

# The package `nicematrix`\*

F. Pantigny  
fpantigny@wanadoo.fr

February 22, 2021

## Abstract

The LaTeX package `nicematrix` provides new environments similar to the classical environments `{tabular}`, `{array}` and `{matrix}` of `array` and `amsmath` but with extended features.

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

Product	dimensions (cm)			Price
	L	l	h	
small	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
special	12	12	0.5	70

The package `nicematrix` is entirely contained in the file `nicematrix.sty`. This file may be put in the current directory or in a `texmf` tree. However, the best is to install `nicematrix` with a TeX distribution as MiKTeX or TeXLive.

*Remark:* If you use LaTeX via Internet with, for example, Overleaf, you can upload the file `nicematrix.sty` in the repertory of your project in order to take full advantage of the latest version de `nicematrix`.<sup>1</sup>

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). However, the file `nicematrix.dtx` of the present documentation should be compiled with XeLaTeX.

This package requires and **loads** the packages `l3keys2e`, `array`, `amsmath`, `pgfcore` and the module `shapes` of PGF (`tikz`, which is a layer over PGF is *not* loaded). The final user only has to load the package with `\usepackage{nicematrix}`.

The idea of `nicematrix` is to create PGF nodes under the cells and the positions of the rules of the tabular created by `array` and to use these nodes to develop new features. As usual with PGF, the coordinates of these nodes are written in the `.aux` to be used on the next compilation and that's why `nicematrix` may need **several compilations**.

Most features of `nicematrix` may be used without explicit use of PGF or Tikz (which, in fact, is not loaded by default).

A command `\NiceMatrixOptions` is provided to fix the options (the scope of the options fixed by this command is the current TeX group: they are semi-global).

\*This document corresponds to the version 5.11 of `nicematrix`, at the date of 2021/02/22.

<sup>1</sup>The latest version of the file `nicematrix.sty` may be downloaded from the SVN server of TeXLive:  
<https://www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/nicematrix/nicematrix.sty>

# 1 The environments of this package

The package `nicematrix` defines the following new environments.

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

The environments `{NiceArray}`, `{NiceTabular}` and `{NiceTabular*}` are similar to the environments `{array}`, `{tabular}` and `{tabular*}` of the package `array` (which is loaded by `nicematrix`).

The environments `{pNiceArray}`, `{bNiceArray}`, etc. have no equivalent in `array`.

The environments `{NiceMatrix}`, `{pNiceMatrix}`, etc. are similar to the corresponding environments of `amsmath` (which is loaded by `nicematrix`): `{matrix}`, `{pmatrix}`, etc.

All the environments of the package `nicematrix` accept, between square brackets, an optional list of *key=value* pairs. **There must be no space before the opening bracket ([) of this list of options.**

## Important

Before the version 5.0, it was mandatory to use, for technical reasons, the letters `L`, `C` et `R` instead of `l`, `c` et `r` in the preambles of the environments of `nicematrix`. If we want to be able to go on using these letters, `nicematrix` must be loaded with the option `define-L-C-R`.

```
\usepackage[define-L-C-R]{nicematrix}
```

# 2 The vertical space between the rows

It's well known that some rows of the arrays created by default with LaTeX are, by default, too close to each other. Here is a classical example.

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

Inspired by the package `cellspace` which deals with that problem, the package `nicematrix` provides two keys `cell-space-top-limit` and `cell-space-bottom-limit` similar to the parameters `\cellspacetoplimit` and `\cellspacebottomlimit` of `cellspace`.

**New 5.9** There is also a key `cell-space-limits` to set both parameters at once.

The initial value of these parameters is 0 pt in order to have for the environments of `nicematrix` the same behaviour as those of `array` and `amsmath`. However, a value of 1 pt would probably be a good choice and we suggest to set them with `\NiceMatrixOptions`.<sup>2</sup>

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

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

---

<sup>2</sup>One should remark that these parameters apply also to the columns of type `S` of `siunitx` whereas the package `cellspace` is not able to act on such columns of type `S`.

### 3 The vertical position of the arrays

The package `nicematrix` provides a option `baseline` for the vertical position of the arrays. This option takes in as value an integer which is the number of the row on which the array will be aligned.

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

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

It's also possible to use the option `baseline` with one of the special values `t`, `c` or `b`. These letters may also be used absolutely like the option of the environments `{tabular}` and `{array}` of `array`. The initial value of `baseline` is `c`.

In the following example, we use the option `t` (equivalent to `baseline=t`) immediately after an `\item` of list. One should remark that the presence of a `\hline` at the beginning of the array doesn't prevent the alignment of the baseline with the baseline of the first row (with `{tabular}` or `{array}` of `array`, one must use `\firsthline`).

```
\begin{enumerate}
\item an 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. an item

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

However, it's also possible to use the tools of `booktabs`: `\toprule`, `\bottomrule`, `\midrule`, etc.

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

1. an item

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

It's also possible to use the key `baseline` to align a matrix on an horizontal rule (drawn by `\hline`). In this aim, one should give the value `line-i` where  $i$  is the number of the row following the horizontal rule.

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

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

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

## 4 The blocks

### 4.1 General case

In the environments of `nicematrix`, it's possible to use the command `\Block` in order to place an element in the center of a rectangle of merged cells of the array.

The command `\Block` must be used in the upper leftmost cell of the array with two arguments.

- The first argument is the size of the block with the syntax  $i$ - $j$  where  $i$  is the number of rows of the block and  $j$  its number of columns.

If this argument is empty, its default value is 1-1. If the number of rows is not specified, or equal to \*, the block extends until the last row (idem for the columns).

- The second argument is the content of the block. It's possible to use `\\` in that content to have a content on several lines. In `{NiceTabular}` the content of the block is composed in text mode whereas, in the other environments, it is composed in math mode.

Here is an example of utilisation of the command `\Block` in mathematical matrices.

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

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

One may wish to raise the size of the “A” placed in the block of the previous example. Since this element is composed in math mode, it's not possible to use directly a command like `\large`, `\Large` and `\LARGE`. That's why the command `\Block` provides an option between angle brackets to specify some TeX code which will be inserted before the beginning of the math mode.<sup>3</sup>

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

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

It's possible to set the horizontal position of the block with one of the keys `l`, `c` and `r`.

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

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

**One must remark that, by default, the commands `\Blocks` don't create space.** There is exception only for the blocks mono-row and the blocks mono-column as explained just below.

In the following example, we have had to enlarge by hand the columns 2 and 3 (with the construction `wc{...}` of `array`).

<sup>3</sup>This argument between angular brackets may also be used to insert a command of font such as `\bfseries` when the command `\\` is used in the content of the block.

```

\begin{NiceTabular}{cwc{2cm}wc{3cm}c}
rose      & tulipe & marguerite & dahlia \\
violette  &        &            &        \\
& \Block[draw=red,fill=red!15]{2-2}{\LARGE De très jolies fleurs} & & souci \\
pervenche & & & lys \\
arum      & iris & jacinthe & muguet
\end{NiceTabular}

```

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

As we can see on this example, it's possible to fill the block by using the key `fill` and to draw the frame with the key `draw`<sup>4</sup>. It's also possible to change the width (thickness) of that rules with the key `line-width`.

## 4.2 The mono-column blocks

The mono-column blocks have a special behaviour.

- The natural width of the contents of these blocks is taken into account for the width of the current column.
- The specification of the horizontal position provided by the type of column (`c`, `r` or `l`) is taken into account for the blocks.
- The specifications of font specified for the column by a construction `>{...}` in the preamble of the array are taken into account for the mono-column blocks of that column (this behaviour is probably expected).

<code>\begin{NiceTabular}{@{}&gt;{\bfseries}lr@{}} \hline</code>		
<code>\Block{2-1}{John}</code>	<code>&amp; 12 \\</code>	<b>John</b> 12
	<code>&amp; 13 \\ \hline</code>	13
Steph	<code>&amp; 8 \\ \hline</code>	<b>Steph</b> 8
<code>\Block{3-1}{Sarah}</code>	<code>&amp; 18 \\</code>	18
	<code>&amp; 17 \\</code>	<b>Sarah</b> 17
	<code>&amp; 15 \\ \hline</code>	15
Ashley	<code>&amp; 20 \\ \hline</code>	<b>Ashley</b> 20
Henry	<code>&amp; 14 \\ \hline</code>	<b>Henry</b> 14
<code>\Block{2-1}{Madison}</code>	<code>&amp; 15 \\</code>	15
	<code>&amp; 19 \\ \hline</code>	<b>Madison</b> 19
<code>\end{NiceTabular}</code>		

## 4.3 The mono-row blocks

For the mono-row blocks, the natural height and depth are taken into account for the height and depth of the current row (as does a standard `\multicolumn` of LaTeX).

## 4.4 The mono-cell blocks

A mono-cell block inherits all the properties of the mono-row blocks and mono-column blocks.

At first sight, one may think that there is no point using a mono-cell block. However, there are some good reasons to use such a block.

- It's possible to use the command `\\` in a (mono-cell) block.

<sup>4</sup>If the key `draw` is used without value, the default color the rules of the tabulars is used

- It's possible to use the option of horizontal alignment of the block in derogation of the type of column given in the preamble of the array.
- It's possible to draw a frame around the cell with the key **draw** of the command `\Block`.<sup>5</sup>

```
\begin{NiceTabular}{cc}
\toprule
Writer & \Block[1]{year of birth} \\
\midrule
Hugo & 1802 \\
Balzac & 1799 \\
\bottomrule
\end{NiceTabular}
```

Writer	year of birth
Hugo	1802
Balzac	1799

We recall that if the first mandatory argument of `\Block` is left blank, the block is mono-cell.<sup>6</sup>

## 4.5 A small remark

One should remark that the horizontal centering of the contents of the blocks is correct even when an instruction such as `!\qquad` has been used in the preamble of the array in order to increase the space between two columns (this is not the case with `\multicolumn`). In the following example, the header “First group” is correctly centered.

```
\begin{NiceTabular}{@{c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
& \Block{1-3}{First group} & & \Block{1-3}{Second group} \\
Rank & 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}
```

	First group			Second group		
Rank	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

## 5 The rules

The usual techniques for the rules may be used in the environments of `nicematrix` (excepted `\vline`). However, there is some small differences with the classical environments.

<sup>5</sup>If one wishes to color the background of a unique celle, there is no point using the command `\Block`: it's possible to use the command `\cellcolor` (when the key `colortbl`-like is used).

<sup>6</sup>One may consider that the default value of the first mandatory argument of `\Block` is 1-1.

## 5.1 Some differences with the classical environments

### 5.1.1 The vertical rules

In the environments of `nicematrix`, the vertical rules specified by `|` in the preambles of the environments are never broken, even by an incomplete row or by a double horizontal rule specified by `\hline\hline` (there is no need to use `hhline`).

```
\begin{NiceTabular}{|c|c|} \hline
First & Second \\ \hline\hline
Peter & \\ \hline
Mary & George \\ \hline
\end{NiceTabular}
```

First	Second
Peter	
Mary	George

However, the vertical rules are not drawn in the blocks.

If you use `booktabs` (which provides `\toprule`, `\midrule`, `\bottomrule`, etc.) and if you really want to add vertical rules (which is not in the spirit of `booktabs`), you should notice that the vertical rules drawn by `nicematrix` are compatible with `booktabs`.

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

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

However, it's still possible to define a specifier (named, for instance, `I`) to draw vertical rules with the standard behaviour of `array`.

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

However, in this case, it is probably more clever to add a command `\OnlyMainNiceMatrix` (cf. p. 34):

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

### 5.1.2 The command `\cline`

The horizontal and vertical rules drawn by `\hline` and the specifier “`|`” make the array larger or wider by a quantity equal to the width of the rule (with `array` and also with `nicematrix`).

For historical reasons, this is not the case with the command `\cline`, as shown by the following example.

```
\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	<u>B</u>	C	D

In the environments of `nicematrix`, this situation is corrected (it's still possible to go to the standard behaviour of `\cline` with the key `standard-cline`).

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

A	B	C	D
A	<u>B</u>	C	D

## 5.2 The thickness and the color of the rules

The environments of `nicematrix` provide a key `rules/width` to set the width (in fact the thickness) of the rules in the current environment. In fact, this key merely sets the value of the length `\arrayrulewidth`.

It's well known that `colortbl` provides the command `\arrayrulecolor` in order to specify the color of the rules.

With `nicematrix`, it's possible to specify the color of the rules even when `colortbl` is not loaded. For sake of compatibility, the command is also named `\arrayrulecolor`. The environments of `nicematrix` also provide a key `rules/color` to fix the color of the rules in the current environment.

```
\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

If one wishes to define new specifiers for columns in order to draw vertical rules (for example with a specific color or thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 34.

## 5.3 The keys `hlines` and `vlines`

The key `hlines` draws all the horizontal rules and the key `vlines` draws all the vertical rules excepted in the blocks (and the virtual blocks determined by dotted lines). In fact, in the environments with delimiters (as `{pNiceMatrix}` or `{bNiceArray}`) the exteriors rules are not drawn (as expected).

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

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

## 5.4 The key `hvlines`

The key `hvlines` draws all the vertical and horizontal rules (excepted in the blocks and the virtual blocks determined by dotted lines and excepted the rules corresponding of the frame of the blocks using the key `draw` which are drawn with their own characteristics).

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

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



## 5.5 The key `hvlines-except-corners`

The key `hvlines-except-corners` draws all the horizontal and vertical rules, excepted in the blocks (and the virtual blocks determined by dotted lines) and excepted in the empty corners.

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

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

As we can see, an “empty corner” is composed by the reunion of all the empty rectangles starting from the cell actually in the corner of the array.

It’s possible to give as value to the key `hvlines-except-corners` a list of the corners to take into consideration. The corners are designed by NW, SW, NE and SE (*north west*, *south west*, *north east* and *south east*).

```
\begin{NiceTabular}{*{6}{c}}%
[hvlines-except-corners=NE,cell-space-top-limit=3pt]
1\\
1&1\\
1&2&1\\
1&3&3&1\\
1&4&6&4&1\\
1&5&10&10&5&1
\end{NiceTabular}
```

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
1	5	10	10	5	1

## 5.6 The command `\diagbox`

The command `\diagbox` (inspired by the package `diagbox`), allows, when it is used in a cell, to slash that cell diagonally downwards.<sup>7</sup>

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

$\begin{smallmatrix} y \\ \diagdown \\ x \end{smallmatrix}$	<i>e</i>	<i>a</i>	<i>b</i>	<i>c</i>
<i>e</i>	<i>e</i>	<i>a</i>	<i>b</i>	<i>c</i>
<i>a</i>	<i>a</i>	<i>e</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>c</i>	<i>e</i>	<i>a</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>e</i>

It’s possible to use the command `\diagbox` in a `\Block`.

<sup>7</sup>The author of this document considers that type of construction as graphically poor.

## 5.7 Dotted rules

In the environments of the package `nicematrix`, it's possible to use the command `\hdottedline` (provided by `nicematrix`) which is a counterpart of the classical commands `\hline` and `\hdashline` (the latter is a command of `arydshln`).

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

$$\left(\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \hdottedline 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{array}\right)$$

In the environments with an explicit preamble (like `{NiceTabular}`, `{NiceArray}`, etc.), it's possible to draw a vertical dotted line with the specifier “:”.

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

$$\left(\begin{array}{ccccc} 1 & 2 & 3 & 4 & : 5 \\ 6 & 7 & 8 & 9 & : 10 \\ 11 & 12 & 13 & 14 & : 15 \end{array}\right)$$

It's possible to change in `nicematrix` the letter used to specify a vertical dotted line with the option `letter-for-dotted-lines` available in `\NiceMatrixOptions`. Thus released, the letter “:” can be used otherwise (for example by the package `arydshln`).

*Remark :* In the package `array` (on which the package `nicematrix` relies), horizontal and vertical rules make the array larger or wider by a quantity equal to the width of the rule<sup>8</sup>. In `nicematrix`, the dotted lines drawn by `\hdottedline` and “:” do likewise.

## 6 The color of the rows and columns

### 6.1 Use of `colortbl`

We recall that the package `colortbl` can be loaded directly with `\usepackage{colortbl}` or by loading `xcolor` with the key `table`: `\usepackage[table]{xcolor}`.

Since the package `nicematrix` is based on `array`, it's possible to use `colortbl` with `nicematrix`.

However, there is two drawbacks:

- The package `colortbl` patches `array`, leading to some incompatibilities (for example with the command `\hdotsfor`).
- The package `colortbl` constructs the array row by row, alternating colored rectangles, rules and contents of the cells. The resulting PDF is difficult to interpret by some PDF viewers and may lead to artefacts on the screen.
  - Some rules seem to disappear. This is because many PDF viewers give priority to graphical element drawn posteriorly (which is in the spirit of the “painting model” of PostScript and PDF). Concerning this problem, MuPDF (which is used, for instance, by SumatraPDF) gives better results than Adobe Reader).
  - A thin white line may appear between two cells of the same color. This phenomenon occurs when each cell is colored with its own instruction `fill` (the PostScript operator `fill` noted `f` in PDF). This is the case with `colortbl`: each cell is colored on its own, even when `\columncolor` or `\rowcolor` is used.

As for this phenomenon, Adobe Reader gives better results than MuPDF.

The package `nicematrix` provides tools to avoid those problems.

---

<sup>8</sup>In fact, this is true only for `\hline` and “|” but not for `\cline`: cf p. 7

## 6.2 The tools of nicematrix in the code-before

The package `nicematrix` provides some tools (independent of `colortbl`) to draw the colored panels first, and, then, the content of the cells and the rules. This strategy is more conform to the “painting model” of the formats PostScript and PDF and is more suitable for the PDF viewers. However, it requires several compilations.

The extension `nicematrix` provides a key `code-before` for some code that will be executed before the drawing of the tabular. In this `code-before`, new commands are available: `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors` and `\chessboardcolors`.<sup>9</sup>

All these commands accept an optional argument (between square brackets and in first position) which is the color model for the specification of the colors.

- The command `\cellcolor` takes its name from the command `\cellcolor` of `colortbl`.

This command takes in as mandatory arguments a color and a list of cells, each of which with the format  $i$ - $j$  where  $i$  is the number of the row and  $j$  the number of the column of the cell.

