \gets L_2-L_1 \\
0      & 0 & 1 & \Ddots & \Vdots & & L_3 \gets L_3-L_1 \\
      & & & \Ddots & & \Vdots & \Vdots \\
\Vdots & & & \Ddots & 0 & & \\
0      & & & \Cdots & 0 & 1 & 0 & L_n \gets L_n-L_1
\end{pNiceArray}
```

$$\left(\begin{array}{cccccc|c} 1 & 1 & 1 & \cdots & 1 & 0 \\ 0 & 1 & 0 & \cdots & 0 & \\ 0 & 0 & 1 & \cdots & 0 & \\ \vdots & & & \ddots & & \\ 0 & \cdots & \cdots & 0 & 1 & \end{array} \right) \begin{array}{l} \\ L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - L_1 \\ \vdots \\ L_n \leftarrow L_n - L_1 \end{array}$$

17.4 Des lignes pointillées qui ne sont plus pointillées

L'option `line-style` permet de changer le style des lignes tracées par `\Ldots`, `\Cdots`, etc. On peut de ce fait tracer des lignes qui ne sont plus pointillées.

```
\NiceMatrixOptions{code-for-first-row = \scriptstyle,code-for-first-col = \scriptstyle }
\setcounter{MaxMatrixCols}{12}
\newcommand{\blue}{\color{blue}}
\begin{pNiceMatrix}[last-row,last-col,nullify-dots,xdots/line-style={dashed,blue}]
1&&\Vdots&&\Vdots& \\
&\Ddots[line-style=standard]& \\
&&1& \\
&\Cdots[color=blue,line-style=dashed]&&\blue{0}& \\
&\Cdots&&\blue{1}&&\Cdots&\blue{\leftarrow i}& \\
&&&1& \\
&&\Vdots&&\Ddots[line-style=standard]&&\Vdots& \\
&&&&&1& \\
&\Cdots&&\blue{1}&\Cdots&&\Cdots&\blue{0}&&\Cdots&\blue{\leftarrow j}& \\
&&&&&&1& \\
&&&&&&&\Ddots[line-style=standard]& \\
&&&\Vdots&&&&\Vdots&&1& \\
&&&\blue{\overset{\uparrow}{i}}&&&\blue{\overset{\uparrow}{j}}& \\
\end{pNiceMatrix}
```

$$\left(\begin{array}{cccc|cc} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ \hline & & & 0 & & 1 \\ & & & & \ddots & \\ & & & & & 1 \\ \hline & & & 1 & & 0 \\ & & & & \ddots & \\ & & & & & 1 \\ \hline & & & & & & \ddots & \\ & & & & & & & 1 \end{array} \right) \begin{array}{l} \\ \\ \\ \leftarrow i \\ \\ \leftarrow j \end{array}$$

On peut même tracer des lignes continues.⁵⁶

56. Dans ce document, la bibliothèque `arrows.meta` de Tikz a été chargée, ce qui a une incidence sur la forme des

```

\NiceMatrixOptions
{nullify-dots,code-for-first-col = \color{blue},code-for-first-row=\color{blue}}
$\begin{pNiceMatrix}[first-row,first-col]
& & \Ldots[line-style={solid,<->},shorten=0pt]^{n \text{ colonnes}} \\
& 1 & 1 & 1 & \Ldots & 1 \\
& 1 & 1 & 1 & & 1 \\
\vdots[line-style={solid,<->}]_{n \text{ rangées}} \\
& 1 & 1 & 1 & & 1 \\
& 1 & 1 & 1 & & 1 \\
& 1 & 1 & 1 & \Ldots & 1
\end{pNiceMatrix}$

```

$$\begin{array}{c}
\begin{array}{c} \textcolor{blue}{\uparrow} \\ \textcolor{blue}{n \text{ rangées}} \\ \textcolor{blue}{\downarrow} \end{array}
\begin{array}{c} \textcolor{blue}{\leftarrow} \\ \textcolor{blue}{n \text{ colonnes}} \\ \textcolor{blue}{\rightarrow} \end{array}
\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix}
\end{array}$$

17.5 Lignes en tiretés

Dans l'exemple suivant, on utilise des commandes `\Block` pour tracer des filets en tiretés. Cet exemple nécessite que Tikz soit chargé (par `\usepackage{tikz}`).

