L'extension nicematrix *

F. Pantigny
fpantigny@wanadoo.fr

9 novembre 2026

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.

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

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 ou TeXPage) 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 LuaLaTeX.*

Cette extension nécessite et charge les extensions 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]. L'utilisateur ne doit pas utiliser la commande `\nofiles` (qui bloque l'écriture du fichier `aux`).

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 7.13 de nicematrix, en date du 2026/11/09.

[1] La dernière version de `nicematrix.sty` peut être téléchargée sur le dépôt GitHub de nicematrix : https://github.com/fpantigny/nicematrix/releases

[2] Si vous utilisez Overleaf, Overleaf effectue automatiquement un nombre de compilations suffisant (en utilisant `latexmk`).

Table des matières

1 Les environnements de cette extension **4**

2 L'espace vertical entre les rangées **4**

3 La clé baseline **5**

4 Les blocs **6**
4.1 Cas général . . . 6
4.2 Les blocs mono-colonne . . . 9
4.3 Les blocs mono-rangée . . . 9
4.4 Les blocs mono-case . . . 9
4.5 Positionnement horizontal du contenu des blocs . . . 10
4.6 Positionnement vertical du contenu des blocs . . . 12
4.7 \\ et & dans les blocs . . . 13
4.8 La clé create-blocks-in-col . . . 14

5 Les filets horizontaux et verticaux **14**
5.1 Quelques différences avec les environnements classiques . . . 14
5.1.1 Les filets verticaux . . . 14
5.1.2 La commande \cline . . . 15
5.2 L'épaisseur et la couleur des filets . . . 15
5.3 Les outils de nicematrix pour tracer des filets . . . 16
5.3.1 Les clés hlines et vlines . . . 17
5.3.2 Les clés hvlines, hlines-except-borders et hvlines-except-borders . . . 17
5.3.3 Les coins (vides) . . . 17
5.3.4 La commande \diagbox . . . 19
5.3.5 Commandes pour filets personnalisés . . . 19
5.3.6 La clé fix-vertex . . . 23
5.3.7 La clé default-line . . . 24
5.3.8 La commande \FalseRow et la lettre F . . . 25

6 Les couleurs de fond des rangées et des colonnes **25**
6.1 Utilisation de colortbl . . . 25
6.2 Les outils de nicematrix dans le \CodeBefore . . . 26
6.3 Outils de coloriage en tableau . . . 31
6.4 La couleur spéciale « nocolor » . . . 33

7 La commande \RowStyle **33**

8 La largeur des colonnes **34**
8.1 Techniques de base . . . 34
8.2 Les colonnes V de varwidth . . . 35
8.3 Les colonnes X . . . 36

9 Les rangées et colonnes extérieures **37**

10 Les lignes de conduite en pointillés dans les matrices **39**
10.1 L'option xdots/nullify . . . 40
10.2 Les commandes \Hdotsfor et \Vdotsfor . . . 41
10.3 Comment créer les lignes de conduite en pointillés de manière transparente . . . 42
10.4 Les labels des lignes en pointillés . . . 42
10.5 Personnalisation des lignes en pointillés . . . 43
10.6 Les lignes pointillées et les filets . . . 45
10.7 Les commandes \Hbrace et \Vbrace . . . 45

11 Délimiteurs dans le préambule de l'environnement **46**

12 Le \CodeAfter **47**

12.1 La commande \line dans le \CodeAfter . . . 48
12.2 La commande \SubMatrix dans le \CodeAfter (et le \CodeBefore) . . . 48
12.3 Les commandes \OverBrace et \UnderBrace dans le \CodeAfter . . . 51
12.4 La commande \TikzEveryCell . . . 52

13 Les légendes et les notes dans les tableaux **54**
13.1 La légendes des tableaux . . . 54
13.2 Les notes de pied de page . . . 54
13.3 Les notes de tableaux . . . 54
13.4 Personnalisation des notes de tableau . . . 56

14 Autres fonctionnalités **59**
14.1 Arbres . . . 59
14.2 La clé rounded-corners des environnements . . . 60
14.3 La commande \ShowCellNames . . . 61
14.4 Utilisation du type de colonne S de siunitx . . . 61
14.5 Type de colonne par défaut dans {NiceMatrix} . . . 61
14.6 La commande \rotate . . . 62
14.7 La clé small . . . 62
14.8 \AutoNiceMatrix et les compteurs iRow et jCol . . . 63
14.9 La clé light-syntax . . . 64
14.10 Couleur des délimiteurs . . . 64
14.11 L'environnement {NiceArrayWithDelims} . . . 65
14.12 La commande \OnlyMainNiceMatrix . . . 65

15 Utilisation explicite de TikZ avec nicematrix **65**
15.1 Les nœuds correspondant aux contenus des cases . . . 65
15.1.1 La clé pgf-node-code . . . 67
15.1.2 Les colonnes V de varwidth . . . 67
15.2 Les « nœuds moyens » et les « nœuds larges » . . . 67
15.3 Les nœuds indiquant la position des filets . . . 69
15.4 Les nœuds correspondant aux commandes \SubMatrix . . . 70

16 API pour les développeurs **70**
16.1 Jeux de clés publics . . . 70
16.2 CodeBefore et CodeAfter . . . 71

17 Remarques techniques **72**
17.1 Lignes diagonales . . . 72
17.2 Les cases « vides » . . . 73
17.3 L'option exterior-arraycolsep . . . 73
17.4 Incompatibilités . . . 74
17.5 Compatibilité avec le Tagging Project de LaTeX . . . 74

18 Exemples **75**
18.1 Notes dans les tableaux . . . 75
18.2 Utilisation de la clé « tikz » de la commande \Block . . . 76
18.3 Utilisation avec tcolorbox . . . 77
18.4 Un tableau de signes . . . 78
18.5 Un tableau de variations . . . 78
18.6 Exemple de description d'une table de base de données . . . 78
18.7 Exemple d'utilisation de la clé borders de \Block . . . 79
18.8 Lignes de conduite . . . 79
18.9 Illustrer les nombres de lignes et de colonnes d'une matrice . . . 81
18.10 Empilements de matrices . . . 83
18.11 Comment surligner les cases d'une matrice . . . 86
18.12 Un tableau triangulaire . . . 88

Index **90**

1 Les environnements de cette extension

L'extension nicematrix définit les nouveaux environnements suivants :

{NiceTabular}	{NiceArray}	{NiceMatrix}	{NiceArrayWithDelims}
{NiceTabular*}	{pNiceArray}	{pNiceMatrix}	
{NiceTabularX}	{bNiceArray}	{bNiceMatrix}	
	{BNiceArray}	{BNiceMatrix}	
	{vNiceArray}	{vNiceMatrix}	
	{VNiceArray}	{VNiceMatrix}	

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]

L'environnement {NiceArrayWithDelims} est une généralisation de {NiceArray} et ses variantes (cf. 14.11, p. 65).

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'accélérer fortement les compilations).

Par ailleurs, quand on utilise un des environnements de nicematrix, il est souvent possible d'accélérer fortement la compilation en utilisant la `no-cell-nodes` (le temps de compilation peut alors être divisé par 3 ou 4). En effet de nombreuses fonctionnalités de nicematrix n'utilisent pas les nœuds PGF-TikZ correspondant aux contenus des cases (cf. 15.1, p. 65) et la clé `no-cell-nodes` désactive leur création (la création de ces nœuds ne peut être désactivée par nicematrix car nicematrix n'a pas de moyen de savoir si l'utilisateur final ne les utilise pas directement par lui-même en utilisant TikZ après l'environnement).

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[4] des tableaux créés par défaut avec LaTeX sont trop proches l'une de l'autre. On en donne ci-dessous un exemple classique.

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

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



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.[5]

[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. 8.3, p. 36. On peut donc toujours se passer de {NiceTabularX}.

[4] Dans ce document, on parlera de *rangée* pour désigner uniquement les rangées horizontales. Les colonnes, par opposition, sont verticales.

[5] 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.

La valeur initiale de ces paramètres vaut 0 pt.

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

Ces clés acceptent une syntaxe normale et une syntaxe « additive », c'est-à-dire que l'on peut écrire, par exemple, `cell-space-limits = 3pt` ou `cell-space-limits += 2pt`.

```
$\begin{pNiceMatrix}[cell-space-limits = 1pt]
\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{3} & \frac{1}{4} \\
\end{pNiceMatrix}$
```

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

Il est également possible de changer ces paramètres pour certaines rangées du tableau seulement grâce à la commande `\RowStyle` (cf. 7, p. 33).

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}` tels que définis par l'extension 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 l'extension 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 un 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. un item
2. $\begin{array}{lcccccc} \toprule n & 0 & 1 & 2 & 3 & 4 & 5 \\ \midrule u_n & 1 & 2 & 4 & 8 & 16 & 32 \\ \bottomrule \end{array}$

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

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

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.[6]
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.

  Dans `{NiceTabular}`, `{NiceTabular*}` et `{NiceTabularX}`, le contenu est composé en mode texte tandis que, dans les autres environnements, il est composé en mode mathématique.

  Pour des raisons techniques liées à LaTeX, on ne peut pas débuter le contenu du bloc par une commande `\color` (mais on peut envisager d'utiliser `\textcolor`).[7]

[6] Les espaces situés après une commande `\Block` sont supprimés.
[7] cf. https://tex.stackexchange.com/questions/674788

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

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

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

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.[8]

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

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

La commande `\Block` accepte en premier argument optionnel (entre crochets) une liste de couples *clé=valeur*.

Les premières clés sont des outils rapides pour contrôler l'apparence du bloc :

- la clé `fill` prend en argument une couleur et remplit le bloc avec cette couleur ;
- la clé `opacity` fixe l'opacité de la couleur de remplissage donnée par `fill` ;[9]
- la clé `draw` trace le cadre du bloc avec la couleur courante pour les filets (si on veut une autre couleur, il est possible de la passer comme valeur de la clé `draw`) ;
- les clés `hlines`, `vlines` et `hvlines` tracent les filets correspondants dans le bloc[10] ;
- la clé `rules/width` fixe la largeur utilisée pour tracer les filets (n'a d'intérêt que si `draw`, `hvlines`, `hlines` ou `vlines` est utilisée) ;
- la clé `rules/color` fixe la couleur utilisée pour tracer les filets (n'a d'intérêt que si `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 pt, ce qui est la valeur initiale des *rounded corners* de TikZ.)[11] ;

Ces outils ne sont parfois pas suffisants pour contrôler l'apparence du bloc. Les clés suivantes sont plus puissantes, mais plus difficiles d'utilisation. Elles nécessitent également que TikZ soit chargé (par `\usepackage{tikz}`). Par défaut, nicematrix ne charge pas TikZ mais uniquement PGF, qui est une sous-couche de TikZ.

[8] 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. On peut aussi y mettre la commande `\rotate` fournie par nicematrix (cf. 14.6, p. 62).

[9] Attention : cette fonctionnalité génère des instructions de transparence dans le PDF résultant et certains lecteurs de PDF n'acceptent pas la transparence. L'application de l'opacité est faite par `\pgfsetfillopacity` de PGF.

[10] 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, sauf si ces blocs possèdent la clé `transparent` (cf. 5, p. 14).

[11] Il existe aussi une clé `rounded-corners` applicable à un environnement de nicematrix : cf. 14.2, p. 60.

- 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`, `bottom` et `all` ; 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 (voir l'exemple 18.7, p. 79). Quand la clé `borders` est utilisée sans valeur, elle est équivalente à la clé `draw` (sans valeur).

- Quand la clé `tikz` est utilisée, le chemin TikZ correspondant au rectangle délimitant le bloc est exécuté avec TikZ[12] 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 18.2, p. 76.

  En fait, dans la liste des clés fournies à `tikz`, on peut mettre une clé `offset`. Cette clé n'est pas fournie par TikZ mais par nicematrix. Elle réduit le rectangle correspondant au bloc par une marge (horizontalement et verticalement) égale à la valeur (passée à `offset`). C'est ce rectangle réduit qui sera le chemin exécuté par TikZ avec comme options les autres clés passées à la clé `tikz`.

  Nouveau 7.11 : Des clés plus précises `xoffset` et `yoffset` sont fournies.

Enfin, il existe quelque clés techniques :

- la clé `name` donne un nom au nœud TikZ rectangulaire correspondant au bloc ; on peut utiliser ce nom avec TikZ dans le `\CodeAfter` (cf. 12, p. 47) ;

- la clé `respect-arraystretch` évite la remise à 1 de `\arraystretch` en début de bloc (qui a lieu par défaut) ;

- Par défaut, les filets ne sont pas tracés dans les blocs (voir à ce sujet la partie sur les filets, section 5, p. 14). Néanmoins, si la clé `transparent` est utilisée, les filets seront tracés.[13]

  Pour un exemple, voir la section 18.2, p. 76.

  Attention : cette clé n'implique pas du tout que le contenu du bloc sera transparent.

- **Nouveau 7.12**

  Avec la clé `empty`, toutes les cases du bloc seront déclarées « vides », ce qui a un impact sur le comportement de plusieurs fonctionnalités de nicematrix, comme la clé `corners` pour le calcul des coins vides (cf. 5.3.3, p. 17).

Il existe aussi des clés de positionnement horizontal et vertical du contenu du bloc qui sont décrites ci-dessous (cf. 4.5, p. 10).

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 dans certaines conditions comme expliqué plus loin.

Dans l'exemple suivant, on a dû élargir à la main les colonnes 2 et 3 (avec la construction classique `w{c}{...}` de l'extension array).

```
\begin{NiceTabular}{cw{c}{2cm}w{c}{3cm}c}
rose      & tulipe & marguerite & dahlia \\
violette
& \Block[C,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}
```



[12] TikZ doit être chargé préalablement (par défaut, nicematrix ne charge que PGF), faute de quoi, une erreur sera levée.
[13] Par ailleurs, la commande `\TikzEveryCell` disponible dans le `\CodeAfter` et le `\CodeBefore`, ne s'applique aux blocs avec la clé `transparent`.

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.

- 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 (`p{...}`, `b{...}`, `m{...}`, `w{...}{...}`, `W{...}{...}`, `V{...}`, similaires aux colonnes `V` de varwidth, et `X`, similaires aux colonnes `X` de tabularx), 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. Pour un bloc dans une colonne de type `p{...}`, `b{...}`, `m{...}`, `V{...}` ou `X`, c'est un alignement `c` qui est retenu par défaut. Néanmoins, ces types de colonnes peuvent avoir une option d'alignement (par ex. `p[l]{...}`), et dans ce cas-là, c'est cette option d'alignement qui est transmise au bloc.

  Notons enfin que le bloc peut avoir sa propre spécification d'alignement horizontal : cf. 4.5, p. 10.

- 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).

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

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



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), sauf lorsqu'une option de placement vertical a été utilisée pour le bloc (une des clés `t`, `b`, `m`, `T` et `B` décrites à la partie 4.6, p. 12).

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 de texte.

- On peut couper la case en plusieurs parties avec `&` quand la clé `&-in-blocks` est activée (cf. 4.7, p. 13).
- On peut utiliser l'option d'alignement horizontal du bloc pour déroger à la consigne générale donnée dans le préambule pour cette colonne (cf. 4.5 p. 10).
- 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`.[14]
- On peut tracer une ou plusieurs bordures de la case avec la clé `borders`.

4.5 Positionnement horizontal du contenu des blocs

La commande `\Block` admet les clés `l`, `c` et `r` pour la position horizontale du contenu du bloc (calé à gauche, centré ou bien calé à droite).

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

$$\left[\begin{array}{ccc|c} & & & 0 \\ & \Block[r]{3-3}<\LARGE>{A} & & \Vdots \\ & & & 0 \\ \hline 0 & \Cdots & 0 & 0 \end{array}\right]$$

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`.[15]

[14] 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`.

[15] On remarquera que les clés `L`, `R` et `C` nécessitent moins de calculs que les clés `l`, `r` et `c`. Si on tient à l'efficacité, on devrait écrire `\Block[C]` systématiquement par défaut.

Voici le même exemple avec la clé `C` pour le premier bloc.

```
\begin{NiceTabular}{@{}c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
Rang & \Block[C]{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

La commande `\Block` accepte aussi les clés `p` et `j`. Avec la clé `p`, le contenu du bloc est composé comme un paragraphe (de manière similaire à une colonne standard de type `p`). Cette clé peut s'utiliser en conjonction avec les clés `l`, `c` ou `r` et, alors, le paragraphe est composé avec `\raggedright`, `\centering` ou `\raggedleft` (en fait, quand `ragged2e` est chargée, ce sont les commandes `\RaggedRight`, `\Centering` et `\RaggedLeft` fournies par cette extension qui seront utilisées au lieu de `\raggedright`, `\centering` et `\raggedleft`). Avec la clé `j` (qui force la clé `p`), le paragraphe est composé de manière justifiée.

On peut mettre un environnement `{itemize}` ou `{enumerate}` dans un bloc qui utilise la clé `p` ou la clé `j` (dans les autres cas, on aura une erreur LaTeX : `Not allowed in LR mode`). Dans l'exemple suivant, on a chargé l'extension `enumitem` (pour pouvoir utiliser la clé `left` de l'environnement `{itemize}`).

```
\begin{NiceTabular}[hvlines]{ccc}
un & deux deux & trois trois \\
un &
\Block[p,l]{*-2}{%
\begin{itemize}[left=0pt]
\item un deux trois quatre cinq
\item deux
\item trois
\end{itemize}%
} \\
un & \\
un & \\
un & \\
un & \\
un & \\
un & \\
\end{NiceTabular}
```



un	deux deux	trois trois
un	• un deux trois quatre cinq • deux • trois	
un		
un		
un		
un		
un		
un		

4.6 Positionnement vertical du contenu des blocs

Concernant le positionnement vertical, la commande `\Block` admet les clés `m`, `t`, `b`, `T` et `B`.

- Avec la clé `m`[16], le contenu du bloc est centré verticalement.
- Avec la clé `t`, la ligne de base du contenu du bloc est alignée avec la ligne de base de la première rangée concernée par le bloc.
- Avec la clé `b`, la ligne de base de la dernière rangée du contenu du bloc (rappelons que le contenu du bloc peut comporter plusieurs lignes de texte séparées par `\\`) est alignée avec la ligne de base de la dernière des rangées du tableau impliquées dans le bloc.
- Avec la clé `T`, le contenu du bloc est calé vers le haut.

  Il n'y a pas de marge verticale. Néanmoins, le contenu du bloc est (toujours) composé en interne dans une `{minipage}`, un `{tabular}` ou un `{array}`, ce qui fait qu'il y a souvent déjà une marge. Si besoin est, on peut toujours ajouter un `\strut`.
- Avec la clé `B`, le contenu du bloc est calé vers le bas.

Quand aucune clé n'est donnée, c'est la clé `m` qui s'applique (sauf pour les blocs mono-rangée).

```
\NiceMatrixOptions{rules/color=[gray]{0.75}, hvlines}

\begin{NiceTabular}{ccc}
\Block[fill=red!10,t,l]{4-2}{deux\\lignes}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

deux lignes		Un
		deux
		trois
		quatre
text	text	

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,b,r]{4-2}{deux\\lignes}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

		Un
		deux
		trois
deux lignes		quatre
text	text	

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,T,L]{4-2}{deux\\lignes}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

deux lignes		Un
		deux
		trois
		quatre
text	text	

[16] Cette clé a un alias : `v-center`.