```
\begin{NiceTabular}{|c|c|c|}[code-before =
\cellcolor{red!15}{3-1,2-2,1-3}]
\hline
a & b & c \\ \hline
e & f & g \\ \hline
h & i & j \\ \hline
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

- The command `\rectanglecolor` takes three mandatory arguments. The first is the color. The second is the upper-left cell of the rectangle and the third is the lower-right cell of the rectangle.

```
\begin{NiceTabular}{|c|c|c|}[code-before =
\rectanglecolor{blue!15}{2-2}{3-3}]
\hline
a & b & c \\ \hline
e & f & g \\ \hline
h & i & j \\ \hline
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

- The command `\rowcolor` takes its name from the command `\rowcolor` of `colortbl`. Its first mandatory argument is the color and the second is a comma-separated list of rows or interval of rows with the form  $a$ - $b$  (an interval of the form  $a$ - represent all the rows from the row  $a$  until the end).

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

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

<sup>9</sup>Remark that, in the `code-before`, some PGF/Tikz nodes corresponding to the position to the potential rules are available: cf. p. 32.

- The command `\columncolor` takes its name from the command `\columncolor` of `colortbl`. Its syntax is similar to the syntax of `\rowcolor`.
- The command `\rowcolors` (with a *s*) takes its name from the command `\rowcolors` of `xcolor`<sup>10</sup>. The *s* emphasizes the fact that there is *two* colors. This command colors alternately the rows of the tabular with the tow colors (provided in second and third argument), beginning with the row whose number is given in first (mandatory) argument.

In fact, the first (mandatory) argument is, more generally, a comma separated list of intervals describing the rows involved in the action of `\rowcolors` (an interval of the form *i-j* describes in fact the interval of all the rows of the tabular, beginning with the row *i*).

The last argument of `\rowcolors` is an optional list of pairs key-value (the optional argument in the first position corresponds to the colorimetric space). The available keys are `cols`, `restart` and `respect-blocks`.

- The key `cols` describes a set of columns. The command `\rowcolors` will color only the cells of these columns. The value is a comma-separated list of intervals of the form *i-j*.
- With the key `restart`, each interval of rows (specified by the first mandatory argument) begins with the same color.<sup>11</sup>
- With the key `respect-blocks` the “rows” alternately colored may extend over several rows if they have to incorporate blocks (created with the command `\Block`: cf. p. 4).

```
\begin{NiceTabular}{clr}
  [hvlines,code-before = {\rowcolors{2}{blue!10}{}[cols=2-3,restart]}]
\Block{1-}{Results} \l
John & 12 \l
Stephen & 8 \l
Sarah & 18 \l
Ashley & 20 \l
Henry & 14 \l
Madison & 15
\end{NiceTabular}
```

Results		
A	John	12
	Stephen	8
B	Sarah	18
	Ashley	20
	Henry	14
	Madison	15

```
\begin{NiceTabular}{lrr}[hvlines,code-before =
\rowcolors{1}{blue!10}{}[respect-blocks]]
\Block{2-1}{John} & 12 \l
& 13 \l
Steph & 8 \l
\Block{3-1}{Sarah} & 18 \l
& 17 \l
& 15 \l
Ashley & 20 \l
Henry & 14 \l
\Block{2-1}{Madison} & 15 \l
& 19
\end{NiceTabular}
```

John	12
	13
Steph	8
Sarah	18
	17
	15
Ashley	20
Henry	14
Madison	15
	19

- The command `\chessboardcolors` takes in as mandatory arguments two colors and it colors the cells of the tabular in quincunx with these colors.

<sup>10</sup>The command `\rowcolors` of `xcolor` is available when `xcolor` is loaded with the option `table`. That option also loads the package `colortbl`.

<sup>11</sup>Otherwise, the color of a given row relies only upon the parity of its number.

```


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


```

We have used the key `r` which aligns all the columns rightwards (cf. p. 28).

One should remark that these commands are compatible with the commands of `booktabs` (`\toprule`, `\midrule`, `\bottomrule`, etc). However, `booktabs` is not loaded by `nicematrix`.

```

\begin{NiceTabular}[c]{lSSSS}%
[code-before = \rowcolor{red!15}{1-2} \rowcolors{3}{blue!15}{}]
\toprule
\Block{2-1}{Product} &
\Block{1-3}{dimensions (cm)} & & &
\Block{2-1}{\rotate Price} \\
\cmidrule{2-4}
& L & l & h & \\
\midrule
small & 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 \\
special & 12 & 12 & 0.5 & 70 \\
\bottomrule
\end{NiceTabular}

```

Product	dimensions (cm)			Price
	L	l	h	
small	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
special	12	12	0.5	70

We have used the type of column `S` of `siunitx`.

### 6.3 Color tools with the syntax of `colortbl`

It's possible to access the preceding tools with a syntax close to the syntax of `colortbl`. For that, one must use the key `colortbl-like` in the current environment.<sup>12</sup>

There are three commands available (they are inspired by `colortbl` but are *independent* of `colortbl`):

- `\cellcolor` which colorizes a cell;
- `\rowcolor` which must be used in a cell and which colorizes the end of the row;
- `\columncolor` which must be used in the preamble of the environment with the same syntax as the corresponding command of `colortbl` (however, unlike the command `\columncolor` of `colortbl`, this command `\columncolor` can appear within another command, itself used in the preamble of the array).

```

\NewDocumentCommand { \Blue } { } { \columncolor{blue!15} }
\begin{NiceTabular}[colortbl-like]{>{\Blue}c>{\Blue}cc}
\toprule
\rowcolor{red!15}
Last name & First name & Birth day \\
\midrule
Achard & Jacques & 5 juin 1962 \\
Lefebvre & Mathilde & 23 mai 1988 \\
Vanesse & Stephany & 30 octobre 1994 \\
Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}

```

<sup>12</sup>As for now, this key is *not* available in `\NiceMatrixOptions`.

Last name	First name	Birth day
Achard	Jacques	5 juin 1962
Lefebvre	Mathilde	23 mai 1988
Vanesse	Stephany	30 octobre 1994
Dupont	Chantal	15 janvier 1998

## 7 The width of the columns

In the environments with an explicit preamble (like `{NiceTabular}`, `{NiceArray}`, etc.), it's possible to fix the width of a given column with the standard letters `w` and `W` of the package `array`.

```
\begin{NiceTabular}{Wc{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

In the environments of `nicematrix`, it's also possible to fix the *minimal* width of all the columns of an array directly with the key `columns-width`.

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

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

Note that the space inserted between two columns (equal to `2 \tabcolsep` in `{NiceTabular}` and to `2 \arraycolsep` in the other environments) is not suppressed (of course, it's possible to suppress this space by setting `\tabcolsep` or `\arraycolsep` equal to 0 pt before the environment).

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.<sup>13</sup>

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

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

Without surprise, it's possible to fix the minimal width of the columns of all the matrices of a current scope with the command `\NiceMatrixOptions`.

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

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

But it's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `{NiceMatrixBlock}` with the option `auto-columns-width`<sup>14</sup>. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 4).

<sup>13</sup>The result is achieved with only one compilation (but PGF/Tikz will have written informations in the `.aux` file and a message requiring a second compilation will appear).

<sup>14</sup>At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

```

\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}

```

Several compilations may be necessary to achieve the job.

## 8 The exterior rows and columns

The options `first-row`, `last-row`, `first-col` and `last-col` allow the composition of exterior rows and columns in the environments of `nicematrix`.

A potential “first row” (exterior) has the number 0 (and not 1). Idem for the potential “first column”.

```

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

```

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

The dotted lines have been drawn with the tools presented p. 16.

We have several remarks to do.

- For the environments with an explicit preamble (i.e. `{NiceTabular}`, `{NiceArray}` and its variants), no letter must be given in that preamble for the potential first column and the potential last column: they will automatically (and necessarily) be of type `r` for the first column and `l` for the last one.<sup>15</sup>
- One may wonder how `nicematrix` determines the number of rows and columns which are needed for the composition of the “last row” and “last column”.
  - For the environments with explicit preamble, like `{NiceTabular}` and `{pNiceArray}`, the number of columns can obviously be computed from the preamble.
  - When the option `light-syntax` (cf. p. 30) is used, `nicematrix` has, in any case, to load the whole body of the environment (and that’s why it’s not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows (but not the number of columns).

<sup>15</sup>The users wishing exteriors columns with another type of alignment should consider the command `\SubMatrix` available in the `\CodeAfter` (cf. p. 22).

- In the other cases, `nicematrix` compute the number of rows and columns during the first compilation and write the result in the `aux` file for the next run.

However, it's possible to provide the number of the last row and the number of the last column as values of the options `last-row` and `last-col`, tending to an acceleration of the whole compilation of the document. That's what we will do throughout the rest of the document.

It's possible to control the appearance of these rows and columns with options `code-for-first-row`, `code-for-last-row`, `code-for-first-col` and `code-for-last-col`. These options specify tokens that will be inserted before each cell of the corresponding row or column.

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

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

#### Remarks

- As shown in the previous example, the horizontal and vertical rules doesn't extend in the exterior rows and columns.  
However, if one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 34.
- A specification of color present in `code-for-first-row` also applies to a dotted line draw in this exterior "first row" (excepted if a value has been given to `xdots/color`). Idem for the other exterior rows and columns.
- Logically, the potential option `columns-width` (described p. 14) doesn't apply to the "first column" and "last column".
- For technical reasons, it's not possible to use the option of the command `\` after the "first row" or before the "last row". The placement of the delimiters would be wrong. If you are looking for a workaround, consider the command `\SubMatrix` in the `\CodeAfter` described p. 22.

## 9 The continuous dotted lines

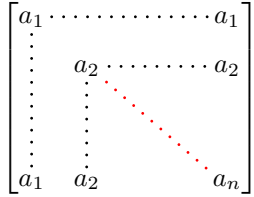
Inside the environments of the package `nicematrix`, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots`. These commands are intended to be used in place of `\dots`, `\cdots`, `\vdots`, `\ddots` and `\iddots`.<sup>16</sup>

<sup>16</sup>The command `\iddots`, defined in `nicematrix`, is a variant of `\ddots` with dots going forward. If `mathdots` is loaded, the version of `mathdots` is used. It corresponds to the command `\adots` of `unicode-math`.



Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells<sup>17</sup> on both sides of the current cell. Of course, for `\Ldots` and `\Cdots`, it's an horizontal line; for `\Vdots`, it's a vertical line and for `\Ddots` and `\Iddots` diagonal ones. It's possible to change the color of these lines with the option `color`.<sup>18</sup>

```
\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}
```



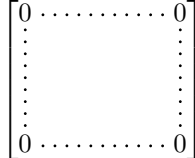
In order to represent the null matrix, one can use the following codage:

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



However, one may want a larger matrix. Usually, in such a case, the users of LaTeX add a new row and a new column. It's possible to use the same method with `nicematrix`:

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



In the first column of this exemple, there are two instructions `\Vdots` but, of course, only one dotted line is drawn.

In fact, in this example, it would be possible to draw the same matrix more easily with the following code:

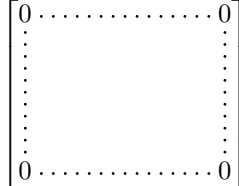
```
\begin{bNiceMatrix}
0      & \Cdots &      & 0      & \\
\Vdots &        &      & \Vdots & \\
          &        &      & \Vdots & \\
0      &        & \Cdots & 0      & \\
\end{bNiceMatrix}
```



There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.<sup>19</sup>

However, a command `\hspace*` might interfere with the construction of the dotted lines. That's why the package `nicematrix` provides a command `\Hspace` which is a variant of `\hspace` transparent for the dotted lines of `nicematrix`.

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



<sup>17</sup>The precise definition of a “non-empty cell” is given below (cf. p. 35).

<sup>18</sup>It's also possible to change the color of all theses dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 20.

<sup>19</sup>In `nicematrix`, one should use `\hspace*` and not `\hspace` for such an usage because `nicematrix` loads `array`. One may also remark that it's possible to fix the width of a column by using the environment `\NiceArray` (or one of its variants) with a column of type `w` or `W`: see p. 14

## 9.1 The option nullify-dots

Consider the following matrix composed classically with the environment `{pmatrix}` of `amsmath`.

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

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

If we add `\ldots` instructions in the second row, the geometry of the matrix is modified.

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

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

By default, with `nicematrix`, if we replace `{pmatrix}` by `{pNiceMatrix}` and `\ldots` by `\Ldots`, the geometry of the matrix is not changed.

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

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

However, one may prefer the geometry of the first matrix  $A$  and would like to have such a geometry with a dotted line in the second row. It's possible by using the option `nullify-dots` (and only one instruction `\Ldots` is necessary).

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

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

The option `nullify-dots` smashes the instructions `\Ldots` (and the variants) horizontally but also vertically.

## 9.2 The commands `\Hdotsfor` and `\Vdotsfor`

Some people commonly use the command `\hdotsfor` of `amsmath` in order to draw horizontal dotted lines in a matrix. In the environments of `nicematrix`, one should use instead `\Hdotsfor` in order to draw dotted lines similar to the other dotted lines drawn by the package `nicematrix`.

As with the other commands of `nicematrix` (like `\Cdots`, `\Ldots`, `\Vdots`, etc.), the dotted line drawn with `\Hdotsfor` extends until the contents of the cells on both sides.

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

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

However, if these cells are empty, the dotted line extends only in the cells specified by the argument of `\Hdotsfor` (by design).

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

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

Remark: Unlike the command `\hdotsfor` of `amsmath`, the command `\Hdotsfor` may be used even when the package `colortbl`<sup>20</sup> is loaded (but you might have problem if you use `\rowcolor` on the same row as `\Hdotsfor`).

<sup>20</sup>We recall that when `xcolor` is loaded with the option `table`, the package `colortbl` is loaded.