```

\begin{pNiceMatrix}
\Block[borders={bottom,right,tikz=dashed}]{2-2}{}
1 & 2 & 0 & 0 & 0 & 0 \\
4 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & \Block[borders={bottom,top,right,left,tikz=dashed}]{2-2}{}
& 7 & 1 & 0 & 0 \\
0 & 0 & -1 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & \Block[borders={left,top,tikz=dashed}]{2-2}{}
& 3 & 4 \\
0 & 0 & 0 & 0 & 1 & 4
\end{pNiceMatrix}

```

$$\begin{pmatrix} 1 & 2 & | & 0 & 0 & 0 & 0 \\ 4 & 5 & | & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & | & 7 & 1 & 0 & 0 \\ 0 & 0 & | & -1 & 2 & 0 & 0 \\ 0 & 0 & | & 0 & 0 & 3 & 4 \\ 0 & 0 & | & 0 & 0 & 1 & 4 \end{pmatrix}$$

17.6 Empilements de matrices

On a souvent besoin de présenter des matrices empilées les unes au-dessus des autres (par exemple pour la résolution de systèmes linéaires).

Pour avoir les colonnes alignées les unes sous les autres, on peut imposer une largeur commune à toutes les colonnes, ce que l'on fait dans l'exemple suivant avec l'environnement `{NiceMatrixBlock}` et l'option `auto-columns-width`.

```

\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
{

```

pointes de flèches.

```

    light-syntax,
    last-col, code-for-last-col = \color{blue} \scriptstyle,
  }
\setlength{\extrarowheight}{1mm}

$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 {} ;
3 -18 12 1 4 ;
-3 -46 29 -2 -15 ;
9 10 -5 4 7
\end{pNiceArray}$

\smallskip
$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 ;
0 64 -41 1 19 { L_2 \gets L_1-4L_2 } ;
0 -192 123 -3 -57 { L_3 \gets L_1+4L_3 } ;
0 -64 41 -1 -19 { L_4 \gets 3L_1-4L_4 } ;
\end{pNiceArray}$

\smallskip
$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 ;
0 64 -41 1 19 ;
0 0 0 0 0 { L_3 \gets 3 L_2 + L_3 }
\end{pNiceArray}$

\smallskip
$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 {} ;
0 64 -41 1 19 ;
\end{pNiceArray}$

\end{NiceMatrixBlock}

```

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{pmatrix}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{pmatrix}
 \begin{matrix} \\ L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{matrix}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}
 \begin{matrix} \\ \\ L_3 \leftarrow 3L_2 + L_3 \end{matrix}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{pmatrix}$$

On constate que la dernière matrice n'est pas parfaitement alignée avec les précédentes. C'est que les parenthèses, en LaTeX, n'ont pas toutes la même largeur suivant leur taille.

Pour résoudre ce problème, on peut demander que les délimiteurs soient composés avec leur largeur maximale grâce à la clé booléenne `delimiters/max-width`.

```

\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
{
  delimiters/max-width,

```

```

    light-syntax,
    last-col, code-for-last-col = \color{blue}\scriptstyle,
  }
\setlength{\extrarowheight}{1mm}

$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 {} ;
3 -18 12 1 4 ;
-3 -46 29 -2 -15 ;
9 10 -5 4 7
\end{pNiceArray}$

...
\end{NiceMatrixBlock}

```

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{pmatrix}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{pmatrix}
 \begin{array}{l} L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}
 \begin{array}{l} L_3 \leftarrow 3L_2 + L_3 \end{array}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{pmatrix}$$

Si on souhaite un alignement des colonnes des différentes matrices sans imposer la même largeur à toutes les colonnes, on peut utiliser un grand tableau unique et placer les parenthèses avec des commandes `\SubMatrix` dans le `\CodeAfter`. Bien sûr, ce tableau ne pourra pas être coupé par un saut de page.