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,B,R]{4-2}{deux\\lignes}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

		Un
		deux
		trois
	deux lignes	quatre
text	text	

4.7 \\ et & dans les blocs

L'extension nicematrix offre la possibilité d'utiliser directement `\\` et `&` dans le contenu d'un bloc (dans le but de formater son contenu) mais il y a quelques restrictions.

- On ne doit pas utiliser à la fois `&` et `\\` dans le même bloc.
- Pour `\\`, il n'y a pas d'autres restrictions. On peut utiliser `\\` dans un bloc pour composer du texte sur plusieurs lignes.
- Pour pouvoir utiliser `&`, la clé `ampersand-in-blocks` (alias : `&-in-blocks`) doit avoir été activée[17]. Le bloc est alors divisé en sous-blocs comme illustré ci-dessous. Attention toutefois : quand `ampersand-in-blocks` est utilisée, l'argument (principal) de la commande `\Block` est découpé syntaxiquement au niveau des esperluettes `&`, celles entre accolades sont masquées mais pas celles dans un environnement.[18]

L'esperluette `&` permet de diviser horizontalement un bloc en sous-blocs *de même taille*.

```
\begin{NiceTabular}{ll}%
   [hvlines,ampersand-in-blocks]
  & les cinq premiers entiers naturels \\
3 & \Block{}{un&deux&trois} \\
4 & \Block{}{un&deux&trois&quatre} \\
5 & \Block{}{un&deux&trois&quatre&cinq} \\
\end{NiceTabular}
```

	les cinq premiers entiers naturels				
3	un	deux	trois		
4	un	deux	trois	quatre	
5	un	deux	trois	quatre	cinq

Comme on le voit, chaque bloc (ici, à chaque fois mono-case) a été divisé en sous-cases de *même taille*. Dans le cas présent, on aurait peut-être préféré le codage suivant :

```
\begin{NiceTabular}{lccccc}%
   [hvlines,ampersand-in-blocks]
  & \Block{1-5}{les cinq premiers
                 entiers naturels} \\
3 & \Block{1-5}{un & deux & trois} \\
4 & \Block{1-5}{un&deux
     &\cellcolor{red!15}trois&quatre} \\
5 & un & deux & trois & quatre & cinq \\
\end{NiceTabular}
```

	les cinq premiers entiers naturels				
3	un	deux	trois		
4	un	deux	trois	quatre	
5	un	deux	trois	quatre	cinq

Dans ce codage, il s'agit de blocs de taille `1-5` qui sont coupés en trois et quatre sous-blocs.

On a illustré la possibilité de colorier une sous-case en utilisant la commande `\cellcolor`.

[17] Si ce n'est pas le cas, l'utilisation de `&` dans l'argument principal de la commande `\Block` provoquera une erreur : `! Extra alignment tab has been changed to \cr` de TeX.

[18] On ne peut donc pas écrire : `\Block[ampersand-in-blocks]{}{\begin{array}{cc}1&2\end{array}}`. Bien sûr, on peut le faire sans la clé `ampersand-in-blocks`.

4.8 La clé create-blocks-in-col

La clé `create-blocks-in-col` prend en argument une liste de numéros de colonnes et pour, chacune de ces colonnes, crée dans celle-ci un bloc pour chaque case non vide de la colonne (le bloc en question regroupe la case non vide et toutes les cases vides successives situées en-dessous).
Ces blocs seront, comme tous les blocs, respectés par les clés de tracés de filets (`vlines`, `hlines`, etc. : cf. 5, p. 14) et par les commandes de coloriage des cases (cf. 6.2, p. 26). C'est là l'intérêt d'utiliser cette clé `create-blocks-in-col`.

```
\begin{NiceTabular}{lr}[hvlines,create-blocks-in-col=1]
\rowcolors{blue!10}{}[respect-blocks]
Pierre    & 12 \\
          & 13 \\
Jacques   &  8 \\
Stéphanie & 18 \\
          & 17 \\
          & 15 \\
Amélie    & 20 \\
Henri     & 14 \\
Estelle   & 15 \\
          & 19
\end{NiceTabular}
```

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



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 rangée incomplète ou de double filet horizontal spécifié par `\hline\hline` (il n'y a pas besoin d'utiliser l'extension hhline[19]).

```
\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. 4, p. 6) ni dans les coins (dont la création est demandée par la clé `corners` : cf. 5.3.3, p. 17), ni dans les éventuelles rangées extérieures (créées par les clés `first-row` et `last-row` : cf. 9, p. 37).

Si on utilise `\multicolumn`, il n'y a pas besoin de rajouter les éventuels `|` ou `||` à la fin du motif (mais, dans les environnements de nicematrix, il est souvent plus commode d'utiliser la commande `\Block`).

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

[19] Mais hhline *peut* être utilisé avec nicematrix : voir l'exemple 18.6, p. 78.

que les filets tracés par nicematrix sont compatibles avec booktabs. Remarquez que nicematrix ne charge *pas* booktabs.

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

a	b	c	d
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. On utilise pour cela la commande standard `\newcolumntype` de array.

```
\newcolumntype{I}{!{\vrule}}
\begin{NiceTabular}{IcIcI} \hline
Premier & Deuxième \\ \hline\hline
Paul  & \\ \hline
Marie & Pauline \\ \hline
\end{NiceTabular}
```

Premier	Deuxième
Paul	
Marie	Pauline



Bien sûr, dans cet exemple, il n'y a pas d'intérêt à utiliser `{NiceTabular}` et il vaut mieux utiliser l'environnement standard `{tabular}` mais peut-être voudra-t-on utiliser d'autres fonctionnalités de nicematrix. On aura aussi intérêt à considérer la commande `\FalseRow` (cf. 5.3.8, p. 25) quand on veut tracer des double-filets horizontaux.

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 de LaTeX `\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`. Néanmoins, nicematrix propose aussi une clé `rules/color`, disponible dans `\NiceMatrixOptions` ou dans un environnement

individuel, pour fixer la couleur des filets. Cette clé fixe localement la couleur des filets (alors que la commande `\arrayrulecolor` agit globalement !). Elle est à privilégier.

```
\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



En fait, dans cet exemple, au lieu de `\hline`, il aurait mieux valu utiliser la commande `\Hline`, fournie par nicematrix et décrite ci-dessous, car elle garantit un meilleur résultat dans les lecteurs de PDF aux bas niveaux de zoom.

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`, `hlines-except-borders` et `hvlines-except-borders` ;
- le spécificateur « | » dans le préambule (pour les environnements à préambule) ;
- la commande `\Hline`.

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

- Les blocs en question sont :
  - ceux créés par la commande `\Block`[20] de nicematrix présentée à la section 4, p. 6 ;
  - ceux créés par la clé `create-blocks-in-col` : cf. 4.8, p. 14 ;
  - ceux délimités implicitement par des lignes en pointillés continues, créées par `\Cdots`, `\Vdots`, etc. : cf. 10, p. 39.
- Les coins sont créés par la clé `corners` détaillée un peu plus loin : cf. 5.3.3, p. 17.
- Pour les rangées et colonnes extérieures, cf. 9, p. 37.

En particulier, cette remarque montre déjà une différence entre la commande standard `\hline` et la commande `\Hline` proposée par nicematrix.
Par ailleurs, la commande `\Hline` admet entre crochets un argument optionnel qui est une liste de couples *clé=valeur* qui décrivent un filet. Pour la description de ces clés, voir la clé `custom-line`, à la section 5.3.5, p. 19.[21]
De même que la commande `\Hline`, le spécificateur « | » admet entre crochets des options qui caractérisent le filet à tracer.

```
\begin{NiceTabular}{| c | [color=blue] c |}
\Hline
a & b \\
\Hline[color=red]
c & d \\
\Hline
\end{NiceTabular}
```

a	b
c	d



[20] Et aussi la commande `\multicolumn` même s'il est recommandé d'utiliser plutôt `\Block` quand on utilise l'extension nicematrix (la commande `\multicolumn` garde néanmoins par rapport à `\Block` la caractéristique d'élargir la dernière colonne au besoin).

[21] Remarque technique. Si l'utilisateur définit une commande par-dessus la commande `\Hline`, il doit veiller à ce qu'elle soit *développable* au sens de TeX (en utilisant `\NewExpandableDocumentCommand` de LaTeX3, `\newcommand` de LaTeX ou même `\def` de TeX). Exemple : `\NewExpandableDocumentCommand{\RedLine}{}{\Hline[color=red]}`

5.3.1 Les clés hlines et vlines

Les clés `hlines` et `vlines` tracent des filets horizontaux et verticaux.
Si aucune valeur n'est donnée, tous les filets sont tracés.

Quand une valeur est présente, il s'agit d'une liste de numéros de filets à tracer.

- Il est possible de mettre des intervalles de numéros de la forme i-j.
- Il est possible de mettre des numéros négatifs qui sont alors comptés à partir de la fin.

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).

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

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}$$

Même quand la clé `hlines` est utilisée, il reste possible d'utiliser `\Hline\Hline` pour placer un filet double horizontal. De même, on peut mettre `||` dans le préambule (d'un environnement à préambule) pour placer un double filet vertical, même quand la clé `vlines` est utilisée.

```
$\begin{NiceArray}{c||ccccc}[hlines,vlines]
  & a & b & c & d & e \\ \Hline\Hline
x & 0 & 0 & 0 & 0 & 0 \\
y & 0 & 0 & 0 & 0 & 0 \\
z & 0 & 0 & 0 & 0 & 0 \\
\end{NiceArray}$
```

	a	b	c	d	e
x	0	0	0	0	0
y	0	0	0	0	0
z	0	0	0	0	0

5.3.2 Les clés hvlines, hlines-except-borders 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,width=1pt}]
rose      & tulipe & marguerite & dahlia \\
violette  & \Block[C,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

On remarquera que quand la clé `rounded-corners` est utilisée pour l'environnement `{NiceTabular}`, la clé `hvlines` trace des coins arrondis pour le tableau : cf. partie 14.2, p. 60.

La clé `hlines-except-borders` est similaire à la clé `hlines` mais ne trace pas les filets sur les bords horizontaux du tableau. 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. Pour un exemple d'utilisation de cette clé, voir l'exemple 18.3, p. 77.

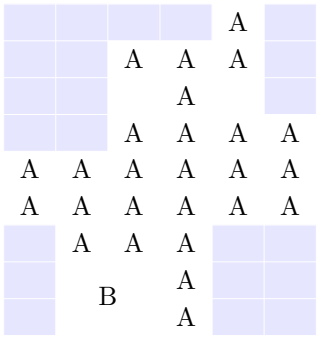
5.3.3 Les coins (vides)

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.[22]

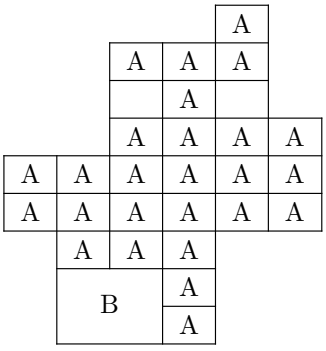
On peut néanmoins imposer à une case sans contenu d'être considérée comme non vide par nicematrix avec la commande `\NotEmpty`.

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



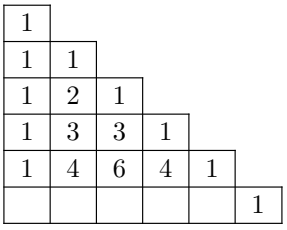
Quand la clé `corners`[23] est utilisée, nicematrix calcule les quatre coins vides et ces coins sont alors pris en compte par les outils de tracés de filets (les filets ne seront pas tracés dans ces coins vides).

```
\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 \\
  & \Block{2-2}{B} & & A \\
  &   &   & A \\
\end{NiceTabular}
```



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}
```



▷ 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. 6.2, p. 26). La commande `\TikzEveryCell` disponible dans le `\CodeAfter` et le `\CodeBefore` (cf. 12.4, p. 52) tient également compte des coins.

[22] 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. 17.2, p. 73).

[23] La clé `corners` dont on parle là n'a pas de rapport direct avec la clé `rounded-corners`, décrite dans la partie 14.2, p. 60.

Nouveau 7.10

Une généralisation des « coins » a été ajoutée avec des valeurs `N`, `S`, `W` et `E`.

```
\begin{NiceTabular}{cccc}[hvlines,corners=E]
1 & 2 \\
1 & 2 & 3 \\
1 & 3 \\
1 \\
1 & 2 & 3 & 4 \\
\end{NiceTabular}
```

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

5.3.4 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 \ 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`.

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\Block{2-2}{\diagbox{x}{y}} &  & a & b & c \\
  &  & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

x \ y		a	b	c
		a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Mais on peut aussi utiliser `\diagbox` uniquement pour le trait diagonal et placer les labels comme habituellement.

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\Block{2-2}{\diagbox{}{}} & y & a & b & c \\
x &  & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

x \ y		a	b	c
		a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Il est de toutes manières toujours possible de tracer tous les traits souhaités avec TikZ dans le `\CodeAfter` (ou le `\CodeBefore`) en utilisant les nœuds PGF-TikZ créés par nicematrix : cf. 15, p. 65.

5.3.5 Commandes pour filets personnalisés

Il est en fait 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 trois clés pour spécifier les outils qui permettront d'utiliser ce nouveau type de filet.

- la clé `command` indique le nom (*avec* ou *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`) ;

- la clé `ccommand` indique le nom (*avec* ou *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 sera une liste d'intervalles de colonnes spécifiés par la syntaxe i ou i-j ;[24]
- la clé `letter` prend en argument une lettre[25] qui pourra être utilisée par l'utilisateur dans le préambule d'un environnement à préambule (comme `{NiceTabular}`) pour spécifier un filet vertical.

On traite maintenant de la description du filet elle-même. Les options qui suivent peuvent aussi s'utiliser dans l'argument optionnel d'une commande `\Hline` individuelle ou dans l'argument optionnel d'un spécificateur « | » dans un préambule d'environnement. Elles sont aussi utilisées par la clé `default-line`.

Il y a quatre possibilités :

- les filets de base, que l'on peut tracer en utilisant (par exemple) colortbl : ce sont des filets simples ou multiples, colorés et avec une couleur entre les filets ;
- des filets en tiretés (clé `dashed`) ;
- des filets en pointillés ronds (clé `dotted`) ;
- tous les filets traçables avec TikZ (clé `tikz`).

On décrit maintenant ces quatre types de filets.

- *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). Le nom de cette clé est inspirée par la commande `\doublerulesepcolor` de colortbl.

  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}{lcIcIc}[custom-line = {letter=I, color=red}]
\Hline
          & \Block{1-3}{dimensions} \\
          & L & l & h \\
\Hline
Produit A & 3 & 1 & 2 \\
Produit B & 1 & 3 & 4 \\
Produit C & 5 & 4 & 1 \\
\Hline
\end{NiceTabular}
```



[24] Il est recommandé de n'utiliser ces commandes qu'une seule fois par rangée car chaque utilisation crée un espace vertical entre les rangées correspondant à la largeur totale du trait qui sera tracé. De toutes manières, on peut tracer plusieurs filets avec une unique utilisation de la commande.

[25] Les lettres suivantes ne sont pas autorisées : `lcrpmbFVX|()[]!@<>`

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

La clé `sep-color` avec la valeur `white` peut être en particulier utile en cas de filet double au-dessus d'une case colorée (en effet, contrairement à ce qui se passe avec `{tabular}` et colortbl, la couleur s'applique par défaut entre les deux filets).

```
\NiceMatrixOptions
  {
    custom-line =
     {
       command = \DoubleRule ,
       multiplicity = 2 ,
       sep-color = white
     }
  }
\begin{NiceTabular}{ccc}
un & deux & trois \\
\DoubleRule
quatre & \cellcolor{yellow} cinq & six \\
\end{NiceTabular}
```

un	deux	trois
quatre	cinq	six

Néanmoins, pour des double-filets en conjonction avec des fonds colorés, peut-être vaut-il mieux utiliser les « fausses rangées » proposées par la commande `\FalseRow` (et la lettre `F` pour les « fausses colonnes ») : cf. 5.3.8, p. 25.

- *Deuxième possibilité* : la clé `dashed`

L'extension nicematrix propose les commandes `\hdashedline` et `\cdashedline` et la lettre « ; » pour des filets en tiretés. Elles sont disponibles quand TikZ est chargé car il s'agit simplement d'un filet TikZ de style `dash pattern = on 4 pt off 2pt`. En fait, si la librairie TikZ decorations est chargée (avec `\usetikzlibrary{decorations}`), la clé `dash expand off` est ajoutée pour un meilleur résultat.

Il est donc possible d'utiliser les commandes `\hdashedline` et `\cdashedline` pour tracer des filets horizontaux en tiretés.

```
\usepackage{tikz} % dans le préambule
\usetikzlibrary{decorations} % dans le préambule

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

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

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

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

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

Si on veut une réservation de place plus importante pour ces filets, il faut définir, avec `custom-line`, des commandes personnalisées utilisant cette clé `dashed` en conjonction avec la clé `total-width`.

```
\NiceMatrixOptions
  {
    custom-line =
     {
       command = \myhdashedline ,
       ccommand = \mycdashedline ,
       dashed,
       total-width = 10 pt
     }
  }
```

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

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

- *Troisième possibilité* : la clé `dotted`

L'extension nicematrix fournit les commandes `\hdottedline` et `\cdottedline` et la lettre « : » pour des filets en pointillés ronds. Pour ces filets, nicematrix n'utilise pas TikZ mais un système interne. En fait, il s'agit du même que celui utilisé par les commandes `\Cdots`, `\Vdots`, etc. (cf. 10, p. 39).

Il est donc possible d'utiliser les commandes `\hdottedline` et `\cdottedline` pour tracer des filets horizontaux en pointillés.

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

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

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

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

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

Comme précédemment avec la clé `dashed`, on peut utiliser la clé `total-width` en conjonction avec la clé `dotted`.

- *Quatrième possibilité* : la clé `tikz`

  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`).

```
\usepackage{tikz}                       % dans le préambule
\usetikzlibrary{decorations.pathmorphing} % dans le préambule

\NiceMatrixOptions
  {
    custom-line =
     {
       letter = I ,
       tikz = { decorate, decoration = zigzag } ,
       total-width = 6 pt
     }
  }
\begin{NiceTabular}{cIcIc}
un & deux & trois \\
quatre & cinq & six \\
sept & huit & neuf
\end{NiceTabular}
```

un	deux	trois
quatre	cinq	six
sept	huit	neuf

Comme on l'a dit, les clés précédentes peuvent être utilisées dans l'argument optionnel d'une commande `\Hline` individuelle ou dans l'argument optionnel d'un spécificateur « | » dans un préambule d'environnement (par exemple pour `{NiceTabular}`).
Mais dans ces cas-là, il est aussi possible d'utiliser deux clés supplémentaires, `start` et `end`, qui indiquent les numéros de rangées ou de colonnes des extrémités du filet.

```
\begin{NiceTabular}{cc|[color=blue,start=2]ccc}
un & deux & trois & quatre
\Hline[start=2,end=3]
cinq & six & sept & huit \\
neuf & dix & onze & douze
\end{NiceTabular}
```

un	deux	trois	quatre
cinq	six	sept	huit
neuf	dix	onze	douze

5.3.6 La clé fix-vertex

Nouveau 7.9

Supposons que l'on ait défini une commande `\RandomLine` pour une ligne qui zigzague de manière aléatoire.