The package `nicematrix` also provides a command `\Vdotsfor` similar to `\Hdotsfor` but for the vertical dotted lines. The following example uses both `\Hdotsfor` and `\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
& \Vdotsfor{1} & \Vdots & \Ddots & \Vdots \\
C[a_n,a_1] & \Cdots & C[a_n,a_n]
& & C[a_n,a_1^{(p)}] & \Cdots & C[a_n,a_n^{(p)}] \\
\rule{0pt}{15mm} & \Vdotsfor{1} & & \Ddots & & \Vdotsfor{1} \\
C[a_1^{(p)},a_1] & \Cdots & C[a_1^{(p)},a_n]
& & C[a_1^{(p)},a_1^{(p)}] & \Cdots & C[a_1^{(p)},a_n^{(p)}] \\
& \Vdots & \Ddots & \Vdots
& \Vdotsfor{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}
```

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

### 9.3 How to generate the continuous dotted lines transparently

Imagine you have a document with a great number of mathematical matrices with ellipsis. You may wish to use the dotted lines of `nicematrix` without having to modify the code of each matrix. It's possible with the keys. `renew-dots` and `renew-matrix`.<sup>21</sup>

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>16</sup> and `\hdotsfor` are redefined within the environments provided by `nicematrix` and behave like `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor`; the command `\dots` (“automatic dots” of `amsmath`) is also redefined to behave like `\Ldots`.

- The option `renew-matrix`

With this option, the environment `{matrix}` is redefined and behave like `{NiceMatrix}`, and so on for the five variants.

Therefore, with the keys `renew-dots` and `renew-matrix`, a classical code gives directly the output of

<sup>21</sup>The options `renew-dots`, `renew-matrix` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage`. There is also a key `transparent` which is an alias for the conjunction of `renew-dots` and `renew-matrix` but it must be considered as obsolete.

nicematrix.

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

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

## 9.4 The labels of the dotted lines

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and the command `\line` in the `\CodeAfter` which is described p. 22) accept two optional arguments specified by the tokens `_` and `^` for labels positionned below and above the line. The arguments are composed in math mode with `\scriptstyle`.

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

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

## 9.5 Customisation of the dotted lines

The dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and by the command `\line` in the `\CodeAfter` which is described p. 22) may be customized by three options (specified between square brackets after the command):

- `color`;
- `shorten`;
- `line-style`.

These options may also be fixed with `\NiceMatrixOptions`, as options of `\CodeAfter` or at the level of a given environment but, in those cases, they must be prefixed by `xdots`, and, thus have for names:

- `xdots/color`;
- `xdots/shorten`;
- `xdots/line-style`.

For the clarity of the explanations, we will use those names.

### The option `xdots/color`

The option `xdots/color` fixes the color of the dotted line. However, one should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. p. 15.

### The option `xdots/shorten`

The option `xdots/shorten` fixes the margin of both extremities of the line. The name is derived from the options “`shorten >`” and “`shorten <`” of Tikz but one should notice that `nicematrix` only provides `xdots/shorten`. The initial value of this parameter is 0.3 em (it is recommended to use a unit of length dependent of the current font).

### The option `xdots/line-style`

It should be pointed that, by default, the lines drawn by Tikz with the parameter `dotted` are composed of square dots (and not rounded ones).<sup>22</sup>

<sup>22</sup>The first reason of this behaviour is that the PDF format includes a description for dashed lines. The lines specified with this descriptor are displayed very efficiently by the PDF readers. It's easy, starting from these dashed lines, to create a line composed by square dots whereas a line of rounded dots needs a specification of each dot in the PDF file.

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

In order to provide lines with rounded dots in the style of those provided by `\ldots` (at least with the *Computer Modern* fonts), the package `nicematrix` embeds its own system to draw a dotted line (and this system uses PGF and not Tikz). This style is called `standard` and that's the initial value of the parameter `xdots/line-style`.

However (when Tikz is loaded) it's possible to use for `xdots/line-style` any style provided by Tikz, that is to say any sequence of options provided by Tikz for the Tikz pathes (with the exception of “color”, “shorten >” and “shorten <”).

Here is for example a tridiagonal matrix with the style `loosely dotted`:

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

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

## 9.6 The dotted lines and the rules

The dotted lines determine virtual blocks which have the same behaviour regarding the rules (the rules specified by the specifier `|` in the preamble and by the keys `hlines`, `vlines`, `hvlines` and `hvlines-except-corners` are not drawn within the blocks).

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

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

## 10 The `\CodeAfter`

The option `code-after` may be used to give some code that will be executed after the construction of the matrix.<sup>23</sup>

For the legibility of the code, an alternative syntax is provided: it's possible to give the instructions of the `\code-after` at the end of the environment, after the keyword `\CodeAfter`. Although `\CodeAfter` is a keyword, it takes in an optional argument (between square brackets). The keys accepted form a subset of the keys of the command `\WithArrowsOptions`.

The experienced users may use, for instance, the PGF/Tikz nodes created by `nicematrix` in the `\CodeAfter`. These nodes are described further beginning on p. 30.

Moreover, two special commands are available in the `\CodeAfter`: `line` and `\SubMatrix`.

<sup>23</sup>There is also a key `code-before` described p. 11.

## 10.1 The command `\line` in the `\CodeAfter`

The command `\line` draws directly dotted lines between nodes. It takes in two arguments for the two cells to link, both of the form  $i$ - $j$  where  $i$  is the number of the row and  $j$  is the number of the column. It may be used, for example, to draw a dotted line between two adjacent cells.

```
\NiceMatrixOptions{xdots/shorten = 0.6 em}
\begin{pNiceMatrix}
I      & 0      & & \Cdots & 0      & \\
0      & I      & & \Ddots & \Vdots & \\
\Vdots & \Ddots & & I      & 0      & \\
0      & \Cdots & & 0      & I      & \\
\CodeAfter \line{2-2}{3-3}
\end{pNiceMatrix}
```

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

The options available for the customisation of the dotted lines created by `\Cdots`, `\Vdots`, etc. are also available for this command (cf. p. 20).

## 10.2 The command `\SubMatrix` in the `\CodeAfter`

**New 5.10** The command `\SubMatrix` provides a way to put delimiters on a portion of the array considered as a submatrix. The command `\SubMatrix` takes in five arguments:

- the first argument is the left delimiter, which may be any extensible delimiter provided by LaTeX : `(`, `[`, `\{`, `\langle`, `\lggroup`, `\lfloor`, etc. but also the null delimiter `.`;
- the second argument is the upper-left corner of the submatrix with the syntax  $i$ - $j$  where  $i$  the number of row and  $j$  the number of column;
- the third argument is the lower-right corner with the same syntax;
- the fourth argument is the right delimiter;
- the last argument, which is optional, is a list of key-value pairs.

One should remark that the command `\SubMatrix` draws the delimiters after the construction of the array: no space is inserted by the command `\SubMatrix` itself. That's why, in the following example, we have used the key `margin` and you have added by hand some space between the third and fourth column with `@{\hspace{1.5em}}` in the preamble of the array.

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

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

The options of the command `\SubMatrix` are as follows:

- `left-xshift` and `right-shift` shift horizontally the delimiters (there exists also the key `xshift` which fixes both parameters);
- `extra-height` adds a quantity to the total height of the delimiters (height `\ht` + depth `\dp`);
- `delimiters/color` fixes the color of the delimiters (also available in `\NiceMatrixOptions`, in the environments with delimiters and as option of the command `\CodeAfter`);
- `slim` is a boolean key: when that key is in force, the horizontal position of the delimiters is computed by using only the contents of the cells of the submatrix whereas, in the general case, the position is computed by taking into account the cells of the whole columns implied in the submatrix (see example below).

These keys are also available in `\NiceMatrixOptions`, at the level of the environments of `nicematrix` or as option of the command `\CodeAfter` with the prefix `sub-matrix` which means that their names are therefore `sub-matrix/left-xshift`, `sub-matrix/right-xshift`, `sub-matrix/xshift`, etc.

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

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

Here is the same example with the key `slim` used for one of the submatrices.

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

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

There is also a key `name` which gives a name to the submatrix created by `\SubMatrix`. That name is used to created PGF/Tikz nodes: cf p. 33.

## 11 The notes in the tabulars

### 11.1 The footnotes

The package `nicematrix` allows, by using `footnote` or `footnotehyper`, the extraction of the notes inserted by `\footnote` in the environments of `nicematrix` and their composition in the footpage with the other notes of the document.

If `nicematrix` is loaded with the option `footnote` (with `\usepackage[footnote]{nicematrix}` or with `\PassOptionsToPackage`), the package `footnote` is loaded (if it is not yet loaded) and it is used to extract the footnotes.

If `nicematrix` is loaded with the option `footnotehyper`, the package `footnotehyper` is loaded (if it is not yet loaded) and it is used to extract footnotes.

Caution: The packages `footnote` and `footnotehyper` are incompatible. The package `footnotehyper` is the successor of the package `footnote` and should be used preferently. The package `footnote` has some drawbacks, in particular: it must be loaded after the package `xcolor` and it is not perfectly compatible with `hyperref`.

### 11.2 The notes of tabular

The package `nicematrix` also provides a command `\tabularnote` which gives the ability to specify notes that will be composed at the end of the array with a width of line equal to the width of the array (excepted the potential exterior columns). With no surprise, that command is available only in the environments without delimiters, that is to say `{NiceTabular}`, `{NiceArray}` and `{NiceMatrix}`.

In fact, this command is available only if the extension `enumitem` has been loaded (before or after `nicematrix`). Indeed, the notes are composed at the end of the array with a type of list provided by the package `enumitem`.

```

\begin{NiceTabular}{@{}llr@{}}[first-row,code-for-first-row = \bfseries]
\toprule
Last name & First name & Birth day \\
\midrule
Achard\tabularnote{Achard is an old family of the Poitou.}
& Jacques & 5 juin 1962 \\
Lefebvre\tabularnote{The name Lefebvre is an alteration of the name Lefebure.}
& Mathilde & 23 mai 1988 \\
Vanesse & Stephany & 30 octobre 1994 \\
Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}

```

Last name	First name	Birth day
Achard <sup>a</sup>	Jacques	June 5, 2005
Lefebvre <sup>b</sup>	Mathilde	January 23, 1975
Vanesse	Stephany	October 30, 1994
Dupont	Chantal	January 15, 1998

<sup>a</sup> Achard is an old family of the Poitou.

<sup>b</sup> The name Lefebvre is an alteration of the name Lefebure.

- If you have several successive commands `\tabularnote{...}` with no space at all between them, the labels of the corresponding notes are composed together, separated by commas (this is similar to the option `multiple` of `footmisc` for the footnotes).
- If a command `\tabularnote{...}` is exactly at the end of a cell (with no space at all after), the label of the note is composed in an overlapping position (towards the right). This structure may provide a better alignment of the cells of a given column.
- If the key `notes/para` is used, the notes are composed at the end of the array in a single paragraph (as with the key `para` of `threeparttable`).
- There is a key `tabularnote` which provides a way to insert some text in the zone of the notes before the numbered tabular notes.
- If the package `booktabs` has been loaded (before or after `nicematrix`), the key `notes/bottomrule` draws a `\bottomrule` of `booktabs` after the notes.
- The command `\tabularnote` may be used *before* the environment of `nicematrix`. Thus, it's possible to use it on the title inserted by `\caption` in an environment `{table}` of LaTeX.
- It's possible to create a reference to a tabular note created by `\tabularnote` (with the usual command `\label` used after the `\tabularnote`).

For an illustration of some of those remarks, see table 1, p. 25. This table has been composed with the following code.

```

\begin{table}
\setlength{\belowcaptionskip}{1ex}
\centering
\caption{Use of \texttt{\textbackslash tabularnote}\tabularnote{It's possible
to put a note in the caption.}}
\label{t:tabularnote}
\begin{NiceTabular}{@{}llc@{}}
[notes/bottomrule, tabularnote = Some text before the notes.]
\toprule
Last name & First name & Length of life \\
\midrule

```



```
Churchill & Wiston & 91\\
Nightingale\{tabularnote{Considered as the first nurse of
history.}\} \{tabularnote{Nicknamed ``the Lady with the Lamp''.}\}
& Florence & 90 \\
Schoelcher & Victor & 89\{tabularnote{The label of the note is overlapping.}\}\\
Touchet & Marie & 89 \\
Wallis & John & 87 \\
\bottomrule
\end{NiceTabular}
\end{table}
```

Table 1: Use of `\tabularnote`<sup>a</sup>

Last name	First name	Length of life
Churchill	Wiston	91
Nightingale <sup>b,c</sup>	Florence	90
Schoelcher	Victor	89 <sup>d</sup>
Touchet	Marie	89
Wallis	John	87

Some text before the notes.

<sup>a</sup> It's possible to put a note in the caption.

<sup>b</sup> Considered as the first nurse of history.

<sup>c</sup> Nicknamed “the Lady with the Lamp”.

<sup>d</sup> The label of the note is overlapping.

### 11.3 Customisation of the tabular notes

The tabular notes can be customized with a set of keys available in `\NiceMatrixOptions`. The name of these keys is prefixed by `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`

For sake of commodity, it is also possible to set these keys in `\NiceMatrixOptions` via a key `notes` which takes in as value a list of pairs `key=value` where the name of the keys need no longer be prefixed by `notes`:

```

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

```

We detail these keys.

- The key `notes/para` requires the composition of the notes (at the end of the tabular) in a single paragraph.

Initial value: `false`

That key is also available within a given environment.

- The key `notes/bottomrule` adds a `\bottomrule` of `booktabs` *after* the notes. Of course, that rule is drawn only if there is really notes in the tabular. The package `booktabs` must have been loaded (before or after the package `nicematrix`). If it is not, an error is raised.

Initial value: `false`

That key is also available within a given environment.

- The key `notes/style` is a command whose argument is specified by `#1` and which gives the style of numerotation of the notes. That style will be used by `\ref` when referencing a tabular note marked with a command `\label`. The labels formatted by that style are used, separated by commas, when the user puts several consecutive commands `\tabularnote`. The marker `#1` is meant to be the name of a LaTeX counter.

Initial value: `\textit{\alph{#1}}`

Another possible value should be a mere `\arabic{#1}`

- The key `notes/label-in-tabular` is a command whose argument is specified by `#1` which is used when formatting the label of a note in the tabular. Internally, this number of note has already been formatted by `notes/style` before sent to that command.

Initial value: `\textsuperscript{#1}`

In French, it's a tradition of putting a small space before the label of note. That tuning could be achieved by the following code:

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

- The key `notes/label-in-list` is a command whose argument is specified by `#1` which is used when formatting the label in the list of notes at the end of the tabular. Internally, this number of note has already been formatted by `notes/style` before sent to that command.

Initial value: `\textsuperscript{#1}`

In French, the labels of notes are not composed in upper position when composing the notes. Such behaviour could be achieved by:

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

The command `\nobreak` is for the event that the option `para` is used.

- The notes are composed at the end of the tabular by using internally a style of list of `enumitem`. The key `notes/enumitem-keys` specifies a list of pairs `key=value` (following the specifications of `enumitem`) to customize that type of list.

Initial value: `noitemsep , leftmargin = * , align = left , labelsep = Opt`

This initial value contains the specification `align = left` which requires a composition of the label leftwards in the box affected to that label. With that tuning, the notes are composed flush left, which is pleasant when composing tabulars in the spirit of `booktabs` (see for example the table 1, p. 25).

- The key `notes/enumitem-keys-para` is similar to the previous one but corresponds to the type of list used when the option `para` is in force. Of course, when the option `para` is used, a list of type `inline` (as called by `enumitem`) is used and the pairs `key=value` should correspond to such a list of type `inline`.

Initial value: `afterlabel = \nobreak, itemjoin = \quad`

- The key `notes/code-before` is a token list inserted by `nicematrix` just before the composition of the notes at the end of the tabular.

Initial value: `empty`

For example, if one wishes to compose all the notes in gray and `\footnotesize`, he should use that key:

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

It's also possible to add `\raggedright` or `\RaggedRight` in that key (`\RaggedRight` is a command of `ragged2e`).

For an example of customisation of the tabular notes, see p. 36.

## 11.4 Use of `{NiceTabular}` with `threeparttable`

If you wish to use the environment `{NiceTabular}` or `{NiceTabular*}` in an environment `{threeparttable}` of the eponymous package, you have to patch the environment `{threeparttable}` with the following code (with a version of LaTeX at least 2020/10/01).

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

## 12 Other features

### 12.1 Use of the column type `S` of `siunitx`

If the package `siunitx` is loaded (before or after `nicematrix`), it's possible to use the `S` column type of `siunitx` in the environments of `nicematrix`. The implementation doesn't use explicitly any private macro of `siunitx`.

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

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

On the other hand, the `d` columns of the package `dcolumn` are not supported by `nicematrix`.

## 12.2 Alignment option in {NiceMatrix}

The environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.) provide two options `l` and `r` which generate all the columns aligned leftwards (or rightwards).