```

\setlength{\extrarowheight}{1mm}
\[\begin{NiceMatrix}%
[ r, last-col=6, code-for-last-col = \scriptstyle \color{blue} ]
12 & -8 & & 7 & 5 & 3 \\
3 & -18 & & 12 & 1 & 4 \\
-3 & -46 & & 29 & -2 & -15 \\
9 & 10 & & -5 & 4 & 7 \\
12 & -8 & & 7 & 5 & 3 \\
0 & 64 & & -41 & 1 & 19 & L_2 \leftarrow L_1 - 4L_2 \\
0 & -192 & & 123 & -3 & -57 & L_3 \leftarrow L_1 + 4L_3 \\
0 & -64 & & 41 & -1 & -19 & L_4 \leftarrow 3L_1 - 4L_4 \\
12 & -8 & & 7 & 5 & 3 \\
0 & 64 & & -41 & 1 & 19 \\
0 & 0 & & 0 & 0 & 0 & L_3 \leftarrow 3L_2 + L_3 \\
12 & -8 & & 7 & 5 & 3 \\
0 & 64 & & -41 & 1 & 19 \\
\CodeAfter [sub-matrix/vlines=4]
\SubMatrix({1-1}{4-5})
\SubMatrix({5-1}{8-5})
\SubMatrix({9-1}{11-5})
\SubMatrix({12-1}{13-5})
\end{NiceMatrix}\]

```

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{pmatrix}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{pmatrix}
\begin{array}{l} L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\begin{array}{l} L_3 \leftarrow 3L_2 + L_3 \end{array}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{pmatrix}$$

Dans ce tableau, les instructions `\SubMatrix` sont exécutées après la composition du tableau et les traits verticaux sont donc tracés sans espacer les colonnes.

En fait, on peut avec la clé `vlines-in-sub-matrix` choisir un spécificateur dans le préambule du tableau pour indiquer des filets verticaux qui seront tracés dans les `\SubMatrix` uniquement (en espaçant les colonnes).

```

\setlength{\extrarowheight}{1mm}
\[\begin{NiceArray}
[
  vlines-in-sub-matrix=I,
  last-col,
  code-for-last-col = \scriptstyle \color{blue}
]
{rrrrIr}
12 & -8 & & 7 & 5 & 3 & \\\
3 & -18 & & 12 & 1 & 4 & \\\
-3 & -46 & & 29 & -2 & -15 & \\\
9 & 10 & & -5 & 4 & 7 & \|[1mm]
12 & -8 & & 7 & 5 & 3 & \\\
0 & 64 & & -41 & 1 & 19 & L_2 \gets L_1-4L_2 \\\
0 & -192 & & 123 & -3 & -57 & L_3 \gets L_1+4L_3 \\\
0 & -64 & & 41 & -1 & -19 & L_4 \gets 3L_1-4L_4 \|[1mm]
12 & -8 & & 7 & 5 & 3 & \\\
0 & 64 & & -41 & 1 & 19 & \\\
0 & 0 & & 0 & 0 & 0 & L_3 \gets 3L_2+L_3 \|[1mm]
12 & -8 & & 7 & 5 & 3 & \\\
0 & 64 & & -41 & 1 & 19 & \\\
\CodeAfter
  \SubMatrix({1-1}{4-5})
  \SubMatrix({5-1}{8-5})
  \SubMatrix({9-1}{11-5})
  \SubMatrix({12-1}{13-5})
\end{NiceArray}\]

```

$$\begin{pmatrix} 12 & -8 & 7 & 5 & | & 3 \\ 3 & -18 & 12 & 1 & | & 4 \\ -3 & -46 & 29 & -2 & | & -15 \\ 9 & 10 & -5 & 4 & | & 7 \end{pmatrix}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & | & 3 \\ 0 & 64 & -41 & 1 & | & 19 \\ 0 & -192 & 123 & -3 & | & -57 \\ 0 & -64 & 41 & -1 & | & -19 \end{pmatrix}
\begin{array}{l} L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & | & 3 \\ 0 & 64 & -41 & 1 & | & 19 \\ 0 & 0 & 0 & 0 & | & 0 \end{pmatrix}
\begin{array}{l} L_3 \leftarrow 3L_2 + L_3 \end{array}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & | & 3 \\ 0 & 64 & -41 & 1 & | & 19 \end{pmatrix}$$

17.7 Comment surligner les cases d'une matrice

Pour mettre en évidence une case d'une matrice, il est possible de « dessiner » cette case avec la clé `draw` de la commande `\Block` (c'est l'un des usages des blocs mono-case⁵⁷).