```
\usepackage{tikz}                       % dans le préambule
\usetikzlibrary{decorations.pathmorphing} % dans le préambule

\NiceMatrixOptions
  {
    custom-line =
     {
       command = \RandomLine ,
       tikz =
```

```
        {
          decorate ,
          decoration = { random steps, segment length = 5pt, amplitude = 3pt }
        }
    }
}
```

Considérons alors le tableau suivant, où cette commande `\RandomLine` est utilisée à la fin :

```
\begin{NiceTabular}{ccc}[vlines]
\Hline
nom & prénom & âge \\
\Hline
Dubourg & Claude & 88 \\
Challa & Rémi & 72 \\
Ducar & Estelle & 75 \\
Migaud & Sophie & 18 \\[1mm]
\RandomLine
\end{NiceTabular}
```

nom	prénom	âge
Dubourg	Claude	88
Challa	Rémi	72
Ducar	Estelle	75
Migaud	Sophie	18

Les filets verticaux internes ne s'arrêtent pas nécessairement au niveau de la ligne en zigzag finale puisque celle-ci est tracée de manière aléatoire directement du coin inférieur gauche au coin inférieur droit.
Pour remédier à ce problème, l'extension nicematrix propose la clé `rules/fix-vertex` qui demande à ce que les filets soient coupés en segments au niveau des nœuds et que ces segments soient tracés indépendamment les uns des autres.

```
\begin{NiceTabular}{ccc}[vlines,rules/fix-vertex]
\Hline
nom & prénom & âge \\
\Hline
Dubourg & Claude & 88 \\
Challa & Rémi & 72 \\
Ducar & Estelle & 75 \\
Migaud & Sophie & 18 \\[1mm]
\RandomLine
\end{NiceTabular}
```

nom	prénom	âge
Dubourg	Claude	88
Challa	Rémi	72
Ducar	Estelle	75
Migaud	Sophie	18

5.3.7 La clé default-line

Nouveau 7.9

L'extension nicematrix propose la clé `default-line` qui fixe le style des filets qui sera utilisé par défaut par la commande `\Hline`, les spécificateurs `|` dans les préambules et les clés `hvlines`, `vlines`, `hlines`, etc.
Les possibilités de spécification de ces filets sont les mêmes que pour les filets personnalisés définis par la clé `custom-line` (cf. 5.3.5, p. 19).

```
\usepackage{tikz}                          % dans le préambule
\usetikzlibrary{decorations.pathmorphing} % dans le préambule

\NiceMatrixOptions
  {
    default-line =
      {
        tikz =
          {
            decorate ,
            decoration = { random steps, segment length = 3pt, amplitude = 1pt }
          }
      }
  }
```

```
\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{ccc}[hvlines,rules/fix-vertex]
un     & deux & trois \\
quatre & cinq & six \\
sept   & \Block{1-2}{huit} \\
\end{NiceTabular}
```

un	deux	trois
quatre	cinq	six
sept	huit	



5.3.8 La commande \FalseRow et la lettre F

Nouveau 7.11

Pour des double-filets parfaitement compatibles avec les fonds colorés, nicematrix propose une commande `\FalseRow` et une lettre `F`.

La commande `\FalseRow` est à utiliser après une commande `\\` de fin de rangée. Elle crée une nouvelle rangée vide dans la hauteur est égale au paramètre `width-of-false`+ (la valeur initiale de ce paramètre est égale à 2 pt, qui est la valeur initiale de `\doublerulesep`). Les filets et les fonds colorés ne s'appliqueront pas dans cette rangée.

La lettre `F` est à utiliser dans le préambule des environnements à préambule (comme `{NiceTabular}`, `{NiceArray}`, etc.). Elle crée une colonne de largeur égale à la valeur du paramètre `width-of-false`. Les filets et les fonds colorés ne s'appliqueront pas dans cette colonne. Attention : cette colonne existe bien et il faut donc utiliser deux esperluettes (`&&`) dans chaque rangée où on la croise.

```
\begin{NiceTabular}{cFcc}[hvlines]
\CodeBefore
  \rowcolor[opacity=0.5]{red!50}{3}
  \columncolor[opacity=0.5]{blue!50}{3}
\Body
\diagbox{x}{y} && a & b \\
\FalseRow
a              && b & a \\
b              && a & b
\end{NiceTabular}
```

x \ y	a	b
a	b	a
b	a	b



Nouveau 7.12

Une clé `color-of-false` est proposée pour la couleur de fond de ces fausses colonnes et fausses rangées.

6 Les couleurs de fond 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}`.

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 rangée par rangée, en alternant rectangles colorés, filets et contenu des cases. Le PDF résultant déroute certains lecteurs de PDF et on a parfois des artéfacts 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 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.[26]

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 [options]
  instructions du code-before
\Body
  contenu de l'environnement
\end{pNiceArray}
```

L'argument optionnel entre crochets est une liste de couples *clé=valeur* qui seront présentées au fur et à mesure (les clés disponibles sont `create-cell-nodes`, `sub-matrix` (et ses sous-clés) et `delimiters/color`).

De nouvelles commandes sont disponibles dans ce `\CodeBefore` : `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors`, `\rowlistcolors`, `\chessboardcolors` et `\arraycolor`.[27] Les noms de certaines de ces commandes sont inspirés des noms des commandes de colortbl.

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 à la section 5.3.3, p. 17.

Ces commandes respectent les coins arrondis si la clé `rounded-corners` (cf. 14.2, p. 60) a été utilisée.

Ces commandes colorient également entre les filets des doubles filets (c'est une différence avec colortbl).

Toutes ces commandes acceptent un argument optionnel, entre crochets et en première position. Cet argument optionel peut contenir deux éléments (séparés par une virgule) :

- le modèle colorimétrique (`RGB`, `rgb`, `HTML`, etc.) comme spécifié par l'extension xcolor ;
- une spécification d'opacité selon la forme `opacity = `*`valeur`*.[28]

On détaille maintenant ces différentes commandes.

- La commande `\cellcolor` tient son nom de la commande `\cellcolor` de colortbl.

  Elle prend en arguments obligatoires une couleur et une **liste de cases**, chacune sous le format *i-j* où *i* est le numéro de rangée et *j* le numéro de colonne. Malgré son nom, elle peut aussi colorier une rangée avec la syntaxe *i-* ou bien une colonne avec la syntaxe *-j*.

[26] Si vous utilisez Overleaf, Overleaf effectue automatiquement un nombre de compilations suffisant (en utilisant `latexmk`).

[27] 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. 15.3, p. 69.

[28] Attention : cette fonctionnalité génère des instructions de transparence dans le PDF résultant et certains lecteurs de PDF n'acceptent pas la transparence. L'application de l'opacité est faite par `\pgfsetfillopacity`.

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \cellcolor[HTML]{FFFF88}{3-1,2-2,-3}
\Body
  a & b & c \\
  e & f & g \\
  h & i & j \\
\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}{ccc}[hvlines]
\CodeBefore
  \rectanglecolor{blue!15}{2-2}{3-3}
\Body
  a & b & c \\
  e & f & g \\
  h & i & j \\
\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. 9, p. 37) avec cette couleur. Ce n'est qu'un cas particulier de la commande `\rectanglecolor`.

- 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. 14.5, p. 61).

- 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*). Remarquer que, contrairement à `\rowcolor` de colortbl, elle colorie aussi entre les double-filets.

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

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



- 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 colortbl. 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. L'un des deux arguments de couleur peut être vide (et alors aucune couleur n'est appliquée dans les rangées correspondantes).

  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.[29]
  - 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. 4, p. 6).

```
\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



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

- 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}[rounded-corners]
\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 peut aussi utiliser dans la commande `\rowlistcolors` une série de couleurs définie par la commande `\definecolorseries` de xcolor (et initialisée avec `\resetcolorseries`[30]).

```
\begin{NiceTabular}{c}
\CodeBefore
   \definecolorseries{BlueWhite}{rgb}{last}{blue}{white}
   \resetcolorseries[\value{iRow}]{BlueWhite}
   \rowlistcolors{1}{BlueWhite!!+}
\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 » *east* (E).

[30] Pour l'initialisation, on a utilisé dans l'exemple qui suit le compteur LaTeX `iRow` (qui correspond en interne au compteur TeX `\c@iRow`) qui, quand il est utilisé dans le `\CodeBefore` (ou le `\CodeAfter`) désigne le nombre de rangées du tableau : cf. 14.8, p. 63. Cela permet un ajustement de la gradation des couleurs à la taille du tableau.

```
\begin{NiceTabular}{ccccccc}[corners=E,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



L'exemple précédent utilise les clés `first-row` et `first-col` qui sont décrites dans la partie sur les rangées et colonnes « extérieures » (cf. 9, p. 37).
Comme on le voit, *par défaut*, les commandes de coloriage décrites précédemment ne s'appliquent pas dans ces rangées et colonnes « extérieures ».
Mais on peut *quand même* colorier ces rangées et colonnes en donnant aux commandes précédentes les numéros explicites de ces rangées et colonnes extérieures.
Dans l'exemple suivant, on demande explicitement le coloriage de la colonne 0 (qui est la « première colonne » et qui existe du fait de la clé `first-col`).

```
\begin{NiceTabular}{ccccccc}[corners=E,margin,hvlines,first-row,first-col]
\CodeBefore
  \rowlistcolors{1}{blue!15, }
  \columncolor{red!15}{0}
\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}[no-cell-nodes] % no-cell-nodes is optional
\CodeBefore
  \rowcolor{red!15}{1-2}
  \rowcolors{3}{blue!15}{}
\Body
\toprule
  \Block[C]{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 (qu'il faut avoir chargé).

On peut aussi, dans le `\CodeBefore`, utiliser les commandes `\EmptyColumn` et `\EmptyRow`. La commande `\EmptyColumn` prend en argument une liste de numéros de colonnes et impose qu'aucun coloriage ni tracé de filets n'aura lieu dans les colonnes correspondantes. La commande `\EmptyRow` est similaire.

```
\begin{NiceTabular}{ccccc}[hvlines,no-cell-nodes] % no-cell-nodes est facultatif
\CodeBefore
  \rowcolor{blue!15}{1}
  \EmptyColumn{3}
\Body
  un & deux && trois & quatre \\
  un & \Block{}{deux\\ lignes} && trois & quatre \\
\end{NiceTabular}
```

un	deux		trois	quatre
un	deux lignes		trois	quatre



Ces commandes peuvent servir typiquement à composer plusieurs tableaux côte à côte. Si on souhaite tracer des double-filets, on devrait plutôt considérer la commande `\FalseRow` et la lettre `F` (cf. 5.3.8, p. 25).

6.3 Outils de coloriage en tableau

L'extension nicematrix propose aussi des commandes de coloriage à utiliser directement dans le tableau (comme celles de l'extension colortbl).
Les commandes sont les suivantes (les trois premières sont inspirées par les commandes similaires de colortbl).

- `\cellcolor` qui colorie la case courante[31] ;
- `\rowcolor` à utiliser dans une case et qui colorie le reste de la rangée[32] ;
- `\columncolor` à utiliser dans le préambule du tableau de la même manière que la commande homonyme de colortbl (néanmoins, contrairement à la commande `\columncolor` de colortbl,

[31] Cette commande `\cellcolor` supprimera les espaces qui la suivent (ce que ne fait pas la commande `\cellcolor` de colortbl). De plus, si on définit une fonction au-dessus de `\cellcolor`, il faudra une fonction protégée au sens de TeX (alors que si c'était la commande `\cellcolor` de colortbl, il faudrait au contaire une fonction *fully expandable*).
[32] Si vous souhaitez une commande pour colorier les n rangées suivantes, considérez la commande `\RowStyle` et sa clé `fill`, section 7, p. 33.

celle de nicematrix peut apparaître à l'intérieur d'une autre commande, elle-même utilisée dans le préambule ; en revanche, elle ne prend pas les deux arguments optionnels à la fin entre crochets pour du débord comme la commande de colortbl) ;

- `\rowcolors` qui prend pour arguments deux couleurs et colorie la suite du tableau avec ces deux couleurs (à utiliser après un éventuel `\hline`, `\Hline` ou `\toprule`!) ;
- `\rowlistcolors` qui prend pour argument une liste de couleurs et colorie la suite du tableau avec ces couleurs.[33]

Ces commandes sont compatibles avec les commandes pour les *overlays* de Beamer (comme `\only`, etc.)

```
\NewDocumentCommand { \Blue } { } {\columncolor{blue!15}}
\begin{NiceTabular}{>{\Blue}c>{\Blue}cc}[no-cell-nodes] % no-cell-nodes facultatif
\toprule
  \rowcolor{red!15}
  Nom & Prénom & Année de naissance \\
\midrule
  Achard & Jacques & 5 juin 1962 \\
  Lefebvre & Mathilde & 23 mai 1988 \\
  Vanesse & Stéphanie & 30 octobre 1994 \\
  Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```



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

Chaque utilisation de `\rowlistcolors` (et de `\rowcolors` qui en est un cas particulier) met un terme aux éventuels schémas[34] de coloriage en cours qui auraient été spécifiés par une commande `\rowlistcolors` précédente.
En particulier, on peut engager un coloriage des rangées avec `\rowlistcolors{...}` et l'arrêter par un `\rowlistcolors{}` avec argument vide.

```
\begin{NiceTabular}{c}[hvlines,no-cell-nodes] % no-cell-nodes est facultatif
  un \\
  deux \\
  \rowlistcolors{red!15}
  trois \\
  quatre \\
  cinq \\
  \rowlistcolors{}
  six \\
  sept \\
\end{NiceTabular}
```

un
deux
trois
quatre
cinq
six
sept



Voici un exemple d'utilisation de `\rowcolors` en conjonction avec `create-blocks-in-col`.

[33] Quand la commande `\rowlistcolors` (ou la commande `\rowcolors`) est utilisée dans une case de la colonne j, le coloriage ne s'applique que sur les colonnes au-delà de j (à dessein).

[34] On a écrit *schémas* au pluriel car on peut avoir plusieurs schémas en cours s'ils portent sur des colonnes différentes.

```
\begin{NiceTabular}{lr}[hvlines,create-blocks-in-col=1]
  \rowcolors{blue!10}{}[respect-blocks]
  Pierre    & 12 \\
            & 13 \\
  Jacques   &  8 \\
  Stéphanie & 18 \\
            & 17 \\
            & 15 \\
  Amélie    & 20 \\
  Henri     & 14 \\
  Estelle   & 15 \\
            & 19
\end{NiceTabular}
```

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



6.4 La couleur spéciale « nocolor »

L'extension nicematrix propose la couleur spéciale `nocolor` utilisable dans toutes les commandes de coloriage fournies par nicematrix (dans le `\CodeBefore` ou bien dans le tableau proprement dit).
Les cases marquées par cette couleur ne seront pas coloriées, quelles que soient les autres commandes de coloriage qui auraient pu s'appliquer à ces cases.
La couleur `nocolor` fournit donc un moyen commode de faire des exceptions à l'action d'une commande de coloriage générale.

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. 2, p. 4).
- La clé `fill` (alias : `rowcolor`) fixe la couleur de fond et la clé `opacity`[35] l'opacité de cette couleur de fond. Si la clé `rounded-corners` est utilisée, ce fond aura des coins arrondis.
- La clé `color` fixe la couleur du texte.[36]
- La clé `bold` impose des caractères gras aux éléments de la rangée, qu'ils soient en mode texte ou bien en mode mathématique.

[35] Attention : cette clé génère des instructions de transparence dans le PDF résultant et certains lecteurs de PDF n'acceptent pas la transparence.

[36] La clé `color` utilise la commande `\color` mais insère aussi une instruction `\leavevmode` devant. Cela évite un espace vertical parasite dans les cases qui correspondent à des colonnes de type `p`, `b`, `m`, et `X` (qui débutent en mode vertical de LaTeX). Pour les colonnes de type `V` (de varwidth), cela ne suffit malheureusement pas sauf si on utilise LuaLaTeX avec luacolor (cf. question 460489 sur TeX StackExchange).

```
\begin{NiceTabular}{cccc}[no-cell-nodes]
\hline
  \RowStyle[cell-space-limits=3pt]{\rotate}
  premier & deuxième & troisième & quatrième \\
  \RowStyle[nb-rows=2,color=white,fill=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 à la section 14.6, p. 62.

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 (qui est chargée par nicematrix).

```
\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. 9, p. 37) 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.[37]

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

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

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`.

[37] 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).

```
\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`[38]. L'environnement `{NiceMatrixBlock}` n'a pas de rapport direct avec la commande `\Block` présentée précédemment dans ce document (cf. 4, p. 6).

```
\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}
```

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

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

Nouveau 7.13

On peut souhaiter créer un tableau avec un quadrillage parfaitement carré. Pour cela, nicematrix fournit pour la clé `columns-width` une valeur spéciale `row-height`. Avec cette valeur, les colonnes auront une largeur minimale absolue[39] égale à la hauteur totale (`\dp + \ht`) du *strut* (*étai* en français) utilisé par LaTeX pour régler la hauteur des rangées. Avec l'extension array (qui est chargée par nicematrix), ce *strut* est ajustable avec `\arraystretch` et `\extrarowheight`.

```
\renewcommand{\arraystretch}{1.5}
$\begin{NiceArray}{cccc}[hvlines,columns-width=row-height]
a & b & c & d \\
b & c & d & a \\
c & d & a & b \\
d & a & b & c
\end{NiceArray}$
```

a	b	c	d
b	c	d	a
c	d	a	b
d	a	b	c

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.

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

[39] On a précisé « absolue » pour signifier que l'espace inter-colonnes (`\tabcolsep` ou `\arraycolsep`) est mis à zéro.

```
\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.

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

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 case en question : cf. 15.1.2, p. 67.

Les colonnes `V` de nicematrix acceptent les clés `t`, `p`, `m`, `b`, `l`, `c` et `r` proposées par les colonnes `X` : voir leur description à la section 8.3, p. 36.

Remarquons que l'extension varwidth a quelques problèmes (au moins dans sa version 0.92). Par exemple, avec LuaLaTeX, elle ne fonctionne pas si le contenu commence par une instruction `\color`. De plus, varwidth doit être chargée après l'extension array (elle-même chargée par nicematrix).

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.[40]
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 le paramètre `\linewidth` de LaTeX (et non `\textwidth`).

[40] L'environnement `{tabularx}` capte son contenu comme un argument et le compose plusieurs fois pour calculer la largeur des colonnes. De son côté, `{NiceTabular}` ne capte pas son contenu comme argument (sauf quand la clé `light-syntax` est utilisée) et ne compose qu'une seule fois le tableau mais écrit des informations sur le fichier `aux`, nécessitant ainsi plusieurs compilations.

Pour se rapprocher davantage de l'environnement `{tabularx}`, nicematrix propose aussi un environnement `{NiceTabularX}` avec une syntaxe similaire à celle de `{tabularx}`, c'est-à-dire que la largeur voulue pour le tableau est spécifiée en premier argument (obligatoire).

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 nombre positif comme argument du spécificateur `X`. Par exemple, une colonne `X[0.5]` aura une largeur moitié de celle d'une colonne `X` (qui a un poids de 1).
- On peut spécifier l'alignement horizontal avec une des lettres `l`, `c` et `r` (qui insèrent respectivement `\raggedright`, `\centering` et `\raggedleft` suivi de `\arraybackslash`).[41]
- 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 initiale est `t`.
- Il est possible d'utiliser la clé `V` dans une colonne de type `X`. Quand cette clé est utilisée, la colonne `X` se comporte en fait comme une colonne `V` de l'extension varwidth (que l'utilisateur doit avoir chargée), ce qui fait que la largeur de la colonne calculée par le processus `X` devient la largeur *maximale* de la colonne.