```


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


```

## 12.3 The command \rotate

The package `nicematrix` provides a command `\rotate`. When used in the beginning of a cell, this command composes the contents of the cell after a rotation of  $90^\circ$  in the direct sens.

In the following command, we use that command in the `code-for-first-row`.

```

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

```

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

If the command `\rotate` is used in the “last row” (exterior to the matrix), the corresponding elements are aligned upwards as shown below.

```

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

```

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

## 12.4 The option small

With the option `small`, the environments of the package `nicematrix` are composed in a way similar to the environment `{smallmatrix}` of the package `amsmath` (and the environments `{psmallmatrix}`, `{bsmallmatrix}`, etc. of the package `mathtools`).

```


$$\begin{bmatrix} 1 & -2 & 3 & 4 & 5 \\ 0 & 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 2 & 3 \end{bmatrix} \begin{matrix} L_2 \leftarrow 2L_1 - L_2 \\ L_3 \leftarrow L_1 + L_3 \end{matrix}$$


```

$$\begin{bmatrix} 1 & -2 & 3 & 4 & 5 \\ 0 & 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 2 & 3 \end{bmatrix} \begin{matrix} L_2 \leftarrow 2L_1 - L_2 \\ L_3 \leftarrow L_1 + L_3 \end{matrix}$$

One should note that the environment `{NiceMatrix}` with the option `small` is not composed *exactly* as the environment `{smallmatrix}`. Indeed, all the environments of `nicematrix` are constructed upon

`{array}` (of the package `array`) whereas the environment `{smallmatrix}` is constructed directly with an `\halign` of TeX.

In fact, the option `small` corresponds to the following tuning:

- the cells of the array are composed with `\scriptstyle`;
- `\arraystretch` is set to 0.47;
- `\arraycolsep` is set to 1.45 pt;
- the characteristics of the dotted lines are also modified.

## 12.5 The counters `iRow` and `jCol`

In the cells of the array, it's possible to use the LaTeX counters `iRow` and `jCol` which represent the number of the current row and the number of the current column<sup>24</sup>. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

In the `code-before` (cf. p. 11) and in the `\CodeAfter` (cf. p. 21), `iRow` represents the total number of rows (excepted the potential exterior rows) and `jCol` represents the total number of columns (excepted the potential exterior columns).

```
$\begin{pNiceMatrix}% don't forget the %
  [first-row,
   first-col,
   code-for-first-row = \mathbf{\alpha{jCol}} ,
   code-for-first-col = \mathbf{\arabic{iRow}} ]
& & & & \\
& 1 & 2 & 3 & 4 \\
& 5 & 6 & 7 & 8 \\
& 9 & 10 & 11 & 12
\end{pNiceMatrix}$
```

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

If LaTeX counters called `iRow` and `jCol` are defined in the document by packages other than `nicematrix` (or by the final user), they are shadowed in the environments of `nicematrix`.

The package `nicematrix` also provides commands in order to compose automatically matrices from a general pattern. These commands are `\AutoNiceMatrix`, `\pAutoNiceMatrix`, `\bAutoNiceMatrix`, `\vAutoNiceMatrix`, `\VAutoNiceMatrix` and `\BAutoNiceMatrix`.

These commands take in two mandatory arguments. The first is the format of the matrix, with the syntax  $n \times p$  where  $n$  is the number of rows and  $p$  the number of columns. The second argument is the pattern (it's a list of tokens which are inserted in each cell of the constructed matrix, excepted in the cells of the potential exterior rows and columns).

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

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

<sup>24</sup>We recall that the exterior “first row” (if it exists) has the number 0 and that the exterior “first column” (if it exists) has also the number 0.

## 12.6 The option `light-syntax`

The option `light-syntax` (inspired by the package `spalign`) allows the user to compose the arrays with a lighter syntax, which gives a better legibility of the TeX source.

When this option is used, one should use the semicolon for the end of a row and spaces or tabulations to separate the columns. However, as usual in the TeX world, the spaces after a control sequence are discarded and the elements between curly braces are considered as a whole.

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

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

It's possible to change the character used to mark the end of rows with the option `end-of-row`. As said before, the initial value is a semicolon.

When the option `light-syntax` is used, it is not possible to put verbatim material (for example with the command `\verb`) in the cells of the array.<sup>25</sup>

## 12.7 Color of the delimiters

For the environments with delimiters (`{pNiceArray}`, `{pNiceMatrix}`, etc.), it's possible to change the color of the delimiters with the key `delimiters/color`.<sup>26</sup>

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

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

## 12.8 The environment `{NiceArrayWithDelims}`

In fact, the environment `{pNiceArray}` and its variants are based upon a more general environment, called `{NiceArrayWithDelims}`. The first two mandatory arguments of this environment are the left and right delimiters used in the construction of the matrix. It's possible to use `{NiceArrayWithDelims}` if we want to use atypical or asymmetrical delimiters.

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

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

# 13 Use of Tikz with `nicematrix`

## 13.1 The nodes corresponding to the contents of the cells

The package `nicematrix` creates a PGF/Tikz node for each (non-empty) cell of the considered array. These nodes are used to draw the dotted lines between the cells of the matrix (inter alia).

**Caution** : By default, no node is created in a empty cell.

However, it's possible to impose the creation of a node with the command `\NotEmpty`.<sup>27</sup>

<sup>25</sup>The reason is that, when the option `light-syntax` is used, the whole content of the environment is loaded as a TeX argument to be analyzed. The environment doesn't behave in that case as a standard environment of LaTeX which only put TeX commands before and after the content.

<sup>26</sup>`delimiters-color` is a synonymous (deprecated) for `delimiters/color`.

<sup>27</sup>One should note that, with that command, the cell is considered as non-empty, which has consequences for the continuous dotted lines (cf. p. 16).

The nodes of a document must have distinct names. That’s why the names of the nodes created by `nicematrix` contains the number of the current environment. Indeed, the environments of `nicematrix` are numbered by a internal global counter.

In the environment with the number  $n$ , the node of the row  $i$  and column  $j$  has for name `nm-n-i-j`. The command `\NiceMatrixLastEnv` provides the number of the last environment of `nicematrix` (for LaTeX, it’s a “fully expandable” command and not a counter).

However, it’s advisable to use instead the key `name`. This key gives a name to the current environment. When the environment has a name, the nodes are accessible with the name “*name-i-j*” where *name* is the name given to the array and  $i$  and  $j$  the numbers of row and column. It’s possible to use these nodes with PGF but the final user will probably prefer to use Tikz (which is a convenient layer upon PGF). However, one should remind that `nicematrix` doesn’t load Tikz by default. In the following examples, we assume that Tikz has been loaded.

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

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

Don’t forget the options `remember picture` and `overlay`.

In the `\CodeAfter`, the things are easier : one must refer to the nodes with the form  $i-j$  (we don’t have to indicate the environment which is of course the current environment).

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

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

In the following example, we have underlined all the nodes of the matrix (we explain below the technic used : cf. p. 40).

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

## 13.2 The “medium nodes” and the “large nodes”

In fact, the package `nicematrix` can create “extra nodes”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.<sup>28</sup>

These nodes are not used by `nicematrix` by default, and that’s why they are not created by default.

The names of the “medium nodes” are constructed by adding the suffix “-medium” to the names of the “normal nodes”. In the following example, we have underlined the “medium nodes”. We consider that this example is self-explanatory.

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

The names of the “large nodes” are constructed by adding the suffix “-large” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.<sup>29</sup>

<sup>28</sup>There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

<sup>29</sup>There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 15).

$$\left( \begin{array}{|c|c|c|} \hline a & a+b & a+b+c \\ \hline a & a & a+b \\ \hline a & a & a \\ \hline \end{array} \right)$$

The “large nodes” of the first column and last column may appear too small for some usage. That’s why it’s possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the “large nodes” of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.<sup>30</sup>

$$\left( \begin{array}{|c|c|c|} \hline a & a+b & a+b+c \\ \hline a & a & a+b \\ \hline a & a & a \\ \hline \end{array} \right)$$

It’s also possible to add more space on both side of the array with the options `extra-left-margin` and `extra-right-margin`. These margins are not incorporated in the “large nodes”. It’s possible to fix both values with the option `extra-margin` and, in the following example, we use `extra-margin` with the value 3 pt.

$$\left( \begin{array}{|c|c|c|} \hline a & a+b & a+b+c \\ \hline a & a & a+b \\ \hline a & a & a \\ \hline \end{array} \right)$$

**Be careful :** These nodes are reconstructed from the contents of the contents cells of the array. Usually, they do not correspond to the cells delimited by the rules (if we consider that these rules are drawn).

Here is an array composed with the following code:

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Here, we have colored all the cells of the array with `\chessboardcolors`.

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Here are the “large nodes” of this array (without use of `margin` nor `extra-margin`).

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

### 13.3 The nodes which indicate the position of the rules

**New 5.11** The package `nicematrix` creates a PGF/Tikz node merely called *i* (with the classical prefix) at the intersection of the horizontal rule of number *i* and the vertical rule of number *j* (more specifically the potential position of those rules because maybe there are not actually drawn). These nodes are available in the `code-before` and the `\CodeAfter`.

<sup>30</sup>The options `left-margin` and `right-margin` take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.



rose	tulipe	lys
arum	iris	violette
muguet	dahlia	souci

If we use Tikz (we remind that `nicematrix` does not load Tikz by default, by only PGF, which is a sub-layer of Tikz), we can access, in the `\CodeAfter` but also in the `code-before`) to the intersection of the horizontal rule  $i$  and the vertical rule  $j$  with the syntax  $(i-j)$ .

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

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

### 13.4 The nodes corresponding to the command `\SubMatrix`

**New 5.10** The command `\SubMatrix` available in the `\CodeAfter` has been described p. 22.

If a command `\SubMatrix` has been used with the key `name` with an expression such as `name=MyName` three PGF/Tikz nodes are created with the names `MyName-left`, `MyName` and `MyName-right`.

The nodes `MyName-left` and `MyName-right` correspond to the delimiters left and right and the node `MyName` correspond to the submatrix itself.

In the following example, we have highlighted these nodes (the submatrix itself has been created with `\SubMatrix\{{2-2}{3-3}\}`).

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

## 14 API for the developpers

The package `nicematrix` provides two variables which are internal but public<sup>31</sup>:

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

These variables contain the code of what we have called the “code-before” and the “code-after”. The developer can use them to add code from a cell of the array (the affectation must be global, allowing to exit the cell, which is a TeX group).

One should remark that the use of `\g_nicematrix_code_before_tl` needs one compilation more (because the instructions are written on the `aux` file to be used during the next run).

*Example* : We want to write a command `\hatchcell` to hatch the current cell (with an optional argument between brackets for the color). It's possible to program such command `\hatchcell` as follows, explicitly using the public variable `\g_nicematrix_code_before_tl` (this code requires the Tikz library `patterns`: `\usetikzlibrary{patterns}`).

```
ExplSyntaxOn
\cs_new_protected:Nn \__pantigny_hatch:nnn
{
  \begin { tikzpicture }
  \fill [ pattern = north-west-lines , pattern-color = #3 ]
  ( #1 -| #2 ) rectangle ( \int_eval:n { #1 + 1 } -| \int_eval:n { #2 + 1 } ) ;
  \end { tikzpicture }
}

\NewDocumentCommand \hatchcell { ! 0 { black } }
{
  \tl_gput_right:Nx \g_nicematrix_code_before_tl
  { \__pantigny_hatch:nnn { \arabic { iRow } } { \arabic { jCol } } { #1 } }
}
\ExplSyntaxOff
```

Here is an example of use:

```
\begin{NiceTabular}{ccc}[hvlines]
Tokyo & Paris & London \\
Lima & \hatchcell[blue!30]Oslo & Miami \\
Los Angeles & Madrid & Roma
\end{NiceTabular}
```

Tokyo	Paris	London
Lima	Oslo	Miami
Los Angeles	Madrid	Roma

## 15 Technical remarks

### 15.1 Definition of new column types

The package `nicematrix` provides the command `\OnlyMainNiceMatrix` which is meant to be used in definitions of new column types. Its argument is evaluated if and only if we are in the main part of the array, that is to say not in a potential exterior row.

For example, one may wish to define a new column type ? in order to draw a (black) heavy rule of width 1 pt. The following definition will do the job<sup>32</sup>:

<sup>31</sup>According to the LaTeX3 conventions, each variable with name beginning with `\g_nicematrix` ou `\l_nicematrix` is public and each variable with name beginning with `\g__nicematrix` or `\l__nicematrix` is private.

<sup>32</sup>The command `\vrule` is a TeX (and not LaTeX) command.

```
\newcolumnntype{?}{\OnlyMainNiceMatrix{\vrule width 1 pt}}
```

The heavy vertical rule won't extend in the exterior rows.<sup>33</sup>

```
$\begin{pNiceArray}{cc?cc}[first-row,last-row=3]
```

```
C_1 & C_2 & C_3 & C_4 \\\
```

```
a & b & c & d \\\
```

```
e & f & g & h \\\
```

```
C_1 & C_2 & C_3 & C_4
```

```
\end{pNiceArray}$
```

$$\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ \left( \begin{array}{cc|cc} a & b & c & d \\ e & f & g & h \end{array} \right) \\ C_1 & C_2 & C_3 & C_4 \end{array}$$

This specifier ? may be used in the standard environments `{tabular}` and `{array}` (of the package `array`) and, in this case, the command `\OnlyMainNiceMatrix` is no-op.

## 15.2 Diagonal lines

By default, all the diagonal lines<sup>34</sup> of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That's why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

In the following examples, the first `\Ddots` instruction is written in color:

Example with parallelization (default):

```
$A = \begin{pNiceMatrix}
```

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

```
a+b & \Ddots & & \Vdots \\\
```

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

```
a+b & \Cdots & a+b & 1
```

```
\end{pNiceMatrix}$
```

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

```
$A = \begin{pNiceMatrix}
```

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

```
a+b & & & \Vdots \\\
```

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

```
a+b & \Cdots & a+b & 1
```

```
\end{pNiceMatrix}$
```

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

It's possible to turn off the parallelization with the option `parallelize-diags` set to `false`:

The same example without parallelization:

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

It's possible to specify the instruction `\Ddots` which will be drawn first (and which will be used to draw the other diagonal dotted lines when the parallelization is in force) with the key `draw-first`: `\Ddots[draw-first]`.

## 15.3 The “empty” cells

An instruction like `\Ldots`, `\Cdots`, etc. tries to determine the first non-empty cell on both sides. However, an “empty cell” is not necessarily a cell with no TeX content (that is to say a cell with no token between the two ampersands `&`). The precise rules are as follow.

- An implicit cell is empty. For example, in the following matrix:

<sup>33</sup>Of course, such rule is defined by the classical technics of `nicematrix` and, for this reason, won't cross the double rules of `\hline\hline`.

<sup>34</sup>We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

```

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

```

the last cell (second row and second column) is empty.

- Each cell whose TeX output has a width equal to zero is empty.
- A cell containing the command `\NotEmpty` is not empty (and a PGF/Tikz node) is created in that cell.
- A cell with a command `\Hspace` (or `\Hspace*`) is empty. This command `\Hspace` is a command defined by the package `nicematrix` with the same meaning as `\hspace` except that the cell where it is used is considered as empty. This command can be used to fix the width of some columns of the matrix without interfering with `nicematrix`.

## 15.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea<sup>35</sup>. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`<sup>36</sup>. The package `nicematrix` does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of `nicematrix` are not affected).

## 15.5 Incompatibilities

The package `nicematrix` is not fully compatible with the package `arydshln` (because this package redefines many internal of `array`).

Anyway, in order to use `arydshln`, one must first free the letter “:” by giving a new letter for the vertical dotted rules of `nicematrix`:

```
\NiceMatrixOptions{letter-for-dotted-lines=;}
```

As for now, the package `nicematrix` is not compatible with `aastex63`. If you want to use `nicematrix` with `aastex63`, send me an email and I will try to solve the incompatibilities.

# 16 Examples

## 16.1 Notes in the tabulars

The tools provided by `nicematrix` for the composition of the tabular notes have been presented in the section 11 p. 23.

<sup>35</sup>In the documentation of `{amsmath}`, we can read: *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).*

<sup>36</sup>And not by inserting `@{}` on both sides of the preamble of the array. As a consequence, the length of the `\hline` is not modified and may appear too long, in particular when using square brackets

Let's consider that we wish to number the notes of a tabular with stars.<sup>37</sup>

First, we write a command `\stars` similar the well-known commands `\arabic`, `\alph`, `\Alph`, etc. which produces a number of stars equal to its argument <sup>38</sup>

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

Of course, we change the style of the labels with the key `notes/style`. However, it would be interesting to change also some parameters in the type of list used to compose the notes at the end of the tabular. First, we required a composition flush right for the labels with the setting `align=right`. Moreover, we want the labels to be composed on a width equal to the width of the widest label. The widest label is, of course, the label with the greatest number of stars. We know that number: it is equal to `\value{tabularnote}` (because `tabularnote` is the LaTeX counter used by `\tabularnote` and, therefore, at the end of the tabular, its value is equal to the total number of tabular notes). We use the key `widest*` of `enumitem` in order to require a width equal to that value: `widest*=\value{tabularnote}`.

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

\begin{NiceTabular}{{}}{llr{}}[first-row,code-for-first-row = \bfseries]
\toprule
Last name & First name & Birth day \\
\midrule
Achard\tabularnote{Achard is an old family of the Poitou.}
& Jacques & 5 juin 1962 \\
Lefebvre\tabularnote{The name Lefebvre is an alteration of the name Lefebure.}
& Mathilde & 23 mai 1988 \\
Vanesse & Stephany & 30 octobre 1994 \\
Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```

Last name	First name	Birth day
Achard*	Jacques	June 5, 2005
Lefebvre**	Mathilde	January 23, 1975
Vanesse	Stephany	October 30, 1994
Dupont	Chantal	January 15, 1998

\*Achard is an old family of the Poitou.

\*\*The name Lefebvre is an alteration of the name Lefebure.

<sup>37</sup>Of course, it's realistic only when there is very few notes in the tabular.

<sup>38</sup>In fact: the value of its argument.

## 16.2 Dotted lines

An example with the resultant of two polynomials:

```
\setlength{\extrarowheight}{1mm}
\[\begin{vNiceArray}{ccccccc}[columns-width=6mm]
a_0 & & & & b_0 & & \\
a_1 & & \Ddots & & b_1 & & \Ddots \\
& \Vdots & & \Ddots & & \Ddots & b_0 \\
a_p & & & a_0 & & & b_1 \\
& & \Ddots & & a_1 & & \Vdots \\
& & & \Vdots & & \Ddots & \\
& & & & a_p & & b_q \\
\end{vNiceArray}\]
```

$$\left| \begin{array}{ccccccc} a_0 & & & & b_0 & & \\ a_1 & & \cdots & & b_1 & & \cdots \\ & \vdots & & \cdots & & \cdots & b_0 \\ a_p & & & a_0 & & & b_1 \\ & & \cdots & & a_1 & & \vdots \\ & & & \vdots & & \cdots & \\ & & & & a_p & & b_q \end{array} \right|$$

An example for a linear system:

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

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

## 16.3 Dotted lines which are no longer dotted

The option `line-style` controls the style of the lines drawn by `\Ldots`, `\Cdots`, etc. Thus, it's possible with these commands to draw lines which are not longer dotted.

```
\NiceMatrixOptions{code-for-first-row = \scriptstyle,code-for-first-col = \scriptstyle}
\setcounter{MaxMatrixCols}{12}
\newcommand{\blue}{\color{blue}}
\[\begin{pNiceMatrix}[last-row,last-col,nullify-dots,xdots/line-style={dashed,blue}]
```

```

1& & & \Vdots & & & \Vdots \\
& \Ddots[line-style=standard] \\
& & 1 \\
\Cdots[color=blue,line-style=dashed]& & & \blue 0 &
\Cdots & & & \blue 1 & & & \Cdots & \blue \leftarrow i \\
& & & & 1 \\
& & & \Vdots & & \Ddots[line-style=standard] & & \Vdots \\
& & & & & & 1 \\
\Cdots & & & \blue 1 & \Cdots & & \Cdots & \blue 0 & & & \Cdots & \blue \leftarrow j \\
& & & & & & & & & & 1 \\
& & & & & & & \Ddots[line-style=standard] \\
& & & \Vdots & & & & \Vdots & & & 1 \\
& & & \blue \overset{\uparrow}{i} & & & \blue \overset{\uparrow}{j} \\
\end{pNiceMatrix}\]

```

$$\begin{pmatrix}
 1 & & & & \\
 & \ddots & & & \\
 & & 1 & & \\
 & & & 0 & 1 \\
 & & & & \ddots \\
 & & & & & 1 \\
 & & & & & & \ddots \\
 & & & & & & & 0 & 1 \\
 & & & & & & & & \ddots \\
 & & & & & & & & & 1 \\
 & & & & & & & & & & \ddots \\
 & & & & & & & & & & & 1
 \end{pmatrix}
 \begin{matrix}
 \\
 \\
 \\
 \leftarrow i \\
 \\
 \\
 \\
 \leftarrow j \\
 \\
 \\
 \\
 \\
 \end{matrix}$$

In fact, it's even possible to draw solid lines with the commands `\Cdots`, `\Vdots`, etc.

```

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

```

$$\begin{matrix}
 \xrightarrow{\text{n columns}} \\
 \begin{pmatrix}
 1 & 1 & 1 & \dots & 1 \\
 1 & 1 & 1 & & 1 \\
 1 & 1 & 1 & & 1 \\
 1 & 1 & 1 & & 1 \\
 1 & 1 & 1 & \dots & 1
 \end{pmatrix} \\
 \uparrow \text{n rows}
 \end{matrix}$$

## 16.4 Width of the columns

In the following example, we use `{NiceMatrixBlock}` with the option `auto-columns-width` because we want the same automatic width for all the columns of the matrices.

```

\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
{ last-col,code-for-last-col = \color{blue}\scriptstyle,light-syntax}
\setlength{\extrarowheight}{1mm}
$\begin{pNiceArray}{cccc:c}
1 1 1 1 1 {} ;
2 4 8 16 9 ;
3 9 27 81 36 ;
4 16 64 256 100
\end{pNiceArray}$
\medskip
$\begin{pNiceArray}{cccc:c}
1 1 1 1 1 ;
0 2 6 14 7 { L_2 \gets -2 L_1 + L_2 } ;
0 6 24 78 33 { L_3 \gets -3 L_1 + L_3 } ;
0 12 60 252 96 { L_4 \gets -4 L_1 + L_4 }
\end{pNiceArray}$
...
\end{NiceMatrixBlock}

```

$$\begin{array}{c}
\left( \begin{array}{ccccc} 1 & 1 & 1 & 1 & \vdots & 1 \\ 2 & 4 & 8 & 16 & \vdots & 9 \\ 3 & 9 & 27 & 81 & \vdots & 36 \\ 4 & 16 & 64 & 256 & \vdots & 100 \end{array} \right) \\
\left( \begin{array}{ccccc} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 2 & 6 & 14 & \vdots & 7 \\ 0 & 6 & 24 & 78 & \vdots & 33 \\ 0 & 12 & 60 & 252 & \vdots & 96 \end{array} \right) \begin{array}{l} L_2 \leftarrow -2L_1 + L_2 \\ L_3 \leftarrow -3L_1 + L_3 \\ L_4 \leftarrow -4L_1 + L_4 \end{array} \\
\left( \begin{array}{ccccc} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 1 & 3 & 7 & \vdots & \frac{7}{2} \\ 0 & 3 & 12 & 39 & \vdots & \frac{33}{2} \\ 0 & 1 & 5 & 21 & \vdots & 8 \end{array} \right) \begin{array}{l} L_2 \leftarrow \frac{1}{2}L_2 \\ L_3 \leftarrow \frac{1}{2}L_3 \\ L_4 \leftarrow \frac{1}{12}L_4 \end{array}
\end{array}
\quad \left| \quad
\begin{array}{c}
\left( \begin{array}{ccccc} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 1 & 3 & 7 & \vdots & \frac{7}{2} \\ 0 & 0 & 3 & 18 & \vdots & 6 \\ 0 & 0 & -2 & -14 & \vdots & -\frac{9}{2} \end{array} \right) \begin{array}{l} L_3 \leftarrow -3L_2 + L_3 \\ L_4 \leftarrow -L_2 - L_4 \end{array} \\
\left( \begin{array}{ccccc} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 1 & 3 & 7 & \vdots & \frac{7}{2} \\ 0 & 0 & 1 & 6 & \vdots & 2 \\ 0 & 0 & -2 & -14 & \vdots & -\frac{9}{2} \end{array} \right) \begin{array}{l} L_3 \leftarrow \frac{1}{3}L_3 \\ L_4 \leftarrow -L_3 + L_4 \end{array} \\
\left( \begin{array}{ccccc} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 1 & 3 & 7 & \vdots & \frac{7}{2} \\ 0 & 0 & 1 & 6 & \vdots & 2 \\ 0 & 0 & 0 & -2 & \vdots & -\frac{1}{2} \end{array} \right) \begin{array}{l} L_4 \leftarrow L_3 + L_4 \end{array}
\end{array}$$

## 16.5 How to highlight cells of a matrix

In order to highlight a cell of a matrix, it's possible to “draw” that cell with the key `draw` of the command `\Block` (this is one of the uses of a mono-cell block<sup>39</sup>).

```

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

```

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

<sup>39</sup>We recall that, if the first mandatory argument of the command `\Block` is left empty, that means that the block is a mono-cell block



We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the command `\hline`, the specifier “|” and the options `hlines`, `vlines` and `hvlines` spread the cells.<sup>40</sup>

It's possible to color a row with `\rowcolor` in the `code-before` (or with `\rowcolor` in the first cell of the row if the key `colortbl-like` is used—even when `colortbl` is not loaded).

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

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

However, it's not possible to do a fine tuning. That's why we describe now a method to highlight a row of the matrix. We create a rectangular Tikz node which encompasses the nodes of the second row with the Tikz library `fit`. This Tikz node is filled after the construction of the matrix. In order to see the text *under* this node, we have to use transparency with the `blend mode` equal to `multiply`.

**Caution** : Some PDF readers are not able to show transparency.<sup>41</sup>

That example and the following ones require Tikz (by default, `nicematrix` only loads PGF, which is a sub-layer of Tikz) and the Tikz library `fit`. The following lines in the preamble of your document do the job:

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

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

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

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

<sup>40</sup>For the command `\cline`, see the remark p. 7.

<sup>41</sup>In Overleaf, the “built-in” PDF viewer does not show transparency. You can switch to the “native” viewer in that case.

We recall that, for a rectangle of merged cells (with the command `\Block`), a Tikz node is created for the set of merged cells with the name *i-j-block* where *i* and *j* are the number of the row and the number of the column of the upper left cell (where the command `\Block` has been issued). If the user has required the creation of the `medium` nodes, a node of this type is also created with a name suffixed by `-medium`.

```

$\begin{pNiceMatrix}[margin,create-medium-nodes]
  \Block{3-3}<\Large>\{A\} & & 0 \\
  & \hspace*{1cm} & \Vdots \\
  & & 0 \\
  0 & \Cdots & 0 & 0
\CodeAfter
  \tikz \node [highlight = (1-1-block-medium)] {};
\end{pNiceMatrix}$

```

$$\begin{pmatrix} \boxed{A} & 0 \\ & \vdots \\ & 0 \\ 0 & \dots & 0 & 0 \end{pmatrix}$$

Consider now the following matrix which we have named `example`.

```

$\begin{pNiceArray}{ccc}[name=example,last-col,create-medium-nodes]
a & a + b & a + b + c & L_1 \\
a & a & a + b & L_2 \\
a & a & a & L_3
\end{pNiceArray}$

```

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

If we want to highlight each row of this matrix, we can use the previous technique three times.

```

\tikzset{mes-options/.style={remember picture,
                             overlay,
                             name prefix = exemple-,
                             highlight/.style = {fill = red!15,
                                                  blend mode = multiply,
                                                  inner sep = 0pt,
                                                  fit = #1}}}

\begin{tikzpicture}[mes-options]
\node [highlight = (1-1) (1-3)] {};
\node [highlight = (2-1) (2-3)] {};
\node [highlight = (3-1) (3-3)] {};
\end{tikzpicture}

```

We obtain the following matrix.

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

The result may seem disappointing. We can improve it by using the “medium nodes” instead of the “normal nodes”.

```

\begin{tikzpicture}[mes-options, name suffix = -medium]
\node [highlight = (1-1) (1-3)] {};
\node [highlight = (2-1) (2-3)] {};
\node [highlight = (3-1) (3-3)] {};
\end{tikzpicture}

```

We obtain the following matrix.

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

## 16.6 Direct use of the Tikz nodes

In the following example, we illustrate the mathematical product of two matrices.

The use of `\NiceMatrixBlock` with the option `auto-columns-width` gives the same width for all the columns and, therefore, a perfect alignment of the two superposed matrices.

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

```
\NiceMatrixOptions{nullify-dots}
```

The three matrices will be displayed using an environment `{array}` (an environment `{tabular}` may also be possible).

```
$\begin{array}{cc}
```

```
&
```

The matrix  $B$  has a “first row” (for  $C_j$ ) and that’s why we use the key `first-row`.

```
\begin{bNiceArray}{c>{\strut}cccc}[name=B,first-row]
```

```
& & C_j & & \\
```

```
b_{11} & \Cdots & b_{1j} & \Cdots & b_{1n} \\
```

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

```
& & b_{kj} & & \\
```

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

```
b_{n1} & \Cdots & b_{nj} & \Cdots & b_{nn}
```

```
\end{bNiceArray} \\ \\
```

The matrix  $A$  has a “first column” (for  $L_i$ ) and that’s why we use the key `first-col`.

```
\begin{bNiceArray}{cc>{\strut}ccc}[name=A,first-col]
```

```
& a_{11} & \Cdots & & & a_{1n} \\
```

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

```
L_i & a_{i1} & \Cdots & a_{ik} & \Cdots & a_{in} \\
```

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

```
& a_{n1} & \Cdots & & & a_{nn}
```

```
\end{bNiceArray}
```

```
&
```

In the matrix product, the two dotted lines have an open extremity.

```
\begin{bNiceArray}{cc>{\strut}ccc}
```

```
& & & & \\
```

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

```
\Cdots & & c_{ij} & & \\
```

```
\\
```

```
\\
```

```
\end{bNiceArray}
```

```
\end{array}$
```

```
\end{NiceMatrixBlock}
```

```
\begin{tikzpicture}[remember picture, overlay]
```

```
\node [highlight = (A-3-1) (A-3-5) ] {} ;
```

```
\node [highlight = (B-1-3) (B-5-3) ] {} ;
```

```
\draw [color = gray] (A-3-3) to [bend left] (B-3-3) ;
```

```
\end{tikzpicture}
```

$$L_i \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \dots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \quad \begin{bmatrix} b_{11} & \dots & b_{1n} \\ \vdots & & \vdots \\ b_{k1} & \dots & b_{kn} \\ \vdots & & \vdots \\ b_{n1} & \dots & b_{nn} \end{bmatrix}$$

## 17 Implementation

By default, the package `nicematrix` doesn't patch any existing code.

However, when the option `renew-dots` is used, the commands `\cdots`, `\ldots`, `\dots`, `\vdots`, `\ddots` and `\iddots` are redefined in the environments provided by `nicematrix` as explained previously. In the same way, if the option `renew-matrix` is used, the environment `{matrix}` of `amsmath` is redefined.

On the other hand, the environment `{array}` is never redefined.

Of course, the package `nicematrix` uses the features of the package `array`. It tries to be independent of its implementation. Unfortunately, it was not possible to be strictly independent. For example, the package `nicematrix` relies upon the fact that the package `{array}` uses `\ialign` to begin the `\halign`.

### Declaration of the package and packages loaded

The prefix `nicematrix` has been registered for this package.

See: <http://mirrors.ctan.org/macros/latex/contrib/l3kernel/l3prefixes.pdf>

<@@=nicematrix>

First, we load `pgfcore` and the module `shapes`. We do so because it's not possible to use `\usepgfmodule` in `\ExplSyntaxOn`.