```

$\begin{pNiceArray}{>{\strut}cccc}[margin, rules/color=blue]
\Block[draw]{a_{11}} & a_{12} & a_{13} & a_{14} \\
a_{21} & \Block[draw]{a_{22}} & a_{23} & a_{24} \\
a_{31} & a_{32} & \Block[draw]{a_{33}} & a_{34} \\
a_{41} & a_{42} & a_{43} & \Block[draw]{a_{44}} \\
\end{pNiceArray}$

```

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

On remarquera que les traits que l'on vient de tracer sont dessinés *après* la matrice sans modifier la position des composantes de celle-ci. En revanche, les traits tracés par `\hline`, `\Hline`, le spécificateur « | » ou les options `hlines`, `vlines`, `hvlines` et `hvlines-except-borders` « écartent » les composantes de la matrice.⁵⁸

Il est possible de colorier une rangée avec `\rowcolor` dans le `\CodeBefore` (ou avec `\rowcolor` dans une case de la rangée si on utilise la clé `colortbl-like` — même si `colortbl` n'est pas chargé).

```

\begin{pNiceArray}{>{\strut}cccc}[margin, extra-margin=2pt, colortbl-like]
\rowcolor{red!15}A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & \rowcolor{red!15}A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & \rowcolor{red!15}A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & \rowcolor{red!15}A_{44} \\
\end{pNiceArray}

```

57. On rappelle que si le premier argument obligatoire de la commande `\Block` est laissé vide, le bloc est considéré comme mono-case.

58. Pour la commande `\cline`, voir la remarque p. 8.

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix}$$

Les possibilités de réglages sont néanmoins limitées. C'est pourquoi nous présentons ici une autre méthode pour surligner une rangée d'une matrice.

Cet exemple et les suivants nécessitent d'avoir chargé Tikz (nicematrix ne charge que PGF, qui est une sous-couche de Tikz) ainsi que la bibliothèque Tikz fit, ce qui peut se faire avec les deux instructions suivantes dans le préambule du document :

```
\usepackage{tikz}
\usetikzlibrary{fit}
```

Nous créons un nœud Tikz rectangulaire qui englobe les nœuds de la deuxième rangée en utilisant les outils de la bibliothèque Tikz fit. Ces nœuds ne sont pas créés par défaut dans le `\CodeBefore` (par souci d'efficacité). Il faut utiliser la clé `create-cell-nodes` du `\CodeBefore` pour demander leur création.

```
\tikzset{highlight/.style={rectangle,
                             fill=red!15,
                             rounded corners = 0.5 mm,
                             inner sep=1pt,
                             fit=#1}}

$\begin{bNiceMatrix}
\CodeBefore [create-cell-nodes]
  \tikz \node [highlight = (2-1) (2-3)] {};
\Body
0 & \Cdots & 0 \\
1 & \Cdots & 1 \\
0 & \Cdots & 0 \\
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 0 & \cdots & 0 \\ 1 & \cdots & 1 \\ 0 & \cdots & 0 \end{bmatrix}$$