```
\begin{NiceTabular}[width=9cm]{X[c,m]X[0.5,c,m]}[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 ».

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

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

[41] En fait, quand ragged2e est chargée, ce sont les commandes `\RaggedRight`, `\Centering` et `\RaggedLeft` de ragged2e qui sont utilisées, pour un meilleur résultat.

Les lignes pointillées ont été tracées avec les outils qui seront présentés à la section 10, p. 39.

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.[42]
- 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. 14.9, p. 64) 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. Ces clés ont une syntaxe normale mais aussi une syntaxe « additive », ce qui fait que l'on peut écrire, par exemple `code-for-first-row = \color{blue}` mais aussi `code-for-first-row += \color{blue}` (dans ce cas-là, les tokens sont ajoutés à droite de la valeur courante du paramètre).

```
\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}}
$\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col,xdots/nullify]
       & C_1    & \Cdots &        & C_4    &        \\
L_1    & a_{11} & a_{12} & a_{13} & a_{14} & L_1    \\
\Vdots & a_{21} & a_{22} & a_{23} & a_{24} & \Vdots \\
\hline
       & a_{31} & a_{32} & a_{33} & a_{34} &        \\
L_4    & a_{41} & a_{42} & a_{43} & a_{44} & L_4    \\
       & C_1    & \Cdots &        & C_4    &
\end{pNiceArray}$
```

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

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`, 5.3.5, p. 19).

[42] Si on souhaite une colonne extérieure avec un autre type d'alignement, on aura intérêt à considérer la commande `\SubMatrix` disponible dans le `\CodeAfter` et le `\CodeBefore` (cf. 12.2, p. 48) ou bien voir à insérer des délimiteurs directement dans le préambule du tableau (cf. 11, p. 46).

- 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.
- Sans surprise, une éventuelle option `columns-width` (cf. 8, p. 34) 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. 12.2, p. 48).

10 Les lignes de conduite en pointillés dans les matrices

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`.[43]

Chacune de ces commandes doit être utilisée seule dans la case du tableau et elle trace une ligne de conduite en pointillés entre les premières cases non vides[44] 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`.[45]

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

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

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 :

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

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

[43] 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.

[44] La définition précise de ce qui est considéré comme une « case vide » est donnée plus loin (cf. 17.2, p. 73).

[45] 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. 10.5, p. 43.

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 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 `\\` pour l'espacement vertical et la commande `\hspace*` dans une case pour l'espacement horizontal.[46]
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.

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

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

10.1 L'option xdots/nullify

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.

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

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

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

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

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

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 `xdots/nullify`[47] (et une seule instruction `\Ldots` suffit).

[46] 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. 8, p. 34

[47] Cette clé a un alias : `nullify-dots`.

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

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

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

Attention : la clé `xdots/nullify` a un nom qui peut prêter à confusion ; elle n'implique pas que la ligne en pointillés ne sera pas tracée !

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 celles tracées par `\Cdots`, `\Vdots`, etc.
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.

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

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & \ldots & \ldots & \ldots & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

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

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

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ & \ldots & \ldots & \ldots & \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Remarque : Contrairement à la commande `\hdotsfor` de amsmath, la commande `\Hdotsfor` est utilisable même lorsque l'extension colortbl[48] est chargée (néanmoins, dans le cadre de nicematrix, le chargement de colortbl est déconseillé puisque nicematrix propose ses propres commandes pour le coloriage des tableaux : cf. 6.2, p. 26).

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}\NotEmpty  & \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
```

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

```
  & \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}$
```

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

10.3 Comment créer les lignes de conduite 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`.[49]

- L'option `renew-dots`

  Avec cette option, les commandes `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`[43] 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 avec délimiteurs.

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

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

Néanmoins, on remarquera que les environnements de nicematrix ne peuvent pas être imbriqués et, par conséquent, on ne peut pas utiliser `renew-matrix` si on a, par exemple, des matrices `{pmatrix}` imbriquées dans d'autres matrices `{pmatrix}`.

10.4 Les labels des lignes en pointillés

Les commandes `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` et `\Vdotsfor` (ainsi que la commande `\line` dans le `\CodeAfter` : cf. 12.1 p. 48) peuvent en fait prendre trois arguments

[49] 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`.

optionnels spécifiés par les caractères _, ^ et : pour des labels situés au-dessous, au-dessus, ou sur la ligne. Les arguments sont composés en mode mathématique avec `\scriptstyle`.

Le label spécifié par le caractère « : » est en fait composé sur un fond blanc qui est superposé sur la ligne en pointillés (voir l'exemple 18.9, p. 81).

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

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

Avec la clé `horizontal-label`, le label reste horizontal.

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

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

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` : cf. 12.1, p. 48) peuvent être paramétrées par les options suivantes (que l'on met entre crochets après la commande) :

- `nullify` ;
- `horizontal-label(s)` ;
- `color` ;
- `radius` ;
- `shorten-start`$^+$, `shorten-end`$^+$ et `shorten`$^+$;
- `inter` ;
- `number-of-dots` ;
- `line-style`

Pour une commande individuelle, on peut utiliser `horizontal-label` au singulier.

Ces options peuvent aussi être fixées avec `\NiceMatrixOptions` 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/nullify` ;
- `xdots/horizontal-labels` ;
- `xdots/color` ;
- `xdots/radius` ;
- `xdots/shorten-start`$^+$, `xdots/shorten-end`$^+$ et `xdots/shorten`$^+$;
- `xdots/inter` ;
- `xdots/number-of-dots` ;
- `xdots/line-style`.

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

La clé `xdots/nullify` a déjà été décrite à la partie 10.1, p. 40.

La clé `xdots/horizontal-labels` demande que les labels (introduits par `_`, `^` et `:`) restent horizontaux.

L'option `xdots/color` indique bien entendu la couleur de la ligne tracée. On peut définir une couleur « à la volée » (ex. : `xdots/color = { [RGB]{204,204,255} }`). On remarquera que les lignes tracées dans les rangées et colonnes extérieures (décrites plus loin) bénéficient d'un régime spécial : cf. 9, p. 37.

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

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.[50]
Ces clés acceptent une syntaxe normale et une syntaxe « additive », c'est-à-dire que l'on peut écrire, par exemple, `xdots/shorten = 0.6em` ou `xdots/shorten += 4pt` (et on rappelle que la valeur initiale n'est pas nulle !).

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).

Nouveau 7.13

La clé `xdots/number-of-dots` impose le nombre de points, indépendamment de la distance entre les deux cases à relier.

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.[51]

```
\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`.

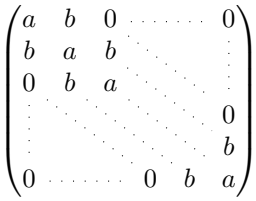
Néanmoins (quand TikZ est chargé), on peut utiliser pour `xdots/line-style` n'importe quel style proposé par TikZ, c'est-à-dire n'importe quelle suite d'options TikZ applicables à un chemin (à l'exception de « `color` », « `shorten >` » et « `shorten <` »).

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

```
$\begin{pNiceMatrix}[xdots={nullify, shorten += 2pt, line-style = loosely dotted}]
a      & b      & 0      &        & \Cdots & 0      \\
b      & a      & b      & \Ddots &        & \Vdots \\
0      & b      & a      & \Ddots &        &        \\
       & \Ddots & \Ddots & \Ddots &        & 0      \\
\Vdots &        &        &        &        & b      \\
0      & \Cdots &        & 0      & b      & a
\end{pNiceMatrix}$
```

[50] En fait, quand on utilise ces clés au niveau de `\NiceMatrixOptions` ou bien d'un environnement, seules les extrémités des lignes qui s'arrêtent au niveau d'un contenu non vide de case sont concernées. Quand on les utilise sur une commande `\Cdots` (ou `\Vdots`, etc.), toutes les extrémités sont concernées.

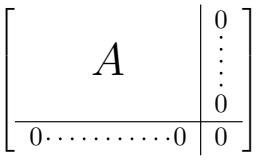
[51] La raison de départ est que le format PDF comporte un système de description de lignes en tiretés, qui, puisqu'il est incorporé dans le PDF, est affiché très rapidement par les lecteurs de PDF. Il est facile à partir de ce type de ligne de créer des lignes de points carrés alors qu'une ligne de 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



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).[52]

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



10.7 Les commandes `\Hbrace` et `\Vbrace`

Puisque, comme dit dans la partie précédente, il est possible d'utiliser, avec les commandes `\Cdots`, `\Ldots`, `\Vdots`, etc. n'importe quel style de ligne fourni par TikZ, on peut envisager de tracer des accolades avec la décoration `brace` fournie par la bibliothèque decorations.pathreplacing de TikZ.

Pour faciliter cet usage, nicematrix propose les deux commandes `\Hbrace` et `\Vbrace`. Celles-ci ne sont disponibles que si TikZ, ainsi que sa bibliothèque decorations.pathreplacing, ont été chargées (avant ou après le chargement de nicematrix). Si elles ne sont pas chargées, une erreur (non fatale) sera levée.

```
\usepackage{tikz}
\usetikzlibrary{decorations.pathreplacing}
```

Les commandes `\Hbrace` et `\Vbrace` ont la même syntaxe. Elles prennent trois arguments :

- un premier argument optionnel (entre crochets) pour une liste de couples *clé=valeur* : les clés autorisées sont `color`, `horizontal-label`, `shorten`, `shorten-start` et `shorten-end`.

  Il existe aussi une clé `brace-shift`$^+$ pour déplacer l'accolade (et son label) vers l'extérieur de la matrice. Cette clé accepte une syntaxe normale et une syntaxe « additive », c'est-à-dire que l'on peut écrire, par exemple, `brace-shift = 3pt` ou `brace-shift += 2pt`.

- un deuxième argument, obligatoire, qui est le nombre de colonnes (pour `\Hbrace`) ou de rangées (pour `\Vbrace`) sur lesquelles l'accolade va s'étendre.

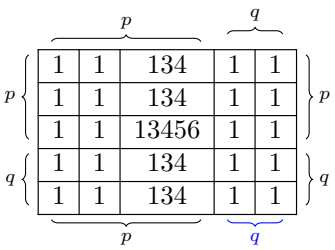
- un troisième argument, obligatoire, qui est le label de l'accolade.

Concernant la commande `\Hbrace`, son comportement vis-à-vis des esperluettes (`&`) est le même que celui des commandes `\multicolumn`, `\hdotsfor`, `\Hdotsfor` (fournie par nicematrix), etc. : on ne doit mettre qu'une seule esperluette après la commande, même si l'accolade s'étend sur plusieurs colonnes.

```
\usepackage{tikz}                       % dans le préambule
\usetikzlibrary{decorations.pathreplacing} % dans le préambule
```

[52] En revanche, la commande `\line` dans le `\CodeAfter` (cf. 12.1, p. 48) ne crée pas de bloc.

```
$\begin{NiceArray}{ccccc}%
  [ hvlines ,
    first-row ,
    last-row = 6,
    first-col ,
    last-col ,
    xdots/horizontal-labels ]
& \Hbrace{3}{p} & \Hbrace[brace-shift=1mm]{2}{q} \\
\Vbrace{3}{p} & 1 & 1 & 134 & 1 & 1 & \Vbrace{3}{p} \\
& 1 & 1 & 134 & 1 & 1 \\
& 1 & 1 & 13456 & 1 & 1 \\
\Vbrace{2}{q} & 1 & 1 & 134 & 1 & 1 & \Vbrace{2}{q}\\
& 1 & 1 & 134 & 1 & 1 \\
& \Hbrace{3}{p} & \Hbrace[color=blue]{2}{q} \\
\end{NiceArray}$
```



Le style TikZ utilisé par nicematrix pour tracer ces accolades est appelé `nicematrix/brace`. Voici la valeur initiale de ce style :

```
decoration = { brace , raise = -0.15 em } ,
decorate
```

L'utilisateur final peut changer ce style TikZ à sa guise (en respectant la structure avec les clés `decorate` et `decoration`).

Pour un autre exemple d'utilisation de `\Hbrace` et `\Vbrace`, voir 18.8, p. 80.

Il est également possible de dessiner des accolades du mode mathématique de LaTeX en utilisant les commandes `\OverBrace` et `\UnderBrace` (cf. 12.3, p. 51) et la commande `\SubMatrix` (cf. 12.2, p. 48) dans le `\CodeAfter`.

11 Délimiteurs dans le préambule de l'environnement

Pour les environnements à préambule (`{NiceArray}`, `{pNiceArray}`, etc.), il est possible de placer des délimiteurs verticaux directement dans le préambule.[53]

Les délimiteurs ouvrants doivent être précédés du mot-clé `\left` et les délimiteurs fermants du mot-clé `\right`. Les mots-clés `\left` et `\right` n'ont pas d'obligation à être utilisés par paires.

Tous les délimiteurs extensibles de LaTeX peuvent être utilisés.

Voici un exemple qui utilise `\lgroup` et `\rgroup`.

```
$\begin{NiceArray}{\left\lgroup ccc\right\rgroup l}
1 & 2 & 3 &
4 & 1 & 6 &
7 & 8 & 9 & \scriptstyle L_3 \gets L_3 + L_1 + L_2
\end{NiceArray}$
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 6 \\ 7 & 8 & 9 \end{pmatrix}_{L_3 \leftarrow L_3+L_1+L_2}$$

Pour cet exemple, on aurait aussi pu utiliser `{NiceArrayWithDelims}` (cf. 14.11, p. 65) et la clé `last-col` (cf. 9, p. 37).

[53] Cette syntaxe est inspirée de l'extension blkarray.

Il y a un cas particulier : pour les délimiteurs `(`, `[`, `\{`[54] et `\lbrace`, et les délimiteurs fermants correspondants, les préfixes `\left` et `\right` sont facultatifs.[55]

Voici un exemple avec un délimiteur `\{` à gauche dans un `{NiceTabular}` (on remarquera la compatibilité avec la clé `t`).

```
On définit $f$ par\quad
\begin{NiceTabular}[t]{\{ll}
$f(x) = 0$ & si $x$ est négatif \\
$f(x) = 1-e^x$ & si $x$ est positif
\end{NiceTabular}
```

On définit f par $\begin{cases} f(x) = 0 & \text{si } x \text{ est négatif} \\ f(x) = 1 - e^x & \text{si } x \text{ est positif} \end{cases}$

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.

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

$$\left(\begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}\left(\begin{matrix} a_{12} \\ \displaystyle\int_0^1 \dfrac{1}{x^2+1}\,dx \\ a_{32} \end{matrix}\right)\begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}\right)$$

Pour des constructions plus complexes, avec en particulier des délimiteurs ne couvrant pas toutes les rangées, on aura intérêt à considérer la commande `\SubMatrix` disponible dans le `\CodeAfter` et le `\CodeBefore` (cf. 12.2, p. 48).

12 Le \CodeAfter

L'extension nicematrix propose dans ses environnements un clé `code-after` pour spécifier 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` (la clé `code-after` reste bien sûr obligatoire dans `\AutoNiceMatrix` et les commandes similaires). Bien que ce soit un mot-clé, `\CodeAfter` accepte quand même un argument optionnel (entre crochets).[56]

Les utilisateurs expérimentés peuvent, en particulier, utiliser les nœuds PGF-TikZ créés par nicematrix dans le `\CodeAfter` (cf. 15, p. 65).

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

Remarque : Depuis la version 7.11a de nicematrix, le `\CodeAfter` est exécuté en mode mathématique, ce qui permet de nullifier les éventuels espaces insérés avec, par exemple, une commande `\SubMatrix`.

[54] Pour les accolades, la protection par la contre-oblique est obligatoire (c'est pourquoi on a écrit `\{`).

[55] Pour les délimiteurs `[` et `]`, les préfixes restent obligatoires en cas de conflit de notation avec des crochets d'options de certains descripteurs de colonnes.

[56] Les clés autorisées dans cet argument optionnel sont les suivantes : `delimiters/color`, `rules` et ses sous-clés, `sub-matrix` (en lien avec la commande `\SubMatrix`) et ses sous-clés et `xdots` (pour la commande `\line`) et ses sous-clés.

Nouveau 7.13

La clé `overlap` contrôle si les éléments graphiques créés dans le `\CodeAfter` sont pris en compte ou non dans la *bounding box* globale de l'environnement (`{NiceTabular}`, `{NiceMatrix}`, etc.).
Quand la clé `overlap` possède la valeur `true`, ces éléments ne sont *pas* pris en compte et peuvent « déborder » de la *bounding box* et se superposer au reste du document LaTeX. À l'opposé, quand la clé `overlap` possède la valeur `false`, il n'y pas d'*overlaping* car les éléments graphiques ajoutés dans le `\CodeAfter` sont pris en compte dans la *bounding box*.
La valeur initiale du paramètre `overlap` est `true` (et non pas `false`).

Remarque : Pour des raisons techniques, le réglage `overlap=false` n'est pas disponible dans le schéma de compilation historique `latex-dvips-ps2pdf` (ou Adobe Distiller).

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

La commande `\line` permet de tracer directement des lignes en pointillés entre les cases. Ces lignes en pointillés sont de même nature que celles tracées par les commandes `\Cdots`, `\Vdots`, etc. (cf. 10, p. 39). La commande `\line` 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 rangée et *j* est le numéro de colonne ;
- 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. 10.5, p. 43).

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

```
$\NiceMatrixOptions{xdots/shorten += 0.3 em}
\begin{pNiceMatrix}
I       & 0      & \Cdots  & 0      \\
0       & I      & \Ddots  & \Vdots \\
\Vdots  & \Ddots & I       & 0      \\
0       & \Cdots & 0       & I
\CodeAfter \line{2-2}{3-3}
\end{pNiceMatrix}$
```

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

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. 17.1, p. 72).

```
$\begin{bNiceMatrix}
1      & \Cdots &   & 1      & 2      & \Cdots          & 2      \\
0      & \Ddots &   & \Vdots & \Vdots & \hspace*{2.5cm} & \Vdots \\
\Vdots & \Ddots &   &        &        &                 &        \\
0      & \Cdots & 0 & 1      & 2      & \Cdots          & 2
\CodeAfter \line[shorten=6pt]{1-5}{4-7}
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & \cdots & 1 & 2 & \cdots & 2 \\ 0 & \ddots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \ddots & & & & \\ 0 & \cdots & 0 & 1 & 2 & \cdots & 2 \end{bmatrix}$$

12.2 La commande `\SubMatrix` dans le `\CodeAfter` (et le `\CodeBefore`)

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`, `\lgroup`, `\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 spécificateur `last` est utilisable) ;
- 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*.[57]

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.

```
$\begin{NiceArray}{ccc@{\hspace{1.5em}}c}[cell-space-limits += 2pt, margin]
 1           & 1           & 1           & x \\
\dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\
 1           & 2           & 3           & z
\CodeAfter
  \SubMatrix({1-1}{3-3})
  \SubMatrix({1-4}{3-4})
\end{NiceArray}$
```

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

En fait, dans cet exemple, il aurait sans doute été plus simple de mettre des délimiteurs directement dans le préambule de l'environnement `{NiceArray}` (cf. 11, p. 46) avec la construction suivante.

```
$\begin{NiceArray}{(ccc)(c)}[cell-space-limits += 2pt]
 1           & 1           & 1           & x \\
\dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\
 1           & 2           & 3           & z
\end{NiceArray}$
```

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

La commande `\SubMatrix` accepte en fait également deux arguments optionnels spécifiés par les symboles traditionnels `^` et `_` pour des éléments en exposant et en indice (mais aucun espace n'est réservé pour ces éléments et c'est pourquoi, dans l'exemple suivant, on a utilisé `right-margin`.).