```
1 \RequirePackage{pgfcore}
2 \usepgfmodule{shapes}
```

We give the traditional declaration of a package written with `expl3`:

```
3 \RequirePackage{l3keys2e}
4 \ProvidesExplPackage
5   {nicematrix}
6   {\myfiledate}
7   {\myfileversion}
8   {Enhanced arrays with the help of PGF/TikZ}
```

The command for the treatment of the options of `\usepackage` is at the end of this package for technical reasons.

We load some packages. The package `xparse` is still loaded for use on Overleaf.

```
9 \RequirePackage { xparse }
10 \RequirePackage { array }
11 \RequirePackage { amsmath }
```

```

12 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { nicematrix } }
13 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { nicematrix } }
14 \cs_new_protected:Npn \@@_error:nnn { \msg_error:nnnn { nicematrix } }
15 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { nicematrix } }
16 \cs_new_protected:Npn \@@_fatal:nn { \msg_fatal:nnn { nicematrix } }
17 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { nicematrix } }
18 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { nicematrix } }

19 \cs_new_protected:Npn \@@_msg_redirect_name:nn
20 { \msg_redirect_name:nnn { nicematrix } }

```

## Technical definitions

```

21 \bool_new:N \c_@@_in_preamble_bool
22 \bool_set_true:N \c_@@_in_preamble_bool
23 \AtBeginDocument { \bool_set_false:N \c_@@_in_preamble_bool }

24 \bool_new:N \c_@@_arydshln_loaded_bool
25 \bool_new:N \c_@@_booktabs_loaded_bool
26 \bool_new:N \c_@@_enumitem_loaded_bool
27 \bool_new:N \c_@@_tikz_loaded_bool
28 \AtBeginDocument
29 {
30   \@ifpackageloaded { arydshln }
31   { \bool_set_true:N \c_@@_arydshln_loaded_bool }
32   { }
33   \@ifpackageloaded { booktabs }
34   { \bool_set_true:N \c_@@_booktabs_loaded_bool }
35   { }
36   \@ifpackageloaded { enumitem }
37   { \bool_set_true:N \c_@@_enumitem_loaded_bool }
38   { }
39   \@ifpackageloaded { tikz }
40   {

```

In some constructions, we will have to use a `{pgfpicture}` which *must* be replaced by a `{tikzpicture}` if Tikz is loaded. However, this switch between `{pgfpicture}` and `{tikzpicture}` can't be done dynamically with a conditional because, when the Tikz library `external` is loaded by the user, the pair `\tikzpicture`–`\endtikzpicture` (or `\begin{tikzpicture}`–`\end{tikzpicture}`) must be statically “visible” (even when externalization is not activated).

That's why we create `\c_@@_pgfortikzpicture_tl` and `\c_@@_endpgfortikzpicture_tl` which will be used to construct in a `\AtBeginDocument` the correct version of some commands.

```

41   \bool_set_true:N \c_@@_tikz_loaded_bool
42   \tl_const:Nn \c_@@_pgfortikzpicture_tl { \exp_not:N \tikzpicture }
43   \tl_const:Nn \c_@@_endpgfortikzpicture_tl { \exp_not:N \endtikzpicture }
44 }
45 {
46   \tl_const:Nn \c_@@_pgfortikzpicture_tl { \exp_not:N \pgfpicture }
47   \tl_const:Nn \c_@@_endpgfortikzpicture_tl { \exp_not:N \endpgfpicture }
48 }
49 }

```

We test whether the current class is `revtex4-1` (deprecated) or `revtex4-2` because these classes redefines `\array` (of `array`) in a way incompatible with our programming. At the date January 2021, the current version `revtex4-2` is 4.2e (compatible with `booktabs`).

```

50 \bool_new:N \c_@@_revtex_bool
51 \ifclassloaded { revtex4-1 }
52 { \bool_set_true:N \c_@@_revtex_bool }
53 { }
54 \ifclassloaded { revtex4-2 }
55 { \bool_set_true:N \c_@@_revtex_bool }
56 { }

```

Maybe one of the previous classes will be loaded inside another class... We try to detect that situation.

```
57 \cs_if_exist:NT \rvtx@ifformat@geq { \bool_set_true:N \c_@@_revtex_bool }
58 \cs_generate_variant:Nn \tl_if_single_token_p:n { V }
```

We define a command `\iddots` similar to `\ddots` (`\ddots`) but with dots going forward (`\iddots`). We use `\ProvideDocumentCommand` and so, if the command `\iddots` has already been defined (for example by the package `mathdots`), we don't define it again.

```
59 \ProvideDocumentCommand \iddots { }
60 {
61   \mathinner
62   {
63     \tex_mkern:D 1 mu
64     \box_move_up:nn { 1 pt } { \hbox:n { . } }
65     \tex_mkern:D 2 mu
66     \box_move_up:nn { 4 pt } { \hbox:n { . } }
67     \tex_mkern:D 2 mu
68     \box_move_up:nn { 7 pt }
69     { \vbox:n { \kern 7 pt \hbox:n { . } } }
70     \tex_mkern:D 1 mu
71   }
72 }
```

This definition is a variant of the standard definition of `\ddots`.

In the `aux` file, we will have the references of the PGF/Tikz nodes created by `nicematrix`. However, when `booktabs` is used, some nodes (more precisely, some `row` nodes) will be defined twice because their position will be modified. In order to avoid an error message in this case, we will redefine `\pgfutil@check@rerun` in the `aux` file.

```
73 \AtBeginDocument
74 {
75   \@ifpackageloaded { booktabs }
76   { \iow_now:Nn \mainaux \nicematrix@redefine@check@rerun }
77   { }
78 }
79 \cs_set_protected:Npn \nicematrix@redefine@check@rerun
80 {
81   \cs_set_eq:NN \@@_old_pgfutil@check@rerun \pgfutil@check@rerun
```

The new version of `\pgfutil@check@rerun` will not check the PGF nodes whose names start with `nm-` (which is the prefix for the nodes created by `nicematrix`).

```
82   \cs_set_protected:Npn \pgfutil@check@rerun ##1 ##2
83   {
84     \str_if_eq:eeF { nm- } { \tl_range:nnn { ##1 } 1 3 }
85     { \@@_old_pgfutil@check@rerun { ##1 } { ##2 } }
86   }
87 }
```

We have to know whether `colortbl` is loaded in particular for the redefinition of `\everycr`.

```
88 \bool_new:N \c_@@_colortbl_loaded_bool
89 \AtBeginDocument
90 {
91   \@ifpackageloaded { colortbl }
92   { \bool_set_true:N \c_@@_colortbl_loaded_bool }
93   { }
```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. We will use it to store the instruction of color for the rules even if `colortbl` is not loaded.

```
94   \cs_set_protected:Npn \CT@arc@ { }
95   \cs_set:Npn \arrayrulecolor #1 # { \CT@arc@ { #1 } }
96   \cs_set:Npn \CT@arc@ #1 #2
97   {
98     \dim_compare:nNt \baselineskip = \c_zero_dim \noalign
99     { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }
```

```

100     }
Idem for \CT@drs@.
101     \cs_set_protected:Npn \CT@drsc@ { }
102     \cs_set:Npn \doublerulesepcolor #1 # { \CT@drs { #1 } }
103     \cs_set:Npn \CT@drs #1 #2
104     {
105         \dim_compare:nNnT \baselineskip = \c_zero_dim \noalign
106         { \cs_gset:Npn \CT@drsc@ { \color #1 { #2 } } }
107     }
108     \cs_set:Npn \hline
109     {
110         \noalign { \ifnum 0 = ` } \fi
111         \cs_set_eq:NN \hskip \vskip
112         \cs_set_eq:NN \vrule \hrule
113         \cs_set_eq:NN \@width \@height
114         { \CT@arc@ \vline }
115         \futurelet \reserved@a
116         \@xhline
117     }
118 }
119 }

```

We have to redefine `\cline` for several reasons. The command `\@@_cline` will be linked to `\cline` in the beginning of `{NiceArrayWithDelims}`. The following commands must *not* be protected.

```

120 \cs_set:Npn \@@_standard_cline #1 { \@@_standard_cline:w #1 \q_stop }
121 \cs_set:Npn \@@_standard_cline:w #1-#2 \q_stop
122 {
123     \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
124     \int_compare:nNnT { #1 } > 1 { \multispan { \@@_pred:n { #1 } } & }
125     \multispan { \int_eval:n { #2 - #1 + 1 } }
126     {
127         \CT@arc@
128         \leaders \hrule \@height \arrayrulewidth \hfill

```

The following `\skip_horizontal:N \c_zero_dim` is to prevent a potential `\unskip` to delete the `\leaders`<sup>42</sup>

```

129     \skip_horizontal:N \c_zero_dim
130 }

```

Our `\everycr` has been modified. In particular, the creation of the `row` node is in the `\everycr` (maybe we should put it with the incrementation of `\c@iRow`). Since the following `\cr` correspond to a “false row”, we have to nullify `\everycr`.

```

131 \everycr { }
132 \cr
133 \noalign { \skip_vertical:N -\arrayrulewidth }
134 }

```

The following version of `\cline` spreads the array of a quantity equal to `\arrayrulewidth` as does `\hline`. It will be loaded excepted if the key `standard-cline` has been used.

```

135 \cs_set:Npn \@@_cline

```

We have to act in a fully expandable way since there may be `\noalign` (in the `\multispan`) to detect. That’s why we use `\@@_cline_i:en`.

```

136 { \@@_cline_i:en \l_@@_first_col_int }

```

The command `\cline_i:nn` has two arguments. The first is the number of the current column (it *must* be used in that column). The second is a standard argument of `\cline` of the form *i-j*.

```

137 \cs_set:Npn \@@_cline_i:nn #1 #2 { \@@_cline_i:w #1-#2 \q_stop }
138 \cs_set:Npn \@@_cline_i:w #1-#2-#3 \q_stop
139 {

```

Now, `#1` is the number of the current column and we have to draw a line from the column `#2` to the column `#3` (both included).

---

<sup>42</sup>See question 99041 on TeX StackExchange.

```

140 \int_compare:nNnT { #1 } < { #2 }
141 { \multispan { \int_eval:n { #2 - #1 } } & }
142 \multispan { \int_eval:n { #3 - #2 + 1 } }
143 {
144   \CT@arc@
145   \leaders \hrule \@height \arrayrulewidth \hfill
146   \skip_horizontal:N \c_zero_dim
147 }

```

You look whether there is another `\cline` to draw (the final user may put several `\cline`).

```

148 \peek_meaning_remove_ignore_spaces:NTF \cline
149 { & \@@_cline_i:en { \@@_succ:n { #3 } } }
150 { \everycr { } \cr }
151 }
152 \cs_generate_variant:Nn \@@_cline_i:nn { e n }

```

The following commands are only for efficiency. They must *not* be protected because it will be used (for instance) in names of PGF nodes.

```

153 \cs_new:Npn \@@_succ:n #1 { \the \numexpr #1 + 1 \relax }
154 \cs_new:Npn \@@_pred:n #1 { \the \numexpr #1 - 1 \relax }

```

The following command is a small shortcut.

```

155 \cs_new:Npn \@@_math_toggle_token:
156 { \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token }

```

```

157 \cs_new_protected:Npn \@@_set_CT@arc@:
158 { \peek_meaning:NTF [ \@@_set_CT@arc@_i: \@@_set_CT@arc@_ii: }
159 \cs_new_protected:Npn \@@_set_CT@arc@_i: [ #1 ] #2 \q_stop
160 { \cs_set:Npn \CT@arc@ { \color [ #1 ] { #2 } } }
161 \cs_new_protected:Npn \@@_set_CT@arc@_ii: #1 \q_stop
162 { \cs_set:Npn \CT@arc@ { \color { #1 } } }

```

## The column S of siunitx

We want to know whether the package `siunitx` is loaded and, if it is loaded, we redefine the `S` columns of `siunitx`.

```

163 \bool_new:N \c_@@_siunitx_loaded_bool
164 \AtBeginDocument
165 {
166   \@ifpackageloaded { siunitx }
167   { \bool_set_true:N \c_@@_siunitx_loaded_bool }
168   { }
169 }

```

The command `\NC@rewrite@S` is a LaTeX command created by `siunitx` in connection with the `S` column. In the code of `siunitx`, this command is defined by:

```

\renewcommand*{\NC@rewrite@S}[1]{}
{
  \@temptokena \exp_after:wN
  {
    \tex_the:D \@temptokena
    > { \__siunitx_table_collect_begin: S {#1} }
    c
    < { \__siunitx_table_print: }
  }
  \NC@find
}

```

We want to patch this command (in the environments of `nicematrix`) in order to have:



```

\renewcommand*{\NC@rewrite@S}[1] []
{
  \@temptokena \exp_after:wN
  {
    \tex_the:D \@temptokena
    > { \@@_Cell: \__siunitx_table_collect_begin: S {#1} }
    \@@_true_c:
    < { \__siunitx_table_print: \@@_end_Cell: }
  }
  \NC@find
}

```

However, we don't want to use explicitly any private command of `siunitx`. That's why we will extract the name of the two `\__siunitx...` commands by their position in the code of `\NC@rewrite@S`. Since the command `\NC@rewrite@S` appends some tokens to the *toks* list `\@temptokena`, we use the LaTeX command `\NC@rewrite@S` in a group (`\group_begin:-\group_end:`) and we extract the two command names which are in the *toks* `\@temptokena`. However, this extraction can be done only when `siunitx` is loaded (and it may be loaded after `nicematrix`) and, in fact, after the beginning of the document — because some instructions of `siunitx` are executed in a `\AtBeginDocument`). That's why this extraction will be done only at the first use of an environment of `nicematrix` with the command `\@@_adapt_S_column:`.

```

170 \cs_set_protected:Npn \@@_adapt_S_column:
171 {
172   \bool_if:NT \c_@@_siunitx_loaded_bool
173   {
174     \group_begin:
175     \@temptokena = { }

```

We protect `\NC@find` which is at the end of `\NC@rewrite@S`.

```

176   \cs_set_eq:NN \NC@find \prg_do_nothing:
177   \NC@rewrite@S { }

```

Conversion of the *toks* `\@temptokena` in a token list of `expl3` (the *toks* are not supported by `expl3` but we can, nevertheless, use the option `V` for `\tl_gset:NV`).

```

178   \tl_gset:NV \g_tmpa_tl \@temptokena
179   \group_end:
180   \tl_new:N \c_@@_table_collect_begin_tl
181   \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
182   \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
183   \tl_new:N \c_@@_table_print_tl
184   \tl_gset:Nx \c_@@_table_print_tl { \tl_item:Nn \g_tmpa_tl { -1 } }

```

The token lists `\c_@@_table_collect_begin_tl` and `\c_@@_table_print_tl` contain now the two commands of `siunitx`.

If the adaptation has been done, the command `\@@_adapt_S_column:` becomes no-op (globally).

```

185   \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
186   }
187 }

```

The command `\@@_renew_NC@rewrite@S:` will be used in each environment of `nicematrix` in order to “rewrite” the *S* column in each environment.

```

188 \AtBeginDocument
189 {
190   \bool_if:nTF { ! \c_@@_siunitx_loaded_bool }
191   { \cs_set_eq:NN \@@_renew_NC@rewrite@S: \prg_do_nothing: }
192   {
193     \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
194     {
195       \renewcommand*{\NC@rewrite@S}[1] []
196       {
197         \@temptokena \exp_after:wN
198         {
199           \tex_the:D \@temptokena
200           > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }

```

`\@@_true_c:` will be replaced statically by `c` at the end of the construction of the preamble.

```

201         \@@_true_c:
202         < { \c_@@_table_print_tl \@@_end_Cell: }
203         }
204     \NC@find
205 }
206 }
207 }
208 }
```

The following regex will be used to modify the preamble of the array when the key `colortbl-like` is used.

```

209 \regex_const:Nn \c_@@_columncolor_regex { \c { columncolor } }
```

If the final user uses `nicematrix`, PGF/Tikz will write instruction `\pgfsyspdfmark` in the `aux` file. If he changes its mind and no longer loads `nicematrix`, an error may occur at the next compilation because of remanent instructions `\pgfsyspdfmark` in the `aux` file. With the following code, we try to avoid that situation.