On considère maintenant la matrice suivante. Si on veut surligner chaque rangée de la matrice, on peut utiliser la technique précédente trois fois.

```
\[ \begin{pNiceArray}{ccc}[last-col]
\CodeBefore [create-cell-nodes]
  \begin{tikzpicture}
    \node [highlight = (1-1) (1-3)] {};
    \node [highlight = (2-1) (2-3)] {};
    \node [highlight = (3-1) (3-3)] {};
  \end{tikzpicture}
\Body
a & a + b & a + b + c & L_1 \\
a & a      & a + b      & L_2 \\
a & a      & a          & L_3
\end{pNiceArray} \]
```

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

Le résultat peut paraître décevant. On peut l'améliorer en utilisant les « nœuds moyens » au lieu des « nœuds normaux ».

```
\[ \begin{pNiceArray}{ccc}[last-col,create-medium-nodes]
\CodeBefore [create-cell-nodes]
\begin{tikzpicture} [name suffix = -medium]
\node [highlight = (1-1) (1-3)] {} ;
\node [highlight = (2-1) (2-3)] {} ;
\node [highlight = (3-1) (3-3)] {} ;
\end{tikzpicture}
\Body
a & a + b & a + b + c & L_1 \\
a & a & a + b & L_2 \\
a & a & a & L_3
\end{pNiceArray} \]
```

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

17.8 Utilisation de `\SubMatrix` dans le `\CodeBefore`

Dans l'exemple suivant, on illustre le produit mathématique de deux matrices.

L'ensemble de la figure est un environnement `{NiceArray}` et les trois paires de parenthèses ont été rajoutées avec `\SubMatrix` dans le `\CodeBefore`.

$$\begin{matrix} & & C_j \\ & & \begin{pmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\ \vdots & & b_{kj} & & \vdots \\ b_{n1} & \cdots & b_{nj} & \cdots & b_{nn} \end{pmatrix} \\ & & \vdots \\ L_i & \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} & \begin{pmatrix} \vdots \\ \vdots \\ \vdots & c_{ij} & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix} \end{matrix}$$

```
\tikzset{highlight/.style={rectangle,
fill=red!15,
rounded corners = 0.5 mm,
inner sep=1pt,
fit=#1}}

\[ \begin{NiceArray}{*{6}{c}@{\hspace{6mm}}*{5}{c}}[nullify-dots]
\CodeBefore [create-cell-nodes]
\SubMatrix({2-7}{6-last})
\SubMatrix({7-2}{last-6})
\SubMatrix({7-7}{last-last})
\begin{tikzpicture}
```



```

\mode [highlight = (9-2) (9-6)] { } ;
\mode [highlight = (2-9) (6-9)] { } ;
\end{tikzpicture}
\Body
& & & & & & & \color{blue}\scriptstyle C_j \\
& & & & & & & b_{11} & \Cdots & b_{1j} & \Cdots & b_{1n} \\
& & & & & & & \Vdots & & \Vdots & & \Vdots \\
& & & & & & & b_{kj} \\
& & & & & & & \Vdots \\
& & & & & & & b_{n1} & \Cdots & b_{nj} & \Cdots & b_{nn} \\
& a_{11} & \Cdots & & & a_{1n} \\
& \Vdots & & & & \Vdots & & \Vdots \\
\color{blue}\scriptstyle L_i
& a_{i1} & \Cdots & a_{ik} & \Cdots & a_{in} & \Cdots & c_{ij} \\
& \Vdots & & & & \Vdots \\
& a_{n1} & \Cdots & & & a_{nn} \\
\CodeAfter
\tikz \draw [gray,shorten > = 1mm, shorten < = 1mm] (9-4.north) to [bend left] (4-9.west) ;
\end{NiceArray}\}

```

Autre documentation

Le document `nicematrix.pdf` (fourni avec l'extension `nicematrix`) contient une traduction anglaise de la documentation ici présente, ainsi que le code source commenté et un historique des versions.

Les versions successives du fichier `nicematrix.sty` fournies par TeXLive sont disponibles sur le serveur SVN de TeXLive :

www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/nicematrix/nicematrix.sty