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

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

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`) ;

[57] 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}]`).

- `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 ;
- `code` permet d'insérer du code, en particulier du code TikZ, après la construction de la matrice. Cette clé est décrite en détail plus loin.

On remarquera que tous les 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.

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

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

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

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

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

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 (cf. 15.4, p. 70).

La commande `\SubMatrix` est en fait aussi disponible dans le `\CodeBefore`. L'intérêt d'utiliser `\SubMatrix` dans le `\CodeBefore` est que les délimiteurs tracés par ces commandes `\SubMatrix` sont alors prises en compte pour limiter les lignes en pointillés continues (créées par `\Cdots`, `\Vdots`, etc.) qui ont une extrémité ouverte.
Pour un exemple, voir 18.11, p. 87.

En dépit de son nom, la commande `\SubMatrix` peut également être utilisée dans `{NiceTabular}` comme dans l'exemple suivant (qui utilise `\bottomrule` et `\toprule` de l'extension booktabs).

```
\begin{NiceTabular}{@{}ll@{}}
\toprule
Part A              & the first part \\
\Block{2-1}{Part B} & a first sub-part \\
                    & a second sub-part \\
\bottomrule
\CodeAfter
  \SubMatrix{\lbrace}{2-2}{3-2}{.}
\end{NiceTabular}
```

Part A	the first part
Part B	a first sub-part
	a second sub-part



▷ Pour tracer des accolades verticales, nicematrix propose aussi une commande `\Vbrace` à utiliser directement dans le tableau et qui trace des accolades avec TikZ (alors qu'avec `\SubMatrix`, c'est une accolade du mode mathématique de LaTeX qui est utilisée) : cf. 10.7, p. 45.

Attention : La fonctionnalité suivante est fragile et ne fonctionne pas avec `latex–dvips–ps2pdf`.

La clé `code` de la commande `\SubMatrix` permet d'insérer du code après la création de la matrice. Elle a surtout pour vocation d'être utilisée pour insérer des instructions TikZ, sachant que, dans les instructions TikZ insérées dans cette clé, les nœuds de la forme `i-j.anchor` ou `i-|j` sont interprétés avec `i` et `j` étant des numéros de rangée et colonne *relatifs à la sous-matrice*.[58]

```
$\begin{NiceArray}{ccc@{}w{c}{5mm}@{}ccc}
    &   &   && -1 & 1  & 2  \\
    &   &   &&  0 & 3  & 4  \\
    &   &   &&  0 & 0  & 5  \\
  1 & 2 & 3 && -1 & 7  & 25 \\
  0 & 4 & 5 && 0  & 12 & 41 \\
  0 & 0 & 6 && 0  & 0  & 30
\CodeAfter
  \NewDocumentCommand{\MyDraw}{}{\tikz \draw [blue] (2-|1) -| (3-|2) -| (4-|3) ;}
  \SubMatrix({1-5}{3-7})[code = \MyDraw]
  \SubMatrix({4-1}{6-3})[code = \MyDraw]
  \SubMatrix({4-5}{6-7})[code = \MyDraw]
\end{NiceArray}$
```



$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{pmatrix} \begin{pmatrix} -1 & 1 & 2 \\ 0 & 3 & 4 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} -1 & 7 & 25 \\ 0 & 12 & 41 \\ 0 & 0 & 30 \end{pmatrix}$$

Comme on le voit, le tracé effectué par la commande `\MyDraw` est *relatif* à la sous-matrice à laquelle elle s'applique.

12.3 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 cases 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 ;

[58] Attention : la syntaxe `j|-i` n'est *pas* autorisée.

- 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`).

  Il est possible de mettre des commandes `\\` dans ce dernier argument pour formater le label sur plusieurs lignes de texte.

Par défaut, aucun espace vertical n'est réservé pour l'accolade et son label. Pour que ce soit le cas, il faut utiliser le réglage `overlap=false`.

```
Voici un exemple :

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

Voici un exemple :

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

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.

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

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

▷ Les commandes `\OverBrace` et `\UnderBrace` utilisent les accolades standard du mode mathématique de LaTeX comme les commandes `\overbrace` et `\underbrace` de LaTeX dont elles s'inspirent. Néanmoins, nicematrix propose aussi une commande `\Hbrace` à utiliser directement en tableau et qui utilise des accolades tracées par TikZ (cf. 10.7, p. 45).

12.4 La commande `\TikzEveryCell`

La commande `\TikzEveryCell` exécute avec TikZ le chemin rectangulaire qui correspond à chaque case du tableau, avec comme paramètres TikZ l'argument de `\TikzEveryCell`. Cet argument doit être une liste de couples *clé=valeur* applicables à un chemin TikZ. En fait, cette commande s'applique à chaque case du tableau, exceptées celles situées dans les rangées et colonnes extérieures (cf. 9, p. 37) et celles situées dans les coins vides (quand la clé `corners` est utilisée : cf. 5.3.3, p. 17). Elle s'applique en fait aussi à chaque bloc (sauf ceux qui ont la clé `transparent`) et ne s'applique pas aux cases individuelles situées dans ces blocs.

En fait, dans la liste des clés passée en argument on peut mettre une clé `offset`. Cette clé n'est *pas* fournie par TikZ mais par nicematrix. Elle réduit le rectangle correspondant au bloc par une marge (horizontalement et verticalement) égale à la valeur (passée à `offset`). C'est ce rectangle réduit qui sera le chemin exécuté par TikZ avec comme options les autres clés de l'argument de `\TikzEveryCell`.

```
\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{ccc}[corners]
  & \Block{1-2}{colonnes} \\
  \Block{2-1}{rangées}
  & case 1 1 & case 1 2 \\
  & case 2 1 & case 2 2
\CodeAfter
  \TikzEveryCell{offset=1pt,draw}
\end{NiceTabular}
```

	colonnes	
rangées	case 1 1	case 1 2
	case 2 1	case 2 2

La commande `\TikzEveryCell` est, en fait, également disponible dans le `\CodeBefore` :

```
\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{ccc}[corners]
\CodeBefore
  \TikzEveryCell
    {
      offset=1pt,
      fill=blue!15,
      rounded corners=2pt
    }
\Body
  & \Block{1-2}{colonnes} \\
  \Block{2-1}{rangées}
  & case 1 1 & case 1 2 \\
  & case 2 1 & case 2 2
\end{NiceTabular}
```

	colonnes	
rangées	case 1 1	case 1 2
	case 2 1	case 2 2

La commande `\TikzEveryCell` possède deux clés, utilisables en argument optionnel, entre crochets.

- avec la clé `empty`, la commande ne s'applique qu'aux cases vides (les cases considérées comme vides sont décrites à la partie 17.2, p. 73) ;
- avec la clé `not-empty`, à l'opposé, la commande ne s'applique qu'aux cases non vides.

```
\renewcommand{\arraystretch}{1.4}
\begin{NiceTabular}{cccccc}[hvlines]
  P & O & U & R & V & U \\
  O &   &   & E & I &   \\
  M & O & R & F & A & L \\
  E & T & A & L &   & E \\
  L & A & S & E & R & S \\
  O &   & E & X & I & T
\CodeAfter
  \TikzEveryCell[empty]{fill=gray,draw}
\end{NiceTabular}
```

P	O	U	R	V	U
O			E	I	
M	O	R	F	A	L
E	T	A	L		E
L	A	S	E	R	S
O		E	X	I	T

```
\begin{NiceTabular}{ccccc}
1 & 2 & 3 & 4 &   \\
1 &   &   & 4 & 5 \\
1 &   &   &   & 5 \\
1 & 2 &   & 4 & 5 \\
1 & 2 & 3 & 4 & 5 \\
\CodeAfter
  \TikzEveryCell[not-empty]{draw}
\end{NiceTabular}
```

1	2	3	4	
1			4	5
1				5
1	2		4	5
1	2	3	4	5

13 Les légendes et les notes dans les tableaux

13.1 La légendes des tableaux

L'environnement `{NiceTabular}` propose des clés `caption`, `short-caption` et `label` à utiliser lorsque le tableau est inséré dans un environnement flottant (comme un environnement `{table}`).

L'intérêt d'utiliser cette clé `caption` plutôt que la commande classique `\caption` est que la légende, si elle est longue, est justifiée à la largeur du tableau (hors éventuelles colonnes extérieures spécifiées par `first-col` et `last-col` : cf. 9, p. 37). Il n'y a pas besoin d'avoir recours à l'extension threeparttable ou l'extension floatrow.

Par défaut, la légende est placée au-dessous du tableau. Pour avoir la légende placée au-dessus, il convient d'utiliser la clé `caption-above` dans `\NiceMatrixOptions` (elle n'est pas disponible dans un environnement individuel).

La clé `short-caption` correspond à l'argument optionnel de la commande classique `\caption` et la clé `label` correspond bien sûr à la commande `\label`.

Voir table 1, p. 56, un exemple d'utilisation des clés `caption` et `label`.

Ces fonctionnalités sont compatibles avec l'extension caption. Par exemple, il est possible d'utiliser le réglage `\captionsetup{justification=raggedright,singlelinecheck=false}` pour avoir la légende alignée à gauche, même si la légende est courte.

13.2 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.

13.3 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 pour les lignes de texte égale à la largeur du tableau (hors éventuelles colonnes extérieures spécifiées par `first-col` et `last-col` : cf. 9, p. 37). Sans surprise, cette commande n'est disponible que dans les environnements de tableaux : `{NiceTabular}`, `{NiceTabular*}` et `{NiceTabularX}`.
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 cette extension enumitem. Si enumitem n'a pas été chargée, une erreur sera levée à la première utilisation de `\tabularnote`.

```
\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 (ou dans la commande `\captionof` de l'extension caption). Sans surprise, il est également possible de l'utiliser dans la légende rentrée avec la clé `caption` de l'environnement `{NiceTabular}`.
- 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) et que le mode d'alignement de la colonne est `c` ou `r`, 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.

  Une syntaxe alternative est proposée : il est possible d'utiliser l'environnement `{TabularNote}` à la fin de l'environnement `{NiceTabular}` (mais *avant* l'éventuel `\CodeAfter`).
- 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.
- 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`.[59]
- Il est possible de référencer une note de tableau (avec la commande `\label` placée après le `\tabularnote`).
- La commande `\tabularnote` admet un argument optionnel (entre crochets) qui permet de changer le symbole de l'appel de note.

  Exemple : `\tabularnote[$\star$]{Une note...}`.

Voir sur la table 1, p. 56, certaines de ces remarques illustrées. Cette table a été composée avec le code suivant (l'extension caption a été chargée dans ce document).

[59] Pour des raisons techniques, il n'est pas autorisé de mettre plusieurs `\tabularnote` avec exactement le même argument dans la légende. Ce n'est pas vraiment contraignant.

```
\begin{table}[hbt]
\centering
\NiceMatrixOptions{caption-above}
\begin{NiceTabular}{@{}llc@{}}%
  [
   caption = Un tableau dont la légende a été rentrée avec la clé \texttt{caption}%
             \tabularnote[$\star$]{On peut mettre une note dans la légende.} ,
   label = t:tabularnote,
   tabularnote = Un peu de texte avant les notes. ,
   notes/bottomrule
  ]
\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}
```



TAB. 1 : Un tableau dont la légende a été rentrée avec la clé `caption`*

Nom	Prénom	Durée de vie
Barrère	Bertrand	86
Nightingale[a,b]	Florence[c]	90
Schœlcher	Victor	89[d]
Touchet	Marie[c]	89
Wallis	John	87

Un peu de texte avant les notes.

* On peut mettre une note dans la légende.

[a] Souvent considérée comme la première infirmière.

[b] Surnommée « la Dame à la Lampe ».

[c] Cette note est commune à deux appels de notes.

[d] L'appel de note déborde à droite.

13.4 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
- notes/no-print

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}

- 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 : `\textsuperscript{#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 = 0pt`

  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. 56).

  La clé `notes/enumitem-keys` fournie par nicematrix permet de 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).

  Cette clé admet une syntaxe normale mais aussi une syntaxe « additive », c'est-à-dire que l'on peut écrire, par exemple, `notes/code-before += \footnotesize` (les tokens rajoutés sont alors rajoutés à *droite* de la valeur courante du paramètre).

- La clé `notes/detect-duplicates` active la détection des commandes `\tabularnote` qui ont le même argument (et alors une seule note est écrite sous le tableau correspondant à plusieurs appels de notes).

  Valeur initiale : `true`

- Quand la clé `notes/no-print` est utilisée, les notes ne sont *pas composées* par nicematrix. Néanmoins, l'utilisateur final peut utiliser la commande `\NiceTabularNotes` pour les insérer à l'endroit souhaité (après la composition du tableau).

  Valeur initiale : `false`

Pour un exemple de personnalisation des notes de tableau, l'exemple 18.1, p. 75.

14 Autres fonctionnalités

14.1 Arbres

La clé `draw-trees-in-col` prend en argument une liste de numéros de colonnes et, dans chacune de ces colonnes, dessine des arbres comme en proposent les extensions dirtree et dirtreex.

```
\usepackage{booktabs} % dans le préambule

\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{w{l}{2.5mm}w{l}{2.5mm}lr}[draw-trees-in-col=1-2]
\toprule
  Category &&&  \% \\
\midrule
  Animal   &&& 100 \\
  & Human   &&  50 \\
  && Man     &  20 \\
  && Woman   &  30 \\
  & Fox     &&  30 \\
  && Vixen   &  16 \\
  && Dog     &  14 \\
  & Chicken &&  20 \\
  && Cock    &   8 \\
  && Hen     &  12 \\
\bottomrule
\end{NiceTabular}
```



Category	%
Animal	100
├── Human	50
│ ├── Man	20
│ └── Woman	30
├── Fox	30
│ ├── Vixen	16
│ └── Dog	14
└── Chicken	20
├── Cock	8
└── Hen	12

Trois clés `trees/line-width`, `trees/color` et `trees/rounded-corners` sont fournies pour la personnalisation de ces traits (le préfixe `trees` est factorisable).

```
\renewcommand{\arraystretch}{1.3}
\NiceMatrixOptions{trees= {line-width = 0.4pt, color = red , rounded-corners}}
\begin{NiceTabular}{w{l}{2.5mm}w{l}{2.5mm}lr}[draw-trees-in-col=1-2]
\toprule
  Category &&&  \% \\
\midrule
  Animal   &&& 100 \\
  & Human   &&  50 \\
  && Man     &  20 \\
  && Woman   &  30 \\
  & Fox     &&  30 \\
```

```
  && Vixen    &  16 \\
  && Dog      &  14 \\
  & Chicken &&  20 \\
  && Cock     &   8 \\
  && Hen      &  12 \\
\bottomrule
\end{NiceTabular}
```



Category	%
Animal	100
Human	50
Man	20
Woman	30
Fox	30
Vixen	16
Dog	14
Chicken	20
Cock	8
Hen	12

14.2 La clé rounded-corners des environnements

La clé `rounded-corners` que l'on décrit maintenant n'a pas de lien direct avec la clé `corners` (qui sert à spécifier les « coins vides ») décrite à la partie 5.3.3, p. 17. Elle est en revanche proche de la clé `rounded-corners` de la commande `\Block` (cf. 4.1, p. 7).

La clé `rounded-corners` spécifie que le tableau ou la matrice devra avoir des coins arrondis avec un rayon égal à la valeur de cette clé (la valeur par défaut est 4 pt[60]). Plus précisément, cette clé a deux effets que l'on décrit maintenant.

- Toutes les commandes de coloriage de cases, colonnes et rangées (que ce soit dans le `\CodeBefore` ou bien directement dans le tableau) respectent ces coins arrondis pour le tableau.
- Quand la clé `hvlines` est utilisée, les filets extérieurs sont tracés avec des coins arrondis.[61]

Cette clé est disponible dans tous les environnements et commandes de nicematrix (comme par exemple `\pAutoNiceMatrix`) et également dans `\NiceMatrixOptions`.

```
\begin{NiceTabular}{ccc}[hvlines,rounded-corners]
\CodeBefore
  \rowcolor{red!15}{1}
\Body
  Nom & Prénom & Profession \\
  Arvy & Jacques & Dentiste \\
  Jalon & Amandine & Dentiste \\
\end{NiceTabular}
```

Nom	Prénom	Profession
Arvy	Jacques	Dentiste
Jalon	Amandine	Dentiste



[60] Cette valeur est la valeur par défaut des « `rounded corners` » de PGF/TikZ.

[61] Bien sûr, lorsqu'il s'agit d'un environnement avec des délimiteurs (`{pNiceArray}`, `{pNiceMatrix}`, etc.) la clé `hvlines` ne trace pas les filets extérieurs.

14.3 La commande \ShowCellNames

La commande `\ShowCellNames`, utilisable dans le `\CodeBefore` et le `\CodeAfter` affiche le nom (sous la forme *i*-*j*) de chaque case. Elle est fournie comme un outil de travail pour aider lors de la conception d'un tableau.

Quand elle est utilisée dans le `\CodeBefore`, les noms sont affichés après les éventuels fonds colorés, mais avant le contenu des cases.

```
\begin{NiceTabular}{ccc}[hvlines,cell-space-limits=3pt]
\CodeBefore
  \ShowCellNames
  \rowcolor{gray!15}{3}
\Body
  \Block{2-2}{} &          & test \\
                &          & blabla \\
                & du texte & rien
\end{NiceTabular}
```

1-1 1-2		1-3 test
2-1 2-2		2-3 blabla
3-1	du texte	3-3 rien

Quand elle est utilisée dans le `\CodeAfter`, cette commande applique un rectangle blanc semi-transparent pour estomper le tableau (attention : certains lecteurs de PDF ne prennent pas en charge la transparence) et les noms des cases sont affichés par-dessus.

```
\begin{NiceTabular}{ccc}[hvlines,cell-space-limits=3pt]
\CodeBefore
  \rowcolor{gray!15}{3}
\Body
  \Block{2-2}{} &          & test \\
                &          & blabla \\
                & du texte & rien
\CodeAfter \ShowCellNames
\end{NiceTabular}
```

1-1 1-2		1-3 test
2-1 2-2		2-3 blabla
3-1	3-2 du texte	3-3 rien

14.4 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.

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

$$\begin{array}{c} C_1 \cdots\cdots\cdots\cdots\cdots C_n \\ \left(\begin{array}{ccc} 2{,}3 & 0 \cdots\cdots\cdots & 0 \\ 12{,}4 & \vdots & \vdots \\ 1{,}45 & \vdots & \vdots \\ 7{,}2 & 0 \cdots\cdots\cdots & 0 \end{array}\right) \end{array}$$

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

14.5 Type de colonne par défaut dans {NiceMatrix}

Les environnements sans préambule (`{NiceMatrix}`, `{pNiceMatrix}`, etc.) ainsi que la commande `\pAutoNiceMatrix` et ses variantes, acceptent la clé `columns-type` qui indique le type de 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 syntaxe est alors similaire à celle de l'environnement `{bmatrix*}`, fourni par l'extension mathtools.

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`.