```

210 \cs_new_protected:Npn \@@_provide_pgfsyspdfmark:
211 {
212     \iow_now:Nn \@mainaux
213     {
214         \ExplSyntaxOn
215         \cs_if_free:NT \pgfsyspdfmark
216         { \cs_set_eq:NN \pgfsyspdfmark \@gobblethree }
217         \ExplSyntaxOff
218     }
219     \cs_gset_eq:NN \@@_provide_pgfsyspdfmark: \prg_do_nothing:
220 }
```

## Parameters

For compatibility with versions prior to 5.0, we provide a load-time option `define_L_C_R`. With this option, it's possible to use the letters `L`, `C` and `R` instead of `l`, `c` and `r` in the preamble of the environments of `nicematrix` as it was mandatory before version 5.0.

```

221 \bool_new:N \c_@@_define_L_C_R_bool
222 \cs_new_protected:Npn \@@_define_L_C_R:
223 {
224     \newcolumntype L l
225     \newcolumntype C c
226     \newcolumntype R r
227 }
```

The following counter will count the environments `{NiceArray}`. The value of this counter will be used to prefix the names of the Tikz nodes created in the array.

```

228 \int_new:N \g_@@_env_int
```

The following command is only a syntactic shortcut. It must *not* be protected (it will be used in names of PGF nodes).

```

229 \cs_new:Npn \@@_env: { nm - \int_use:N \g_@@_env_int }
```

The command `\NiceMatrixLastEnv` is not used by the package `nicematrix`. It's only a facility given to the final user. It gives the number of the last environment (in fact the number of the current environment but it's meant to be used after the environment in order to refer to that environment — and its nodes — without having to give it a name). This command *must* be expandable since it will be used in `pgf` nodes.

```

230 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
231 { \int_use:N \g_@@_env_int }
```

The following command is only a syntactic shortcut. The `q` in `qpoint` means *quick*.

```
232 \cs_new_protected:Npn \@@_qpoint:n #1
233   { \pgfpointanchor { \@@_env: - #1 } { center } }
```

The following counter will count the environments `{NiceMatrixBlock}`.

```
234 \int_new:N \g_@@_NiceMatrixBlock_int
```

The dimension `\l_@@_columns_width_dim` will be used when the options specify that all the columns must have the same width (but, if the key `columns-width` is used with the special value `auto`, the boolean `\l_@@_auto_columns_width_bool` also will be raised).

```
235 \dim_new:N \l_@@_columns_width_dim
```

The following token list will contain the type of the current cell (`l`, `c` or `r`). It will be used by the blocks.

```
236 \tl_new:N \l_@@_cell_type_tl
237 \tl_set:Nn \l_@@_cell_type_tl { c }
```

When there is a mono-column block (created by the command `\Block`), we want to take into account the width of that block for the width of the column. That's why we compute the width of that block in the `\g_@@_blocks_wd_dim` and, after the construction of the box `\l_@@_cell_box`, we change the width of that box to take into account the length `\g_@@_blocks_wd_dim`.

```
238 \dim_new:N \g_@@_blocks_wd_dim
```

Idem pour the mono-row blocks.

```
239 \dim_new:N \g_@@_blocks_ht_dim
240 \dim_new:N \g_@@_blocks_dp_dim
```

The sequence `\g_@@_names_seq` will be the list of all the names of environments used (via the option `name`) in the document: two environments must not have the same name. However, it's possible to use the option `allow-duplicate-names`.

```
241 \seq_new:N \g_@@_names_seq
```

We want to know whether we are in an environment of `nicematrix` because we will raise an error if the user tries to use nested environments.

```
242 \bool_new:N \l_@@_in_env_bool
```

If the user uses `{NiceArray}` or `{NiceTabular}` the flag `\l_@@_NiceArray_bool` will be raised.

```
243 \bool_new:N \l_@@_NiceArray_bool
```

If the user uses `{NiceTabular}` or `{NiceTabular*}`, we will raise the following flag.

```
244 \bool_new:N \l_@@_NiceTabular_bool
```

If the user uses `{NiceTabular*}`, the width of the tabular (in the first argument of the environment `{NiceTabular*}`) will be stored in the following dimension.

```
245 \dim_new:N \l_@@_tabular_width_dim
```

If the user uses an environment without preamble, we will raise the following flag.

```
246 \bool_new:N \l_@@_Matrix_bool
```

The following boolean will be raised when the command `\rotate` is used.

```
247 \bool_new:N \g_@@_rotate_bool
```

```

248 \cs_new_protected:Npn \@@_test_if_math_mode:
249 {
250   \if_mode_math: \else:
251     \@@_fatal:n { Outside-math-mode }
252   \fi:
253 }

```

The letter used for the vlins which will be drawn only in the sub-matrices. `vlism` stands for *vertical lines in sub-matrices*.

```

254 \tl_new:N \l_@@_letter_vlism_tl

```

The list of the columns where vertical lines in sub-matrices (vlism) must be drawn. Of course, the actual value of this sequence will be known after the analyse of the preamble of the array.

```

255 \seq_new:N \g_@@_cols_vlism_seq

```

The following colors will be used to memorize the color of the potential “first col” and the potential “first row”.

```

256 \colorlet { nicematrix-last-col } { . }
257 \colorlet { nicematrix-last-row } { . }

```

The following string is the name of the current environment or the current command of `nicematrix` (despite its name which contains *env*).

```

258 \str_new:N \g_@@_name_env_str

```

The following string will contain the word *command* or *environment* whether we are in a command of `nicematrix` or in an environment of `nicematrix`. The default value is *environment*.

```

259 \tl_set:Nn \g_@@_com_or_env_str { environment }

```

The following command will be able to reconstruct the full name of the current command or environment (despite its name which contains *env*). This command must *not* be protected since it will be used in error messages and we have to use `\str_if_eq:VnTF` and not `\tl_if_eq:NnTF` because we need to be fully expandable).

```

260 \cs_new:Npn \@@_full_name_env:
261 {
262   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
263     { command \space \c_backslash_str \g_@@_name_env_str }
264     { environment \space \{ \g_@@_name_env_str \} }
265 }

```

The following token list corresponds to the option `code-after` (it’s also possible to set the value of that parameter with the keyword `\CodeAfter`).

```

266 \tl_new:N \g_nicematrix_code_after_tl

```

The following token list has a function similar to `\g_nicematrix_code_after_tl` but it is used internally by `nicematrix`. In fact, we have to distinguish between `\g_nicematrix_code_after_tl` and `\g_@@_internal_code_after_tl` because we must take care of the order in which instructions stored in that parameters are executed.

```

267 \tl_new:N \g_@@_internal_code_after_tl

```

The counters `\l_@@_old_iRow_int` and `\l_@@_old_jCol_int` will be used to save the values of the potential LaTeX counters `iRow` and `jCol`. These LaTeX counters will be restored at the end of the environment.

```

268 \int_new:N \l_@@_old_iRow_int
269 \int_new:N \l_@@_old_jCol_int

```

The TeX counters `\c@iRow` and `\c@jCol` will be created in the beginning of `{NiceArrayWithDelims}` (if they don't exist previously).

The following token list corresponds to the key `rules/color` available in the environments.

```
270 \tl_new:N \l_@@_rules_color_tl
```

This boolean will be used only to detect in an expandable way whether we are at the beginning of the (potential) column zero, in order to raise an error if `\Hdotsfor` is used in that column.

```
271 \bool_new:N \g_@@_after_col_zero_bool
```

A kind of false row will be inserted at the end of the array for the construction of the `col` nodes (and also to fix the width of the columns when `columns-width` is used). When this special row will be created, we will raise the flag `\g_@@_row_of_col_done_bool` in order to avoid some actions set in the redefinition of `\everycr` when the last `\cr` of the `\halign` will occur (after that row of `col` nodes).

```
272 \bool_new:N \g_@@_row_of_col_done_bool
```

It's possible to use the command `\NotEmpty` to specify explicitly that a cell must be considered as non empty by `nicematrix` (the Tikz nodes are constructed only in the non empty cells).

```
273 \bool_new:N \g_@@_not_empty_cell_bool
```

`\l_@@_code_before_tl` may contain two types of informations:

- A `code-before` written in the `aux` file by a previous run. When the `aux` file is read, this `code-before` is stored in `\g_@@_code_before_i_tl` (where *i* is the number of the environment) and, at the beginning of the environment, it will be put in `\l_@@_code_before_tl`.
- The final user can explicitly add material in `\l_@@_code_before_tl` by using the key `code-before`.

```
274 \tl_new:N \l_@@_code_before_tl
```

```
275 \bool_new:N \l_@@_code_before_bool
```

The following dimensions will be used when drawing the dotted lines.

```
276 \dim_new:N \l_@@_x_initial_dim
```

```
277 \dim_new:N \l_@@_y_initial_dim
```

```
278 \dim_new:N \l_@@_x_final_dim
```

```
279 \dim_new:N \l_@@_y_final_dim
```

`expl3` provides scratch dimensions `\l_tmpa_dim` and `\l_tmpb_dim`. We create two more in the same spirit (if they don't exist yet: that's why we use `\dim_zero_new:N`).

```
280 \dim_zero_new:N \l_tmpc_dim
```

```
281 \dim_zero_new:N \l_tmpd_dim
```

Some cells will be declared as “empty” (for example a cell with an instruction `\Cdots`).

```
282 \bool_new:N \g_@@_empty_cell_bool
```

The following dimensions will be used internally to compute the width of the potential “first column” and “last column”.

```
283 \dim_new:N \g_@@_width_last_col_dim
```

```
284 \dim_new:N \g_@@_width_first_col_dim
```

The following sequence will contain the characteristics of the blocks of the array, specified by the command `\Block`. Each block is represented by 6 components surrounded by curly braces: `{imin}{jmin}{imax}{jmax}{options}{contents}`.

The variable is global because it will be modified in the cells of the array.

```
285 \seq_new:N \g_@@_blocks_seq
```

We also manage a sequence of the *positions* of the blocks. Of course, it's redundant with the previous sequence, but it's for efficiency. In that sequence, each block is represented by only the four first components:  $\{imin\}\{jmin\}\{imax\}\{jmax\}$ .

```
286 \seq_new:N \g_@@_pos_of_blocks_seq
```

In fact, this sequence will also contain the positions of the cells with a `\diagbox`. The sequence `\g_@@_pos_of_blocks_seq` will be used when we will draw the rules (which respect the blocks).

We will also manage a sequence for the positions of the dotted lines. These dotted lines are created in the array by `\Cdots`, `\Vdots`, `\Ddots`, etc. However, their positions, that is to say, their extremities, will be determined only after the construction of the array. In this sequence, each item contains four components:  $\{imin\}\{jmin\}\{imax\}\{jmax\}$ .

```
287 \seq_new:N \g_@@_pos_of_xdots_seq
```

The sequence `\g_@@_pos_of_xdots_seq` will be used when we will draw the rules required by the key `hvlines` (these rules won't be drawn within the virtual blocks corresponding to the dotted lines).

The final user may decide to “stroke” a block (using, for example, the key `draw=red!15` when using the command `\Block`). In that case, the rules specified, for instance, by `hvlines` must not be drawn around the block. That's why we keep the information of all that stroken blocks in the following sequence.

```
288 \seq_new:N \g_@@_pos_of_stroken_blocks_seq
```

The list of the names of the potential `\SubMatrix` in the `\CodeAfter` of an environment. Unfortunately, that list has to be global (we have to use it inside the group for the options of a given `\SubMatrix`).

```
289 \seq_new:N \g_@@_submatrix_names_seq
```

We are able to determine the number of columns specified in the preamble (for the environments with explicit preamble, of course and without the potential exterior columns).

```
290 \int_new:N \g_@@_static_num_of_col_int
```

The following token lists correspond to the keys `fill` and `draw` of a command `\Block`.

```
291 \tl_new:N \l_@@_fill_tl
```

```
292 \tl_new:N \l_@@_draw_tl
```

The following token list correspond to the key `color` of the command `\Block`. However, as of now (v. 5.7 of `nicematrix`), the key `color` linked to `fill` with an error. We will give to the key `color` of `\Block` its new meaning in a few months (with its new definition, the key `color` will draw the frame with the given color but also color the content of the block (that is to say the text) as does the key `color` of a Tikz node).

```
293 \tl_new:N \l_@@_color_tl
```

Here is the dimension for the width of the rule when a block (created by `\Block`) is stroked.

```
294 \dim_new:N \l_@@_line_width_dim
```

The parameter of position of the label of a block (`c`, `r` or `l`).

```
295 \tl_new:N \l_@@_pos_of_block_tl
```

```
296 \tl_set:Nn \l_@@_pos_of_block_tl { c }
```

Used when the key `draw-first` is used for `\Ddots` or `\Iddots`.

```
297 \bool_new:N \l_@@_draw_first_bool
```

The blocks which use the key `-` will store their content in a box. These boxes are numbered with the following counter.

```
298 \int_new:N \g_@@_block_box_int
```

```
299 \dim_new:N \l_@@_submatrix_extra_height_dim
```

```
300 \dim_new:N \l_@@_submatrix_left_xshift_dim
```

```
301 \dim_new:N \l_@@_submatrix_right_xshift_dim
```

```
302 \clist_new:N \l_@@_submatrix_hlines_clist
```

```
303 \clist_new:N \l_@@_submatrix_vlines_clist
```

## Variables for the exterior rows and columns

The keys for the exterior rows and columns are `first-row`, `first-col`, `last-row` and `last-col`. However, internally, these keys are not coded in a similar way.

- **First row**

The integer `\l_@@_first_row_int` is the number of the first row of the array. The default value is 1, but, if the option `first-row` is used, the value will be 0.

```
304 \int_new:N \l_@@_first_row_int
305 \int_set:Nn \l_@@_first_row_int 1
```

- **First column**

The integer `\l_@@_first_col_int` is the number of the first column of the array. The default value is 1, but, if the option `first-col` is used, the value will be 0.

```
306 \int_new:N \l_@@_first_col_int
307 \int_set:Nn \l_@@_first_col_int 1
```

- **Last row**

The counter `\l_@@_last_row_int` is the number of the potential “last row”, as specified by the key `last-row`. A value of `-2` means that there is no “last row”. A value of `-1` means that there is a “last row” but we don’t know the number of that row (the key `last-row` has been used without value and the actual value has not still been read in the `aux` file).

```
308 \int_new:N \l_@@_last_row_int
309 \int_set:Nn \l_@@_last_row_int { -2 }
```

If, in an environment like `{pNiceArray}`, the option `last-row` is used without value, we will globally raise the following flag. It will be used to know if we have, after the construction of the array, to write in the `aux` file the number of the “last row”.<sup>43</sup>

```
310 \bool_new:N \l_@@_last_row_without_value_bool
```

Idem for `\l_@@_last_col_without_value_bool`

```
311 \bool_new:N \l_@@_last_col_without_value_bool
```

- **Last column**

For the potential “last column”, we use an integer. A value of `-2` means that there is no last column. A value of `-1` means that we are in an environment without preamble (e.g. `{bNiceMatrix}`) and there is a last column but we don’t know its value because the user has used the option `last-col` without value. A value of 0 means that the option `last-col` has been used in an environment with preamble (like `{pNiceArray}`): in this case, the key was necessary without argument.

```
312 \int_new:N \l_@@_last_col_int
313 \int_set:Nn \l_@@_last_col_int { -2 }
```

However, we have also a boolean. Consider the following code:

```
\begin{pNiceArray}{cc}[last-col]
1 & 2 \\
3 & 4
\end{pNiceArray}
```

---

<sup>43</sup>We can’t use `\l_@@_last_row_int` for this usage because, if `nicematrix` has read its value from the `aux` file, the value of the counter won’t be `-1` any longer.

In such a code, the “last column” specified by the key `last-col` is not used. We want to be able to detect such a situation and we create a boolean for that job.

```
314 \bool_new:N \g_@@_last_col_found_bool
```

This boolean is set to `false` at the end of `\@@_pre_array:`.

## The command `\tablarnote`

The LaTeX counter `tablarnote` will be used to count the tabular notes during the construction of the array (this counter won’t be used during the composition of the notes at the end of the array). You use a LaTeX counter because we will use `\refstepcounter` in order to have the tabular notes referenceable.