14.6 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`.[62]

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

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

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

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

La commande `\rotate` accepte une option `c` entre crochets : `\rotate[c]` (les espaces sont supprimés après `\rotate[c]`). Quand cette clé est utilisée, le contenu, après rotation, est composé dans une `\vcenter` (une primitive de TeX), ce qui fait que, le plus souvent, on obtiendra un centrage vertical.

La commande `\rotate` admet aussi une clé nommée `-90`. Avec cette clé, le contenu subit une rotation d'angle $-90°$ au lieu d'une rotation d'angle $90°$.

Attention : la commande `\rotate` est prévue pour être utilisée dans un `\Block` ou bien dans des colonnes de type `l`, `c`, `r`, `w` ou `W` ; si elle est utilisée dans un autre type de colonne (commme `p{...}`), les résultats ne seront peut-être pas ceux attendus. On aura certainement intérêt le cas échéant à considérer la commande `\rotatebox` de l'extension graphicx (extension chargée par nicematrix).

14.7 La clé 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).

[62] On peut aussi l'utiliser dans `\RowStyle` (cf. 7, p. 33).

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

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

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.

Quand la clé `small` est active, certaines fonctionnalités de nicematrix ne sont plus disponibles : par exemple, il n'est plus possible de mettre des délimiteurs directement dans le préambule d'un environnement avec préambule (cf. 11, p. 46).

14.8 `\AutoNiceMatrix` et les compteurs iRow et jCol

Dans les cases du tableau, il est possible d'utiliser les compteurs LaTeX `iRow` et `jCol`[63] qui représentent le numéro de la rangée courante et le numéro de la colonne courante. 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. Bien entendu, l'utilisateur ne doit pas modifier les valeurs de ces compteurs `iRow` et `jCol` qui sont utilisés en interne par nicematrix.
Dans le `\CodeBefore` (cf. 6.2, p. 26) et dans le `\CodeAfter` (cf. 12, p. 47), `iRow` représente le nombre total de rangées (hors éventuelles rangées extérieures : cf. 9, p. 37) et `jCol` le nombre total de colonnes (hors potentielles colonnes extérieures).

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

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

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.

[63] Il s'agit bien de compteurs LaTeX, ce qui fait que les compteurs TeX sous-jacents sont `\c@iRow` et `\c@jCol`.

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*-*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). Ces commandes prennent aussi un argument optionnel pour des options en première position et également un autre en dernière position (il est en particulier possible d'utiliser les clés `code-before` et `code-after` parmi ces options).

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

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

Il existe aussi `\AutoNiceArrayWithDelims` similaire à `{NiceArrayWithDelims}`.

14.9 La clé 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.

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

$$\begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{bmatrix} 2\cos a & \cos a + \cos b \\ \cos a + \cos b & 2\cos b \end{bmatrix} \end{array}$$

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.[64]

La clé `light-syntax-expanded` a le même comportement que la clé `light-syntax` mais avec cette différence que le corps de l'environnement est complètement développé (au sens de TeX[65]) avant découpe en rangées (mais après l'extraction de l'éventuel `\CodeAfter`).

14.10 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. 12.2, p. 48) et aux délimiteurs spécifiés directement dans le préambule des environnements à préambule (cf. 11, p. 46).

[64] 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.
[65] Plus précisément, il s'agit d'une expansion de type `e` de L3.

14.11 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.

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

$$\left\downarrow \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \right\uparrow$$

14.12 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 est ni dans les rangées extérieures, ni dans les colonnes extérieures. Si elle est utilisée hors d'un environnement de nicematrix, elle est sans effet.
Cette commande est disponible dans tous les environnements fournis par nicematrix (et aurait pu s'appler \OnlyMainNiceTabular).
Pour des exemples d'utilisation, voir tex.stackexchange.com/questions/488566 et également tex.stackexchange.com/questions/73412.

15 Utilisation explicite de TikZ avec nicematrix

15.1 Les nœuds correspondant aux contenus des cases

L'extension nicematrix crée un nœud PGF-TikZ[66] pour chaque **contenu** de case non vide du tableau considéré. Ces nœuds, qui par défaut, sont invisibles pour l'utilisateur, sont utilisés, entre autres, pour tracer les lignes en pointillés entre les cases du tableau.

Par défaut, ces nœuds ne sont pas disponibles dans le \CodeBefore (pour accélerer les compilations). Si on veut les utiliser dans le \CodeBefore, on doit utiliser la clé create-cell-nodes du mot-clé \CodeBefore.

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.[67]

Comme la création de ces nœuds requiert du temps et de la mémoire, il est possible de la désactiver ponctuellement avec la clé no-cell-nodes pour accélérer les compilations. Attention toutefois : ces nœuds sont utilisés en interne par certaines fonctionnalités de nicematrix. On ne peut donc utiliser no-cell-nodes que si on n'utilise pas ces fonctionnalités, parmi lesquelles figurent la clé corners et les commandes de lignes pointillées continues (\Cdots, etc.).

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.

[66] On rappelle que TikZ est une sur-couche de PGF. L'extension nicematrix charge PGF et ne charge pas TikZ. On parle de « nœud PGF-TikZ » pour rappeler que, en fait, les nœuds créés par nicematrix avec PGF sont en fait aussi utilisables avec TikZ. L'utilisateur final préférera sans doute les utiliser avec TikZ qu'avec PGF.

[67] 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. 10, p. 39) et la détermination des « coins » (cf. 5.3.3, p. 17).

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`[68]. 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 (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.

```
$\begin{pNiceMatrix}[name = ma-matrice]
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
\end{pNiceMatrix}$
\tikz[remember picture,overlay]
     \draw (ma-matrice-2-2) circle (2mm) ;
```

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

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.

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

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

Les nœuds de la dernière colonne (hors éventuelle « colonne extérieure » spécifiée par `last-col`[69]) peuvent aussi être désignés par i-`last`. De même, les nœuds de la dernière rangée peuvent être désignés par `last`-j.

Dans l'exemple suivant, nous avons encadré tous les nœuds de la matrice.

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

Puisque ces nœuds sont des nœuds PGF, on ne sera pas étonné d'appendre qu'ils sont tracés en utilisant un style PGF spécifique. Ce style est nommé `nicematrix/cell-node` et sa définition dans le fichier source `nicematrix.sty` est la suivante :

```
\pgfset
  {
    nicematrix / cell-node /.style =
     {
       inner sep = 0 pt ,
       minimum width = 0 pt
     }
  }
```

L'utilisateur peut modifier ce style en changeant les valeurs des clés `text/rotate`, `inner xsep`, `inner ysep`, `inner sep`, `outer xsep`, `outer ysep`, `outer sep`, `minimum width`, `minimum height` et `minimum size`.

Pour un exemple d'utilisation, voir la partie 18.12, p. 88.

[68] La valeur passée à la clé `name` est *développée*, au sens de TeX.

[69] Pour les colonnes extérieures, cf. 9, p. 37.

15.1.1 La clé pgf-node-code

Pour les utilisateurs expérimentés, nicematrix fournit la clé `pgf-node-code` qui correspond à du code PGF qui sera exécuté à la création, par PGF, des nœuds correspondants aux cases du tableau. Plus précisément, la valeur fournie à la clé `pgf-node-code` sera passée en cinquième argument de la commande `\pgfnode`. Cette valeur doit contenir au moins une instruction comme `\pgfusepath`, `\pgfusepathqstroke`, `\pgfusepathqfill`, etc.

15.1.2 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 case située dans une colonne de type `V`, le nœud PGF-TikZ créé par nicematrix pour le contenu de cette case a une largeur ajustée au contenu de cette case. 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.

```
\begin{NiceTabular}{V{10cm}}
  \bfseries \large
  Titre \\
  \lipsum[1][1-4]
\CodeAfter
  \tikz \draw [rounded corners] (1-1) -| (last-|2) -- (last-|1) |- (1-1) ;
\end{NiceTabular}
```

Titre

Lorem ipsum dolor sit amet, consectetuer 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, à la partie 15.3, p. 69.
Bien sûr, il existe des extensions spécialisées dans ce type de constructions graphiques, comme par exemple tcolorbox.

15.2 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`.[70]

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.[71]

[70] Il existe aussi l'option `create-extra-nodes` qui est un alias pour la conjonction de `create-medium-nodes` et `create-large-nodes`.
[71] Il n'y a pas de « nœuds larges » créés dans les rangées et colonnes extérieures (pour ces rangées et colonnes, voir 9, p. 37).

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

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`.[72]

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

Il est aussi possible d'ajouter de l'espace sur les côtés du tableau avec les clés `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}{w{l}{2cm}ll}[hvlines]
  fraise & amande & abricot \\
  prune & pêche & poire  \\[1ex]
  noix & noisette & brugnon
\end{NiceTabular}
```

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Ci-contre, on a colorié toutes les cases de ce tableau avec `\chessboardcolors` (cf. 6.2, p. 27).

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 à la section 6.2, p. 26), de manière à accélérer la compilation.
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 ; ce mécanisme n'est pas actif par défaut pour des questions d'efficacité).

[72] 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`.

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

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



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

15.3 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 des nœuds nommés i`.1`, i`.2`, i`.25`, i`.3`, i`.4`, i`.5`, i`.6`, i`.7`, i`.75`, i`.8` et i`.9` intermédiaires entre le nœud i et le nœud $i+1$.

Ces nœuds sont accessibles dans le `\CodeAfter` mais aussi dans le `\CodeBefore`.

	tulipe	lys
arum		violette mauve
muguet	dahlia	

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}$
```



$$\begin{array}{ccccccccc}
1 & & & & & & & & \\
1 & 1 & & & & & & & \\
1 & 2 & 1 & & & & & & \\
1 & 3 & 3 & 1 & & & & & \\
1 & 4 & 6 & 4 & 1 & & & & \\
1 & 5 & 10 & 10 & 5 & 1 & & & \\
1 & 6 & 15 & 20 & 15 & 6 & 1 & & \\
1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 & \\
1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1
\end{array}$$

Les nœuds « décimaux » (comme $i.4$) peuvent être utilisés par exemple pour barrer une rangée (si on a chargé TikZ).

```
$\begin{pNiceArray}{ccc|c}
  2 & 1 & 3 & 0 \\
  3 & 3 & 1 & 0 \\
  3 & 3 & 1 & 0
\CodeAfter
  \tikz \draw [red] (3.4-|1) -- (3.4-|last) ;
\end{pNiceArray}$
```

$$\left(\begin{array}{ccc|c}
2 & 1 & 3 & 0 \\
3 & 3 & 1 & 0 \\
3 & 3 & 1 & 0
\end{array}\right)$$



15.4 Les nœuds correspondant aux commandes \SubMatrix

La commande `\SubMatrix` disponible dans le `\CodeAfter` a été présentée à la section 12.2, p. 48.

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}\}`).

$$\left(\begin{array}{cccc}
121 & 23 & 345 & 345 \\
45 & \left\{\begin{array}{cc} 346 & 863 \\ 38458 & 34 \end{array}\right\} & & 444 \\
3462 & & & 294 \\
34 & 7 & 78 & 309
\end{array}\right)$$

16 API pour les développeurs

16.1 Jeux de clés publics

L'extension nicematrix propose plusieurs jeux de clés publics :

- `nicematrix/environments` qui s'applique à tous les environnements de nicematrix ;
- `nicematrix/NiceMatrix` qui s'applique spécifiquement à l'environnement `{NiceMatrix}` et à ses variantes avec délimiteurs (`{pNiceMatrix}`, etc.) ;
- `nicematrix/NiceTabular` qui s'applique spécifiquement aux environnements `{NiceTabular}`, `{NiceTabularX}` et `{NiceTabular*}`.

Par exemple, si on veut pouvoir fixer le paramètre `\arraystretch` avec une clé `arraystretch` dans les environnements de nicematrix, on peut utiliser la définition suivante dans le préambule du document LaTeX.

```
\DeclareKeys[nicematrix/environments]{arraystretch.store=\arraystretch}
```

On peut alors écrire :

```
\begin{NiceTabular}{cc}[hvlines,arraystretch=2]
  a & b \\
  c & d
\end{NiceTabular}
```

a	b
c	d

16.2 CodeBefore et CodeAfter

L'extension nicematrix fournit deux variables internes mais publiques[73] :

- `\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 case 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_before_tl`. On utilise `\g_nicematrix_code_before_tl` au lieu de `\g_nicematrix_code_after_tl` pour que la croix soit tracée avant les éventuels filets verticaux (de manière à avoir un résultat parfait).

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

\NewDocumentCommand \crossbox { ! O { } }
  {
    \tl_gput_right:Ne \g_nicematrix_code_before_tl
      {
        \__pantigny_crossbox:nnn
          { \arabic { iRow } }
          { \arabic { jCol } }
          { \exp_not:n { #1 } }
```

[73] 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.

```
      }
    \ignorespaces
  }
\ExplSyntaxOff
```

On a utilisé les compteurs LaTeX `iRow` et `jCol` fournis par nicematrix (cf. 14.8, p. 63).

Voici un exemple d'utilisation :

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \arraycolor{gray!10}
\Body
  merlan & requin & cabillaud \\
  baleine & \crossbox[red] & morue \\
  mante & raie & poule
\end{NiceTabular}
```

merlan	requin	cabillaud
baleine		morue
mante	raie	poule

17 Remarques techniques

Première remarque : l'extension nicematrix doit être chargée après l'extension underscore. Si elle est chargée avant, une erreur sera levée.

17.1 Lignes diagonales

Par défaut, toutes les lignes de conduite pointillées qui sont diagonales[74] 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) :

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

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

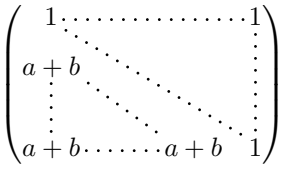
```
$\begin{pNiceMatrix}
  1      & \Cdots &        & 1      \\
  a+b    &        &        & \Vdots \\
  \Vdots & \Ddots & \Ddots & \\
  a+b    & \Cdots & a+b    & 1
\end{pNiceMatrix}$
```

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

[74] On parle des lignes créées par `\Ddots` et `\Iddots` et non des lignes créées par une commande `\line` dans le `\CodeAfter`.

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

Le même exemple sans parallélisation :



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]`.

17.2 Les cases « vides »

L'extension nicematrix utilise en plusieurs occassions le concept de « case vide ». Par exemple, une instruction comme `\Ldots`, `\Cdots`, etc. essaye de déterminer la première case non vide de part et d'autre de la case considérée. De même, quand la clé `corners` (cf. 5.3.3, p. 17) est utilisée, les coins qui sont déterminés sont composés de cases vides.
La définition précise de « case vide » est la suivante.

- 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.

- Pour les colonnes de type `p`, `m`, `b`, `V`[75] ou `X`[76], la case est vide si (et seulement si) son contenu dans le codage TeX est vide (il n'y a que des espaces entre les deux esperluettes `&`).

- Pour les colonnes de type `c`, `l`, `r`, `w{...}{...}` ou `W{...}{...}`, la case est vide si (et seulement si) son rendu TeX est de largeur nulle.

- Une case qui contient la commande `\NotEmpty` est non vide (et un nœud PGF-TikZ est créé pour cette case).

- Toute case située dans un bloc (commande `\Block`) qui utilise la clé `empty` est considérée comme vide.

- Une case avec seulement 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.

17.3 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 idée[77]. L'environnement `{matrix}` et ses variantes (`{pmatrix}`, `{vmatrix}`, etc.) de amsmath préfèrent supprimer ces espaces

[75] Les colonnes de type `V` sont fournies par l'extension varwidth, qui doit être chargée : cf. 8.2, p. 35.
[76] Pour les colonnes `X`, voir 8.3, p. 36
[77] Dans la documentation de l'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).*

avec des instructions explicites `\hskip -\arraycolsep`[78]. 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}` de 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).

17.4 Incompatibilités

L'extension nicematrix ne peut pas être utilisée dans revtex4-2 car cette classe n'est plus compatible avec les versions récentes de LaTeX. En effet, revtex4-2 modifie des commandes internes à l'extension array et ces modifications n'ont pas été mises à jour pour tenir compte des modifications récentes de array liées à la génération de PDF balisé.

Il peut y avoir des incompatibilités de nicematrix avec babel pour les langues qui activent (au sens de TeX) certains caractères, en particulier le caractère `<`.
Par exemple, pour l'espagnol, il vaut mieux désactiver au chargement les abréviations avec :
`\usepackage[spanish,es-noshorthands]{babel}`

L'extension nicematrix est incompatible avec certaines classes qui redéfinissent les environnements `{tabular}` et `{array}`. C'est en particulier le cas de la classe `socg-lipics-v2021`. Néanmoins, dans ce cas-là, il est possible de charger la classe avec l'option `notab` qui requiert justement que `{tabular}` ne soit pas redéfini.

L'extension nicematrix n'est pas compatible avec la classe ieeeaccess car cette classe n'est pas compatible avec PGF-TikZ. Il existe néanmoins une parade simple qui consiste à écrire :[79]
```
\let\TeXyear\year
\documentclass{IEEEaccess}
\let\year\TeXyear
```

Pour pouvoir utiliser nicematrix avec la classe aastex631 (de l'*American Astronomical Society*), on doit ajouter dans le préambule du fichier les instructions suivantes :
```
\BeforeBegin{NiceTabular}{\let\begin\BeginEnvironment\let\end\EndEnvironment}
\BeforeBegin{NiceArray}{\let\begin\BeginEnvironment}
\BeforeBegin{NiceMatrix}{\let\begin\BeginEnvironment}
```

L'extension nicematrix n'est pas parfaitement compatible avec les classes et extensions de LuaTeX-ja : la détection des coins vides (cf. 5.3.3, p. 17) risque d'être erronée dans certaines circonstances.

L'extension nicematrix n'est pas compatible avec l'extension arydshln et les colonnes `V` de l'extension boldline ne sont pas prises en charge (car la lettre `V` est réservée pour les colonnes `V` de varwidth). De toutes manières, nicematrix fournit des outils pour tracer des lignes en tiretés (commande `\hdashedline` et lettre `;`) ou de différentes épaisseurs, grâce à la clé `custom-line` (cf. 5.3.5, p. 19).

Les colonnes `d` de l'extension dcolumn et les colonnes `n` de l'extension numprint ne sont pas prises en compte (mais on peut utiliser les colonnes `S` de siunitx).

17.5 Compatibilité avec le Tagging Project de LaTeX

Depuis la version 7.0, l'extension nicematrix est compatible avec le *Tagging Project* de LaTeX visant à la création automatique de PDF balisés. Pour le moment, seuls les tableaux simples sont correctement balisés.
Voici un exemple de code qui va produire un PDF correctement balisé.

[78] 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.
[79] Voir https://tex.stackexchange.com/questions/528975

```
\DocumentMetadata{tagging = on}
\documentclass{article}
\usepackage{lmodern}
\usepackage{nicematrix}

\begin{document}

\begin{center}
\tagpdfsetup{table/header-rows=1}
\begin{NiceTabular}{ccc}[hvlines]
  First name & Last name & Age \\
  Paul       & Imbert    & $66$ \\
  John       & Sarrus    & $23$ \\
  Liz        & Taylor    & $100$ \\
  George     & Adams     & $34$
\end{NiceTabular}
\end{center}

\end{document}
```



18 Exemples

18.1 Notes dans les tableaux

Les outils de nicematrix pour les notes dans les tableaux ont été présentés à la partie 13 p. 54.

Imaginons que l'on souhaite numéroter les notes de tableau (celles construites avec `\tabularnote`) avec des astérisques.[80]
On commence par écrire une commande `\stars` similaire aux commandes classiques `\arabic`, `\alph`, `\Alph`, etc. mais qui produit un nombre d'astérisques égal à son argument.[81]

```
\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 =
     {
       style = \stars{#1} ,
       enumitem-keys =
        {
          widest* = \value{tabularnote} ,
          align = right
        }
     }
  }
```

[80] Bien entendu, il faut qu'il y en ait très peu : trois paraît un maximum.
[81] Ou plutôt : à la valeur de son argument.

```
\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 \\
  Lefèbvre\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*	Jacques	5 juin 1962
Lefèbvre**	Mathilde	23 mai 1988
Vanesse	Stéphanie	30 octobre 1994
Dupont	Chantal	15 janvier 1998

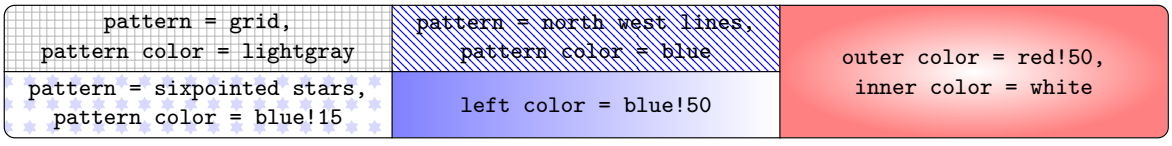
*La famille Achard est une très ancienne famille du Poitou.
**Le patronyme Lefebvre est une altération de Lefébure.

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

La clé `tikz` de la commande `\Block` n'est disponible que lorsque TikZ est chargé.[82]

```
\usepackage{tikz}        % dans le préambule
\usetikzlibrary{patterns} % dans le préambule

\ttfamily \small
\begin{NiceTabular}{X[m]X[m]X[m]}[hvlines,cell-space-limits=3pt,rounded-corners]
  \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}
```



Dans l'exemple suivant, on utilise la clé `tikz` pour hachurer une rangée du tableau. On remarquera que l'on utilise la clé `transparent` de la commande `\Block` pour que les filets soient tracés dans le bloc.[83]

[82] Par défaut, nicematrix ne charge que PGF, qui est une sous-couche de TikZ.
[83] Par défaut, les filets ne sont pas tracés dans les blocs créés avec la commande `\Block` (cf. 5 p. 14) : la clé `transparent` permet qu'ils le soient (le bloc devient ainsi *transparent* aux filets extérieurs).

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \columncolor[RGB]{169,208,142}{2}
\Body
  un & deux & trois \\
  \Block[transparent,tikz={pattern = north west lines, pattern color = red}]{1-*}{}
  quatre & cinq & six \\
  sept & huit & neuf
\end{NiceTabular}
```

un	deux	trois
quatre	cinq	six
sept	huit	neuf

18.3 Utilisation avec tcolorbox

Voici un exemple d'utilisation de `{NiceTabular}` dans une commande `\tcbox` de tcolorbox. On a utilisé la clé `hvlines-except-borders` pour faire afficher tous les filets sauf ceux sur les bords (qui sont, bien entendu, ajoutés par tcolorbox).

```
\tcbset
  {
    colframe = blue!50!black ,
    colback = white ,
    fonttitle = \bfseries ,
    nobeforeafter ,
    center title
  }

\tcbox
  [
    left = 0mm ,
    right = 0mm ,
    top = 0mm ,
    bottom = 0mm ,
    boxsep = 0mm ,
    toptitle = 0.5mm ,
    bottomtitle = 0.5mm ,
    title = My table
  ]
  {
    \renewcommand{\arraystretch}{1.2}% <-- the % is mandatory here
    \begin{NiceTabular}{rcl}[hvlines-except-borders,rules/color=blue!50!black]
    \CodeBefore
      \rowcolor{red!15}{1}
    \Body
      One & Two & Three \\
      Men & Mice & Lions \\
      Upper & Middle & Lower
    \end{NiceTabular}
  }
```

My table		
One	Two	Three
Men	Mice	Lions
Upper	Middle	Lower

18.4 Un tableau de signes

```
\color{blue}
\[\begin{NiceArray}
  [columns-width=1cm,hvlines,rules/color=black]
  {>{\color{black}}cccccc}
  f(x)   & \Block[C]{1-5}{+} \\
  f'(x)  & \Block[C]{1-2}{+} && \Block[C]{1-2}{+} && + \\
  f''(x) & - & \Block[C]{1-2}{-} && \Block[C]{1-2}{+} \\
\end{NiceArray}\]
```



$f(x)$		+			
$f'(x)$	+		+		+
$f''(x)$	−		−		+

La clé `C` de la commande `\Block` permet un centrage absolu du contenu du bloc par rapport aux limites des colonnes (alors qu'avec la clé `c`, on a un centrage par rapport aux *contenus* des colonnes correspondantes et ici il y a beaucoup de cellules sans contenu).

18.5 Un tableau de variations

Pour le tableau de variations suivant, on trace les flèches (ainsi que quelques traits auxiliaires) avec TikZ en utilisant les nœuds PGF/TikZ créés par nicematrix (cf. 15, p. 65).

```
$\begin{NiceArray}{r|cw{c}{1cm}cw{c}{1cm}c}[cell-space-limits=2pt]
x     & 0 & - & \frac{1}{e} & +  & +\infty \\
\Hline
f'(x) &   & - & 0 & + & \\
\Hline
      & 1 &   &   &   & +\infty \\
f     &   &   &   &   &         \\
      &   &   & (1/e)^{1/e} \\
\CodeAfter
  \begin{tikzpicture}
  \draw (3-|4.5) -- (5-|4.5) ;
  \draw ([xshift=3pt]2-|2) -- ([xshift=3pt]3-|2) ;
  \draw [->,red,shorten < = 3pt] (3-2) -- (5-4) ;
  \draw [->,red,shorten < = 3pt] (5-4) -- (3-6) ;
  \end{tikzpicture}
\end{NiceArray}$
```


x	0	$-$	$\frac{1}{e}$	$+$	$+\infty$
$f'(x)$		$-$	0	$+$	
f	1	↘	$(1/e)^{1/e}$	↗	$+\infty$

18.6 Exemple de description d'une table de base de données

Dans l'exemple suivant, on utilise l'extension hhline (on aurait aussi pu utiliser ehhline) dans un environnement `{NiceTabular}` pour tracer le double trait horizontal traditionnel qui marque la clé primaire de la table.

```
\usepackage{hhline} % dans le préambule
```

```
\begin{NiceTabular}{>{\columncolor{red!15}}cllr}[vlines]
\Hline
clé  & prénom & nom      & âge \\
\hhline{=---}
2072 & Jean   & Marcier  & 68  \\
2073 & Claude & Beaumont & 23  \\
\Hline
\end{NiceTabular}
```

clé	prénom	nom	âge
2072	Jean	Marcier	68
2073	Claude	Beaumont	23



Comme on le voit l'utilisation de la clé `vlines` a permis d'écrire `\hhline{=---}` et non `\hhline{|=|-|-|-|}`.

18.7 Exemple d'utilisation de la clé borders de \Block

Dans l'exemple suivant, on utilise des commandes `\Block` pour tracer des filets.

```
\usepackage{tikz} % dans le préambule

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



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

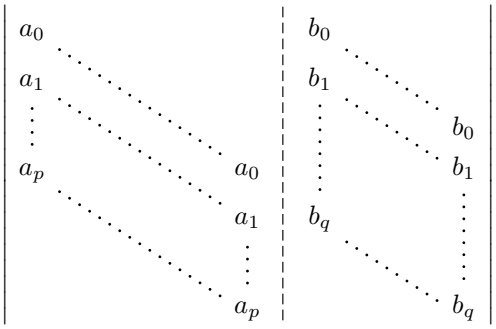
18.8 Lignes de conduite

Un exemple pour le résultant de deux polynômes (cet exemple nécessite que TikZ soit chargé pour que la lettre « ; » soit disponible).

```
\usepackage{tikz} % dans le préambule

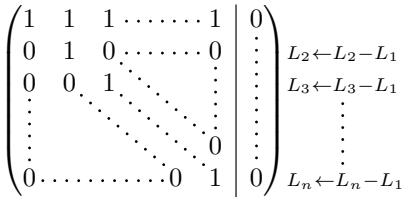

\setlength{\extrarowheight}{1mm}
$\begin{vNiceArray}{cccc;ccc}[columns-width=6mm, xdots/shorten += 3pt]
  a_0   &      &&       &b_0    &      &     \\
  a_1   &\Ddots&&       &b_1    &\Ddots&     \\
  \Vdots&\Ddots&&       &\Vdots &\Ddots&b_0  \\
  a_p   &      &&a_0    &       &      &b_1   \\
        &\Ddots&&a_1    &b_q    &      &\Vdots\\
        &      &&\Vdots &       &\Ddots&      \\
        &      &&a_p    &       &      &b_q
\end{vNiceArray}$
```





Un exemple avec un système linéaire. Rappelons que `xdots/nullify` annule les espaces horizontaux et verticaux créés par les commandes `\Cdots`, `\Vdots`, etc. dans le tableau (mais les lignes sont quand même tracées, en dépit du nom !).

```
$\begin{pNiceMatrix}%
  [vlines=-1,xdots/nullify,last-col,code-for-last-col=\scriptstyle]
  1       & 1 & 1 &\Cdots &   & 1       & 0       & \\
  0       & 1 & 0 &\Cdots &   & 0       &         & L_2 \gets L_2-L_1 \\
  0       & 0 & 1 &\Ddots &   & \Vdots &         & L_3 \gets L_3-L_1 \\
          &   &   &\Ddots &   &         & \Vdots & \Vdots \\
  \Vdots &   &   &\Ddots &   & 0       & \\
  0       &   &   &\Cdots & 0 & 1       & 0       & L_n \gets L_n-L_1
\end{pNiceMatrix}$
```

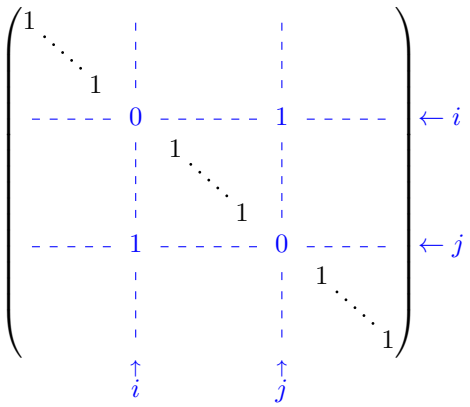


L'option `line-style` permet de changer le style des lignes de conduite tracées par `\Ldots`, `\Cdots`, `\Vdots`, etc. On peut de ce fait tracer des lignes qui ne sont plus pointillées.

```
\usepackage{tikz}             % dans le préambule
\usetikzlibrary{decorations}  % dans le préambule

\setcounter{MaxMatrixCols}{12}
\newcommand{\blue}{\color{blue}}
\[
\begin{pNiceMatrix}%
   [last-row,last-col,xdots={nullify,line-style={dashed,dash expand off,blue}}]
  1& & & \Vdots & & & & \Vdots \\
  & \Ddots[line-style=standard] \\
  & & 1 \\
  \Cdots & & & \blue 0 & \Cdots & & & \blue 1 & & & \Cdots & \blue \leftarrow i \\
  & & & & 1 \\
  & & &\Vdots & & \Ddots[line-style=standard] & & \Vdots \\
  & & & & & & 1 \\
  \Cdots & & & \blue 1 & \Cdots & & \Cdots & \blue 0 & & & \Cdots & \blue \leftarrow j \\
  & & & & & & & & 1 \\
  & & & & & & & & & \Ddots[line-style=standard] \\
  & & & \Vdots & & & & \Vdots & & & 1 \\
  & & & \blue \overset{\uparrow}{i} & & & & \blue \overset{\uparrow}{j} \\
\end{pNiceMatrix}
\]
```

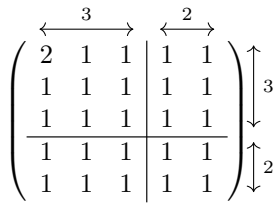




18.9 Illustrer les nombres de lignes et de colonnes d'une matrice

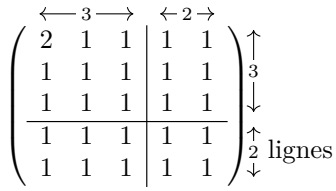
Dans ce document, la bibliothèque arrows.meta de TikZ a été chargée, ce qui a une incidence sur la forme des pointes de flèches.

```
\NiceMatrixOptions{xdots={horizontal-labels,line-style = <->}}
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hdotsfor{3}^{3} & \Hdotsfor{2}^{2} \\
  2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}^{3}\\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
  \Hline
  1 & 1 & 1 & 1 & 1 & \Vdotsfor{2}^{2}\\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```



Si on veut mettre les labels sur les flèches, il convient d'utiliser le caractère spécial « : » au lieu de « ^ » :

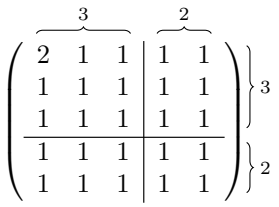
```
\NiceMatrixOptions{xdots={horizontal-labels,line-style = <->}}
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hdotsfor{3}:{3} & \Hdotsfor{2}:{2} \\
  2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}:{3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
  \Hline
  1 & 1 & 1 & 1 & 1 &
  \Vdotsfor{2}:{2 \rlap{ \smash{lignes}}} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```



Si on préfère des accolades comme celles proposées par la bibliothèque decorations.pathreplacing de TikZ, le mieux est d'utiliser les commandes \Hbrace et \Vbrace proposées par nicematrix dans ce but (cf. 10.7, p. 45).

```
\usepackage{tikz}                          % dans le préambule
\usetikzlibrary{decorations.pathreplacing} % dans le préambule
```

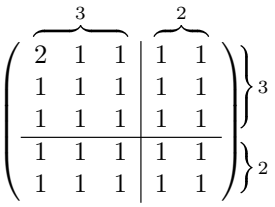
```
\NiceMatrixOptions{xdots/horizontal-labels}
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hbrace{3}{3} & \Hbrace{2}{2} \\
  2 & 1 & 1 & 1 & 1 & \Vbrace{3}{3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
  \Hline
  1 & 1 & 1 & 1 & 1 & \Vbrace{2}{2} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```



Il est possible de changer le style TikZ des accolades en changeant le style `nicematrix/brace`. L'exemple suivant utilise des accolades « calligraphiques » (l'extension TikZ calligraphy doit être chargée *après* decorations.pathreplacing).

```
\usepackage{tikz}                                        % dans le préambule
\usetikzlibrary{decorations.pathreplacing,calligraphy} % dans le préambule


\NiceMatrixOptions{xdots/horizontal-labels}
\tikzset
  {
    nicematrix/brace/.style =
      {
        decoration =
          {
            calligraphic brace ,
            amplitude = 0.4 em ,
            raise = -0.25 em
          } ,
        line width = 0.1 em ,
        decorate
      }
  }
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hbrace{3}{3} & \Hbrace{2}{2} \\
  2 & 1 & 1 & 1 & 1 & \Vbrace{3}{3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
  \Hline
  1 & 1 & 1 & 1 & 1 & \Vbrace{2}{2} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```



Si on préfère les accolades de la fonte mathématique courante, il convient d'utiliser les commandes `\SubMatrix`, `\OverBrace` et `\UnderBrace` dans le `\CodeAfter`.

```
$\begin{pNiceArray}{ccc|cc}[margin,last-col]
  2 & 1 & 1 & 1 & 1 & \Block{3-1}{\quad 3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
  \Hline
  1 & 1 & 1 & 1 & 1 & \Block{2-1}{\quad 2} \\
  1 & 1 & 1 & 1 & 1 \\
\CodeAfter
  \OverBrace[shorten,yshift=1.5mm]{1-1}{1-3}{3}
  \OverBrace[shorten,yshift=1.5mm]{1-4}{1-5}{2}
  \SubMatrix{.}{1-1}{3-5}{\rbrace}[xshift=3.5mm]
  \SubMatrix{.}{4-1}{5-5}{\rbrace}[xshift=3.5mm]
\end{pNiceArray}$
```

$$\begin{pNiceArray}{ccc|cc}[margin,last-col]
  2 & 1 & 1 & 1 & 1 & \Block{3-1}{\quad 3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
  \Hline
  1 & 1 & 1 & 1 & 1 & \Block{2-1}{\quad 2} \\
  1 & 1 & 1 & 1 & 1 \\
\CodeAfter
  \OverBrace[shorten,yshift=1.5mm]{1-1}{1-3}{3}
  \OverBrace[shorten,yshift=1.5mm]{1-4}{1-5}{2}
  \SubMatrix{.}{1-1}{3-5}{\rbrace}[xshift=3.5mm]
  \SubMatrix{.}{4-1}{5-5}{\rbrace}[xshift=3.5mm]
\end{pNiceArray}$$



18.10 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
  {
    light-syntax, no-cell-nodes , % no-cell-nodes est facultatif
    last-col, code-for-last-col = \color{blue}\scriptstyle,
    vlines = 5 ,
    matrix/columns-type = r
  }
\setlength{\extrarowheight}{1mm}

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

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

\smallskip
\quad $\begin{pNiceMatrix}
12 -8   7 5  3 ;
0  64 -41 1 19 ;
0   0   0 0  0 { L_3 \gets 3 L_2 + L_3 }
\end{pNiceMatrix}$

\smallskip
\quad $\begin{pNiceMatrix}
12 -8   7 5  3 {} ;
0  64 -41 1 19    ;
\end{pNiceMatrix}$
\end{NiceMatrixBlock}
```



$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

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

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \\ \\ \\ \\ \\ \\ L_3 \leftarrow 3L_2 + L_3 \end{array}$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

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,
    vlines = 5 ,
    matrix/columns-type = r ,
    no-cell-nodes % facultatif
  }
\setlength{\extrarowheight}{1mm}

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

\smallskip
\quad $\begin{pNiceMatrix}
12  -8   7  5   3                       ;
0   64 -41  1  19  { L_2 \gets L_1-4L_2  } ;
0 -192 123 -3 -57  { L_3 \gets L_1+4L_3  } ;
0 -64   41 -1 -19  { L_4 \gets 3L_1-4L_4 } ;
\end{pNiceMatrix}$
etc.
\end{NiceMatrixBlock}
```



$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

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

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \\ \\ L_3 \leftarrow 3L_2+L_3 \end{array}$$
$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

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



$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$
$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right) \begin{array}{l} \\ L_2 \leftarrow L_1-4L_2 \\ L_3 \leftarrow L_1+4L_3 \\ L_4 \leftarrow 3L_1-4L_4 \end{array}$$
$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \\ \\ L_3 \leftarrow 3L_2+L_3 \end{array}$$
$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

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 (et les colonnes seront espacées pour réserver une place pour ces filets).

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



$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

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

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) L_3 \leftarrow 3L_2+L_3$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

18.11 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-case[84]).

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

```
$\begin{pNiceArray}{>{\strut}cccc}[margin,rules/color=blue,no-cell-nodes]
  \Block[draw]{}{a_{11}} & a_{12} & a_{13} & a_{14} \\
  a_{21} & \Block[draw]{}{a_{22}} & a_{23} & a_{24} \\
  a_{31} & a_{32} & \Block[draw]{}{a_{33}} & a_{34} \\
  a_{41} & a_{42} & a_{43} & \Block[draw]{}{a_{44}} \\
\end{pNiceArray}$
```

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

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`, `hlines-except-borders` et `hvlines-except-borders` « écartent » les composantes de la matrice.[85]

Il est possible de colorier une rangée avec `\rowcolor` dans le `\CodeBefore`, ou avec `\rowcolor` dans la première case d'une rangée (quand elle est utilisée dans une autre case, la commande `\rowcolor` colorie jusqu'à la fin de la rangée).

```
$\begin{pNiceArray}{>{\strut}cccc}[margin, extra-margin=2pt,no-cell-nodes]
  \rowcolor{red!15}A_{11} & A_{12} & A_{13} & A_{14} \\
  A_{21} & \rowcolor{red!15}A_{22} & A_{23} & A_{24} \\
  A_{31} & A_{32} & \rowcolor{red!15}A_{33} & A_{34} \\
  A_{41} & A_{42} & A_{43} & \rowcolor{red!15}A_{44}
\end{pNiceArray}$
```

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

Pour un réglage plus fin, on peut utiliser la commande `\Block`.

```
\usepackage{tikz} % dans le préambule

\NewDocumentCommand{\OneRow}{}
  {\Block[tikz={xoffset=2.5pt,yoffset=0.8pt,fill=red!15,rounded corners = 2pt}]{1-*}{}}
$\begin{pNiceArray}{ccc}[last-col, margin = 2pt]
\OneRow a & a + b & a + b + c & L_1 \\
\OneRow a & a     & a + b     & L_2 \\
\OneRow a & a     & a         & L_3
\end{pNiceArray}$
```

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

Cette technique n'est pas parfois pas suffisante quand on veut surligner une partie d'une ligne ou bien une partie d'une colonne.
C'est pourquoi on propose maintenant une technique plus générale.
On commence par charger la bibliothèque fit de TikZ et on définit un style TikZ nommé `highlight`.

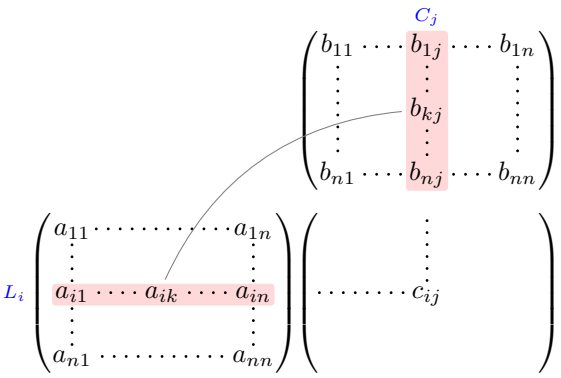
[85] Pour la commande `\cline`, voir la remarque à la section 5.1.2, p. 15.

```
\usepackage{tikz} % dans le préambule
\usetikzlibrary{fit} % dans le préambule

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

On utilise alors ce style dans le `\CodeBefore` de l'environnement.

```
\[\begin{NiceArray}{*{6}{c}@{\hspace{6mm}}*{5}{c}}[xdots/nullify]
\CodeBefore [create-cell-nodes]
  \SubMatrix({2-7}{6-last})
  \SubMatrix({7-2}{last-6})
  \SubMatrix({7-7}{last-last})
  \begin{tikzpicture}
    \node [highlight = (9-2) (9-6)] { } ;
    \node [highlight = (2-9) (6-9)] { } ;
  \end{tikzpicture}
\Body
    &        &        &        &        &        &        & \color{blue}\scriptstyle C_j \\
    &        &        &        &        & b_{11} & \Cdots & b_{1j} & \Cdots & b_{1n} \\
    &        &        &        &        & \Vdots &        & \Vdots &        & \Vdots \\
    &        &        &        &        &        &        & b_{kj} \\
    &        &        &        &        &        &        & \Vdots \\
    &        &        &        &        &        & b_{n1} & \Cdots & b_{nj} & \Cdots & b_{nn} \\[3mm]
    & a_{11} & \Cdots &        &        & a_{1n} \\
    & \Vdots &        &        &        & \Vdots &        &        & \Vdots \\
\color{blue}\scriptstyle L_i
    & a_{i1} & \Cdots & a_{ik} & \Cdots & a_{in} & \Cdots &        & c_{ij} \\
    & \Vdots &        &        &        & \Vdots \\
    & a_{n1} & \Cdots &        &        & a_{nn}  \\
\CodeAfter
\tikz \draw [gray,shorten > = 1mm, shorten < = 1mm] (9-4.north) to [bend left] (4-9.west) ;
\end{NiceArray}\]
```



L'ensemble de la figure est un environnement `{NiceArray}` et les trois paires de parenthèses ont été rajoutées avec `\SubMatrix` dans le `\CodeBefore` (le fait de les mettre dans le `\CodeBefore` et non le `\CodeAfter` permet de limiter à la matrice produit C les lignes de conduite pointillées qui y sont spécifiées).

18.12 Un tableau triangulaire

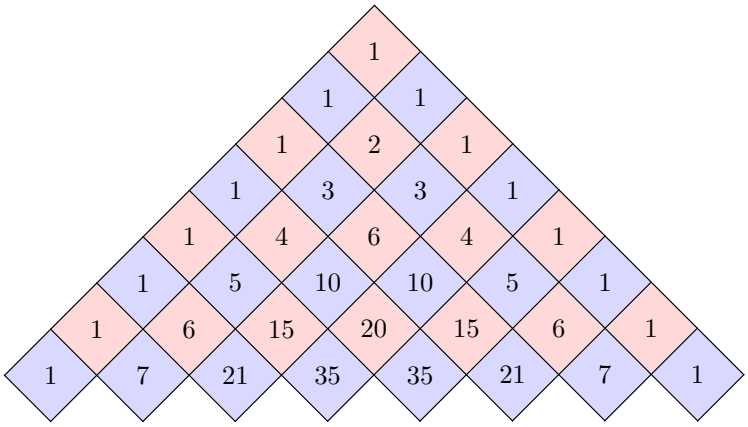
Dans l'exemple suivant, on utilise le style PGF-TikZ `nicematrix/cell-node` (cf. 15.1) pour faire tourner le contenu des cases (rotation que l'on compense ensuite en faisant tourner tout le tableau avec la commande `\adjustbox` de l'extension éponyme, qui doit donc être chargée préalablement).

```
\usepackage{adjustbox} % dans le préambule
```

```
\pgfset
  {
    nicematrix/cell-node/.append style =
      { text/rotate = 45, minimum size = 6 mm }
  }

\setlength{\tabcolsep}{0pt}

\adjustbox{rotate = -45, set depth = 6mm + 1.414 \arrayrulewidth}
  {\begin{NiceTabular} [ hvlines, corners=S, baseline = line-9 ] { cccccccc }
   \CodeBefore
     \chessboardcolors{red!15}{blue!15}
   \Body
     1 & 1 & 1  & 1  & 1  & 1 & 1 & 1 \\
     1 & 2 & 3  & 4  & 5  & 6 & 7 \\
     1 & 3 & 6  & 10 & 15 & 21 \\
     1 & 4 & 10 & 20 & 35 \\
     1 & 5 & 15 & 35 \\
     1 & 6 & 21 \\
     1 & 7 \\
     1
   \end{NiceTabular}}
```



Bien sûr, si on a de nombreux diagrammes à faire de la sorte, on recommande d'utiliser une extension dédiée comme atableau.

Index

Symbols

&-in-blocks, 13

A

ampersand-in-blocks, 13
arbres (clé « draw-trees-in-col »), 59
`\arraycolor` (commande du `\CodeBefore`), 26
`\arrayrulecolor`, 15
`\arrayrulewidth`, 15
auto-columns-width
    (clé de `{NiceMatrixBlock}`), 35, *84*
`\AutoNiceMatrix`, 63

B

baseline (clé pour un environnement), 5
`\BAutoNiceMatrix`, 63
`\bAutoNiceMatrix`, 63
blkarray (extension), 46
`\Block`, 6
Blocs dans les tableaux, 6–14
`{BNiceArray}`, 4
`{bNiceArray}`, 4
`{BNiceMatrix}`, 4
`{bNiceMatrix}`, 4
`\Body`, *voir* `\CodeBefore`
bold (clé de `\RowStyle`), 33
booktabs (extension), 14
borders (clé de `\Block`), 8
bottomrule (sous-clé de « notes »), 56
brace (nicematrix/brace est un style TikZ
        utilisé par `\Hbrace` et `\Vbrace`), 45
brace-shift (clé de `\Hbrace` et `\Vbrace`), 45

C

caption (clé de `{NiceTabular}`), 54
caption-above, 54
ccommand (clé de « custom-line »), 19
`\cdashedline`, 21
`\Cdots`, 39
`\cdottedline`, 22
cell-space-bottom-limit, 4, 33
cell-space-limits, 4, 33
cell-space-top-limit, 4, 33
`\cellcolor`
    commande du `\CodeBefore`, 26
    commande en tableau, 31
cellspace (extension), 5
`\chessboardcolors`
    (commande du `\CodeBefore`), 26, *88*
`\cline` (commande de LaTeX), 15
code (clé de `\SubMatrix`), 50
code-after, 47
code-before
    clé pour un environnement, 26
    sous-clé de « notes », 56
code-for-first-col, 38
code-for-first-row, 38, *62*, *80*
code-for-last-col, 38, *62*, *80*
code-for-last-row, 38, *62*
`\CodeAfter`, 47–53, *87*
`\CodeBefore`...`\Body`, 26, *87*, *88*
Coins arrondis
    pour un bloc, 7
Coins (les — vides), 17, *88*
Coins arrondis
    pour un tableau, 60
color
    clé de `\OverBrace` et `\UnderBrace`, 52
    clé de `\RowStyle`, 33
    clé de « custom-line », 20
    clé des lignes pointillées, 43
    pour les délimiteurs de matrices, 64
color-of-false (clé), 25
colortbl (extension), 25
cols (clé de `\rowcolors` du `\CodeBefore`), 28
`\columncolor`
    commande dans le préambule d'un environnement, 31
    commande du `\CodeBefore`, 26, *76*
columns-type (clé de `{NiceMatrix}`, etc.), 61
columns-width, 34
command (clé de « custom-line »), 19
corners (clé d'un environnement), 17, *29*
Couleur
    de fond pour les cases, 25
    des délimiteurs de matrices, 64
    des filets, 15
create-blocks-in-col, 14
create-cell-nodes (clé de `\CodeBefore`), 68, *87*
create-extra-nodes, 67
create-large-nodes, 67
create-medium-nodes, 67
`\crossbox` (définie dans un exemple), *71*
custom-line, 19–23

D

`\Ddots`, **39**, 72, *79*
default-line, 24
`\definecolorseries` (commande de xcolor), 29
delimiters
    —/color pour `\SubMatrix`, 49
    —/color pour un environnement, 64
    —/max-width, 84
délimiteurs dans les préambules, 46
detect-duplicates (sous-clé de « notes »), 56
`\diagbox`, 19
dotted (clé de « custom-line »), 22

draw (clé de \Block), 7, *86*
draw-first (clé de \Ddots et \Iddots), 73
draw-trees-in-col, 59

E

empty (clé de \Block), 8
empty (clé de \TikzEveryCell), 52
\EmptyColumn (commande du \CodeBefore), 31
\EmptyRow (commande du \CodeBefore), 31
end (clé pour les filets), 23
end-of-row (à utiliser avec light-syntax), 64
enumitem (extension requise pour utiliser \tabularnote), 54, *75*
enumitem-keys (sous-clé de « notes »), 56, *75*
enumitem-keys-para (sous-clé de « notes »), 56
exterior-arraycolsep, 73
extra-height (clé de \SubMatrix), 49
extra-left-margin, 68
extra-right-margin, 68

F

F (lettre dans les préambules), 25
\FalseRow, 25
Filets dans les tableaux, 14–25
fill
  clé de \Block, 7
  clé de \RowStyle, 33
first-col, 37
first-row, 37, *62*
fix-vertex (sous-clé de rules), 23
footnote (extension), 54
footnote (clé), 54
footnotehyper (extension), 54
footnotehyper (clé), 54

G

\g_nicematrix_code_after_tl, 71
\g_nicematrix_code_before_tl, 71

H

\Hbrace, 45
\hdashedline, 21
\Hdotsfor, 41
\hdottedline, 22
hhline (extension), *78*
\Hline, 16
hlines, *voir* Filets
  clé de \Block, 7
  clé de \SubMatrix, 50
  clé pour un environnement, 17
hlines-except-borders, 17
horizontal-label(s) (clé des lignes pointillées), 43
\Hspace, 40
hvlines, *voir* Filets
  clé de \Block, 7
  clé de \SubMatrix, 50
  clé pour un environnement, 17
hvlines-except-borders, 17, *77*

I

\Iddots, **39**, 72
Incompatibilités, 74
inter (clé des lignes pointillées), 43
iRow (compteur LaTeX), 63

J

jCol (compteur LaTeX), 63

L

label (clé de {NiceTabular}), 54
label-in-list (sous-clé de « notes »), 56
label-in-tabular (sous-clé de « notes »), 56
Largeur des colonnes, 34–37
last-col, 37, *62*, *80*
last-row, 37, *62*
\Ldots, 39
\left : utilisé par nicematrix
  pour des délimiteurs
  dans les préambules, 46
left-margin, 68
left-shorten (clé de \OverBrace et \UnderBrace), 52
left-xshift (clé de \SubMatrix), 49
Légende des tableaux, 54
letter (clé de « custom-line »), 19
light-syntax, 64
light-syntax-expanded, 64
Lignes en pointillés, *voir* Pointillés
\line (commande du \CodeAfter), 48
line-style (clé des lignes pointillées), 43
line-style (clé pour les lignes pointillées), *80*

M

mathdots (extension), 39
max-width (sous-clé de « delimiters »), 84
multiplicity (clé de « custom-line »), 20

N

name
  clé de \Block, 8
  clé de \SubMatrix, 70
  clé pour un environnement, 66
nb-rows (clé de \RowStyle), 33
{NiceArray}, 4
{NiceArrayWithDelims}, 65
{NiceMatrix}, 4
{NiceMatrixBlock}, 35, *83*
\NiceMatrixLastEnv, 65
\NiceMatrixOptions, 1
{NiceTabular}, 4
{NiceTabular*}, 4
\NiceTabularNotes, 58
{NiceTabularX}, 36
no-cell-nodes, 65
no-print (sous-clé de « notes »), 56
nocolor, 33
Nœuds PGF/Tikz, 65–70
non-empty (clé de \TikzEveryCell), 52

Notes dans les tableaux, 54–58, *75*
\NotEmpty, 18
notes (clé pour paramétrer les notes de tableau), 55, *75*
nullify (clé des lignes pointillées), 43
nullify-dots (obsolète), 40
number-of-dots (clé des lignes pointillées), 43

O

\OnlyMainNiceMatrix, 65
opacity
  clé de \RowStyle, 33
  clé de la commande \Block, 7
  clé des commandes comme \rowcolor, etc., 26
\OverBrace (commande du \CodeAfter et du \CodeBefore), 51
overlap (clé), 48

P

para (sous-clé de « notes »), 56
parallelize-diags, 72
\pAutoNiceMatrix, 63
pgf-node-code, 67, *88*
{pNiceArray}, 4
{pNiceMatrix}, 4
Pointillés (lignes de conduite en —), *79*
Pointillés (lignes en —), 39–46

R

radius (clé des lignes pointillées), 43
\rectanglecolor (commande du \CodeBefore), 26
renew-dots, 42
renew-matrix, 42
\resetcolorseries (commande de xcolor), 29
respect-arraystretch (clé de \Block), 8
respect-blocks (clé de \rowcolors du \CodeBefore), 28
restart (clé de \rowcolors du \CodeBefore), 28
\right : utilisé par nicematrix pour des délimiteurs dans les préambules, 46
right-margin, 68
right-shorten (clé de \OverBrace et \UnderBrace), 52
right-xshift (clé de \SubMatrix), 49
\rotate, *30*, *33*, 62
rounded-corners
  clé de \Block, 7
  clé de \RowStyle, 33
  clé de {NiceTabular}, 60
row-height (valeur pour columns-width), 35
\rowcolor
  commande du \CodeBefore, *77*
  commande du \CodeBefore, 26
  commande en tableau, 31, *87*
rowcolor (clé de \RowStyle), 33
\rowcolors (commande du \CodeBefore), 26
\rowlistcolors (commande du \CodeBefore), 26
\RowStyle, 33
rules (clé pour un environnement), 15, *77*
rules/color (clé de \Block), 7
rules/width (clé de \Block), 7

S

S (les colonnes S de siunitx), *31*, 61
sep-color (clé de « custom-line »), 20
short-caption, 54
shorten (clé des lignes pointillées), 43
shorten-end (clé des lignes pointillées), 43
shorten-start (clé des lignes pointillées), 43
\ShowCellNames (commande du \CodeAfter et du \CodeBefore), 61
siunitx (extension), 61
slim (clé de \SubMatrix), 50
small (clé pour un environnement), 62
{smallmatrix} (environnement de amsmath), 62
standard-cline, 15
start (clé pour les filets), 23
style (sous-clé de « notes »), 56, *75*
sub-matrix (clé de \CodeAfter, avec sous-clés), 47
\SubMatrix (commande du \CodeAfter et du \CodeBefore), **48**, 70, *85*, *87*

T

\tabularnote, 54, *75*
tabularnote (clé de {NiceTabular}), 55
{TabularNote}, 55
tabularx (extension), 36
Tagging Project, 74
tcolorbox (extension), *77*
TikZ (utilisation avec nicematrix), 65
\TikzEveryCell (commande du \CodeAfter et du \CodeBefore), **52**
tikz
  clé de \Block, 8, *76*
  clé de « borders » de \Block, 8, *79*
  clé de « custom-line », 23
total-width (clé de « custom-line »), 23
transparent (clé de \Block), 8, *76*
trees (plusieurs sous-clés), 59

U

\UnderBrace (commande du \CodeAfter et du \CodeBefore), 51

V

V (les colonnes V de varwidth), 35, 67
v-center (clé de \Block)), 12
varwidth (extension), 35, 67
\VAutoNiceMatrix, 63
\vAutoNiceMatrix, 63
\Vbrace, 45
\Vdots, 39
\Vdotsfor, 41

vlines, *voir* Filets
    clé de `\Block`, 7
    clé de `\SubMatrix`, 50
    clé pour un environnement, 17
vlines-in-sub-matrix, 85
`{VNiceArray}`, 4
`{vNiceArray}`, 4
`{VNiceMatrix}`, 4
`{vNiceMatrix}`, 4

W

width
    clé de `{NiceTabular}`, 36
    sous-clé de « rules », 15
width-of-false, 25

X

X (les colonnes X), 36
xdots (et ses sous-clés), 39
xshift (clé de `\SubMatrix`), 49

Y

yshift (clé de `\OverBrace` et `\UnderBrace`), 52

Autre documentation

Le document `nicematrix.pdf` (fourni avec l'extension nicematrix) contient une traduction anglaise de la documentation ici présente ainsi qu'un historique des versions.
Le document `nicematrix-code.pdf` (fourni également avec l'extension nicematrix) contient le code LaTeX commenté (à partir du fichier `nicematrix-code.dtx`).

Le développement de l'extension nicematrix se fait sur le dépôt GitHub suivant :
`https://github.com/fpantigny/nicematrix`

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

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