```
315 \newcounter { tablarnote }
```

We will store in the following sequence the tabular notes of a given array.

```
316 \seq_new:N \g_@@_tablarnotes_seq
```

However, before the actual tabular notes, it’s possible to put a text specified by the key `tablarnote` of the environment. The token list `\l_@@_tablarnote_tl` corresponds to the value of that key.

```
317 \tl_new:N \l_@@_tablarnote_tl
```

The following counter will be used to count the number of successive tabular notes such as in `\tablarnote{Note 1}\tablarnote{Note 2}\tablarnote{Note 3}`. In the tabular, the labels of those nodes are composed as a comma separated list (e.g.  $a,b,c$ ).

```
318 \int_new:N \l_@@_number_of_notes_int
```

The following function can be redefined by using the key `notes/style`.

```
319 \cs_new:Npn \@@_notes_style:n #1 { \textit { \alph { #1 } } }
```

The following fonction can be redefined by using the key `notes/label-in-tabular`.

```
320 \cs_new:Npn \@@_notes_label_in_tabular:n #1 { \textsuperscript { #1 } }
```

The following function can be redefined by using the key `notes/label-in-list`.

```
321 \cs_new:Npn \@@_notes_label_in_list:n #1 { \textsuperscript { #1 } }
```

We define `\thetablarnote` because it will be used by LaTeX if the user want to reference a footnote which has been marked by a `\label`. The TeX group is for the case where the user has put an instruction such as `\color{red}` in `\@@_notes_style:n`.

```
322 \cs_set:Npn \thetablarnote { { \@@_notes_style:n { tablarnote } } }
```

The tabular notes will be available for the final user only when `enumitem` is loaded. Indeed, the tabular notes will be composed at the end of the array with a list customized by `enumitem` (a list `tablarnotes` in the general case and a list `tablarnotes*` if the key `para` is in force). However, we can test whether `enumitem` has been loaded only at the beginning of the document (we want to allow the user to load `enumitem` after `nicematrix`).

```
323 \AtBeginDocument
324 {
325   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
326   {
327     \NewDocumentCommand \tablarnote { m }
328     { \@@_error:n { enumitem-not-loaded } }
329   }
330 }
```



The type of list `tabularnotes` will be used to format the tabular notes at the end of the array in the general case and `tabularnotes*` will be used if the key `para` is in force.

```

331     \newlist { tabularnotes } { enumerate } { 1 }
332     \setlist [ tabularnotes ]
333     {
334         topsep = Opt ,
335         noitemsep ,
336         leftmargin = * ,
337         align = left ,
338         labelsep = Opt ,
339         label =
340             \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
341     }
342     \newlist { tabularnotes* } { enumerate* } { 1 }
343     \setlist [ tabularnotes* ]
344     {
345         afterlabel = \nobreak ,
346         itemjoin = \quad ,
347         label =
348             \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
349     }

```

The command `\tabularnote` is available in the whole document (and not only in the environments of `nicematrix`) because we want it to be available in the caption of a `{table}` (before the following `{NiceTabular}` or `{NiceArray}`). That's also the reason why the variables `\c@tabularnote` and `\g_@@_tabularnotes_seq` will be cleared at the end of the environment of `nicematrix` (and not at the beginning).

Unfortunately, if the package `caption` is loaded, the command `\caption` evaluates its argument twice and since it is not aware (of course) of `\tabularnote`, the command `\tabularnote` is, in fact, not usable in `\caption` when `caption` is loaded.<sup>44</sup>

```

350     \NewDocumentCommand \tabularnote { m }
351     {
352         \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
353         { \@@_error:n { tabularnote~forbidden } }
354         {

```

`\l_@@_number_of_notes_int` is used to count the number of successive tabular notes such as in `\tabularnote{Note 1}\tabularnote{Note 2}\tabularnote{Note 3}`. We will have to compose the labels of theses notes as a comma separated list (e.g.  $a,b,c$ ).

```

355         \int_incr:N \l_@@_number_of_notes_int

```

We expand the content of the note at the point of use of `\tabularnote` as does `\footnote`.

```

356         \seq_gput_right:Nn \g_@@_tabularnotes_seq { #1 }
357         \peek_meaning:NF \tabularnote
358         {

```

If the following token is *not* a `\tabularnote`, we have finished the sequence of successive commands `\tabularnote` and we have to format the labels of these tabular notes (in the array). We compose those labels in a box `\l_tmpa_box` because we will do a special construction in order to have this box in a overlapping position if we are at the end of a cell.

```

359         \hbox_set:Nn \l_tmpa_box
360         {

```

We remind that it is the command `\@@_notes_label_in_tabular:n` that will (most of the time) put the labels in a `\textsuperscript`.

```

361         \@@_notes_label_in_tabular:n
362         {
363             \stepcounter { tabularnote }
364             \@@_notes_style:n { tabularnote }
365             \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
366             {
367                 ,

```

---

<sup>44</sup>We should try to find a solution to that problem.

```

368             \stepcounter { tabularnote }
369             \@@_notes_style:n { tabularnote }
370         }
371     }
372 }

```

We use `\refstepcounter` in order to have the (last) tabular note referenceable (with the standard command `\label`) and that's why we have to go back with a decrementation of the counter `tabularnote` first.

```

373         \addtocounter { tabularnote } { -1 }
374         \refstepcounter { tabularnote }
375         \int_zero:N \l_@@_number_of_notes_int
376         \hbox_overlap_right:n { \box_use:N \l_tmpa_box }

```

If the command `\tabularnote` is used exactly at the end of the cell, the `\unskip` (inserted by `array`?) will delete the skip we insert now and the label of the footnote will be composed in an overlapping position (by design).

```

377         \skip_horizontal:n { \box_wd:N \l_tmpa_box }
378     }
379 }
380 }
381 }
382 }

```

## Command for creation of rectangle nodes

The following command should be used in a `{pgfpicture}`. It creates a rectangle (empty but with a name).

**#1** is the name of the node which will be created; **#2** and **#3** are the coordinates of one of the corner of the rectangle; **#4** and **#5** are the coordinates of the opposite corner.

```

383 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
384 {
385     \begin { pgfscope }
386     \pgfset
387     {
388         outer~sep = \c_zero_dim ,
389         inner~sep = \c_zero_dim ,
390         minimum~size = \c_zero_dim
391     }
392     \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
393     \pgfnode
394     { rectangle }
395     { center }
396     {
397         \vbox_to_ht:nn
398         { \dim_abs:n { #5 - #3 } }
399         {
400             \vfill
401             \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
402         }
403     }
404     { #1 }
405     { }
406     \end { pgfscope }
407 }

```

The command `\@@_pgf_rect_node:nnn` is a variant of `\@@_pgf_rect_node:nnnnn`: it takes two PGF points as arguments instead of the four dimensions which are the coordinates.

```

408 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
409 {
410     \begin { pgfscope }
411     \pgfset

```

```

412 {
413     outer~sep = \c_zero_dim ,
414     inner~sep = \c_zero_dim ,
415     minimum~size = \c_zero_dim
416 }
417 \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
418 \pgfpointdiff { #3 } { #2 }
419 \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
420 \pgfnode
421 { rectangle }
422 { center }
423 {
424     \vbox_to_ht:nn
425     { \dim_abs:n \l_tmpb_dim }
426     { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
427 }
428 { #1 }
429 { }
430 \end { pgfscope }
431 }

```

## The options

By default, the commands `\cellcolor` and `\rowcolor` are available for the user in the cells of the tabular (the user may use the commands provided by `\colortbl`). However, if the key `colortbl-like` is used, these commands are available.

```
432 \bool_new:N \l_@@_colortbl-like_bool
```

By default, the behaviour of `\cline` is changed in the environments of `nicematrix`: a `\cline` spreads the array by an amount equal to `\arrayrulewidht`. It's possible to disable this feature with the key `\l_@@_standard_line_bool`.

```
433 \bool_new:N \l_@@_standard_cline_bool
```

The following dimensions correspond to the options `cell-space-top-limit` and `co` (these parameters are inspired by the package `cellspace`).

```
434 \dim_new:N \l_@@_cell_space_top_limit_dim
435 \dim_new:N \l_@@_cell_space_bottom_limit_dim
```

The following dimension is the distance between two dots for the dotted lines (when `line-style` is equal to `standard`, which is the initial value). The initial value is 0.45 em but it will be changed if the option `small` is used.

```
436 \dim_new:N \l_@@_inter_dots_dim
437 \AtBeginDocument { \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em } }
```

The `\AtBeginDocument` is only a security in case `revtex4-1` is used (even though it is obsolete).

The following dimension is the minimal distance between a node (in fact an anchor of that node) and a dotted line (we say “minimal” because, by definition, a dotted line is not a continuous line and, therefore, this distance may vary a little).

```
438 \dim_new:N \l_@@_xdots_shorten_dim
439 \AtBeginDocument { \dim_set:Nn \l_@@_xdots_shorten_dim { 0.3 em } }
```

The `\AtBeginDocument` is only a security in case `revtex4-1` is used (even though it is obsolete).

The following dimension is the radius of the dots for the dotted lines (when `line-style` is equal to `standard`, which is the initial value). The initial value is 0.53 pt but it will be changed if the option `small` is used.

```
440 \dim_new:N \l_@@_radius_dim
441 \AtBeginDocument { \dim_set:Nn \l_@@_radius_dim { 0.53 pt } }
```

The `\AtBeginDocument` is only a security in case `revtex4-1` is used (even if it is obsolete).

The token list `\l_@@_xdots_line_style_tl` corresponds to the option `tikz` of the commands `\Cdots`, `\Ldots`, etc. and of the options `line-style` for the environments and `\NiceMatrixOptions`. The constant `\c_@@_standard_tl` will be used in some tests.

```
442 \tl_new:N \l_@@_xdots_line_style_tl
443 \tl_const:Nn \c_@@_standard_tl { standard }
444 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

The boolean `\l_@@_light_syntax_bool` corresponds to the option `light-syntax`.

```
445 \bool_new:N \l_@@_light_syntax_bool
```

The string `\l_@@_baseline_tl` may contain one of the three values `t`, `c` or `b` as in the option of the environment `{array}`. However, it may also contain an integer (which represents the number of the row to which align the array).

```
446 \tl_new:N \l_@@_baseline_tl
447 \tl_set:Nn \l_@@_baseline_tl c
```

The flag `\l_@@_exterior_arraycolsep_bool` corresponds to the option `exterior-arraycolsep`. If this option is set, a space equal to `\arraycolsep` will be put on both sides of an environment `{NiceArray}` (as it is done in `{array}` of `array`).

```
448 \bool_new:N \l_@@_exterior_arraycolsep_bool
```

The flag `\l_@@_parallelize_diags_bool` controls whether the diagonals are parallelized. The initial value is `true`.

```
449 \bool_new:N \l_@@_parallelize_diags_bool
450 \bool_set_true:N \l_@@_parallelize_diags_bool
```

If the flag `\l_@@_vlines_bool` is raised, horizontal space will be reserved in the the preamble of the array (for the vertical rules) and, after the construction of the array, the vertical rules will be drawn.

```
451 \bool_new:N \l_@@_vlines_bool
```

If the flag `\l_@@_hlines_bool` is raised, vertical space will be reserved between the rows of the array (for the horizontal rules) and, after the construction of the array, the vertical rules will be drawn.

```
452 \bool_new:N \l_@@_hlines_bool
```

The flag `\l_@@_except_corners_bool` will be raised when the key `except-corners` will be used. In that case, the corners will be computed before we draw rules and the rules won't be drawn in the corners. As expected, the key `hvlines-except-corners` raises the key `except-corners`.

```
453 \clist_new:N \l_@@_except_corners_clist
454 \dim_new:N \l_@@_notes_above_space_dim
455 \AtBeginDocument { \dim_set:Nn \l_@@_notes_above_space_dim { 1 mm } }
```

The `\AtBeginDocument` is only a security in case `revtex4-1` is used (even if it is obsolete).

The flag `\l_@@_nullify_dots_bool` corresponds to the option `nullify-dots`. When the flag is down, the instructions like `\vdots` are inserted within a `\hphantom` (and so the constructed matrix has exactly the same size as a matrix constructed with the classical `{matrix}` and `\ldots`, `\vdots`, etc.).

```
456 \bool_new:N \l_@@_nullify_dots_bool
```

The following flag will be used when the current options specify that all the columns of the array must have the same width equal to the largest width of a cell of the array (except the cells of the potential exterior columns).

```
457 \bool_new:N \l_@@_auto_columns_width_bool
```

The string `\l_@@_name_str` will contain the optional name of the environment: this name can be used to access to the Tikz nodes created in the array from outside the environment.

```
458 \str_new:N \l_@@_name_str
```

The boolean `\l_@@_medium_nodes_bool` will be used to indicate whether the “medium nodes” are created in the array. Idem for the “large nodes”.

```
459 \bool_new:N \l_@@_medium_nodes_bool
```

```
460 \bool_new:N \l_@@_large_nodes_bool
```

The dimension `\l_@@_left_margin_dim` correspond to the option `left-margin`. Idem for the right margin. These parameters are involved in the creation of the “medium nodes” but also in the placement of the delimiters and the drawing of the horizontal dotted lines (`\hdottedline`).

```
461 \dim_new:N \l_@@_left_margin_dim
```

```
462 \dim_new:N \l_@@_right_margin_dim
```

The dimensions `\l_@@_extra_left_margin_dim` and `\l_@@_extra_right_margin_dim` correspond to the options `extra-left-margin` and `extra-right-margin`.

```
463 \dim_new:N \l_@@_extra_left_margin_dim
```

```
464 \dim_new:N \l_@@_extra_right_margin_dim
```

The token list `\l_@@_end_of_row_tl` corresponds to the option `end-of-row`. It specifies the symbol used to mark the ends of rows when the light syntax is used.

```
465 \tl_new:N \l_@@_end_of_row_tl
```

```
466 \tl_set:Nn \l_@@_end_of_row_tl { ; }
```

The following parameter is for the color the dotted lines drawn by `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots`, `\iddots` and `\Hdotsfor` but *not* the dotted lines drawn by `\hdottedline` and “:”.

```
467 \tl_new:N \l_@@_xdots_color_tl
```

The following token list corresponds to the key `delimiters/color`.

```
468 \tl_new:N \l_@@_delimiters_color_tl
```

Sometimes, we want to have several arrays vertically juxtaposed in order to have an alignment of the columns of these arrays. To achieve this goal, one may wish to use the same width for all the columns (for example with the option `columns-width` or the option `auto-columns-width` of the environment `{NiceMatrixBlock}`). However, even if we use the same type of delimiters, the width of the delimiters may be different from an array to another because the width of the delimiter is fonction of its size. That’s why we create an option called `max-delimiter-width` which will give to the delimiters the width of a delimiter (of the same type) of big size. The following boolean corresponds to this option.

```
469 \bool_new:N \l_@@_max_delimiter_width_bool
```

```
470 \keys_define:nn { NiceMatrix / xdots }
```

```
471 {
```

```
472   line-style .code:n =
```

```
473   {
```

```
474     \bool_lazy_or:nnTF
```

We can’t use `\c_@@_tikz_loaded_bool` to test whether tikz is loaded because `\NiceMatrixOptions` may be used in the preamble of the document.

```
475     { \cs_if_exist_p:N \tikzpicture }
```

```
476     { \str_if_eq_p:nn { #1 } { standard } }
```

```
477     { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
```

```
478     { \@@_error:n { bad-option~for~line-style } }
```

```
479   } ,
```

```
480   line-style .value_required:n = true ,
```

```
481   color .tl_set:N = \l_@@_xdots_color_tl ,
```

```
482   color .value_required:n = true ,
```

```
483   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
```

```
484   shorten .value_required:n = true ,
```

The options `down` and `up` are not documented for the final user because he should use the syntax with `^` and `_`.

```
485   down .tl_set:N = \l_@@_xdots_down_tl ,
486   up .tl_set:N = \l_@@_xdots_up_tl ,
```

The key `draw-first`, which is meant to be used only with `\Ddots` and `\Iddots`, which be caught when `\Ddots` or `\Iddots` is used (during the construction of the array and not when we draw the dotted lines).

```
487   draw-first .code:n = \prg_do_nothing: ,
488   unknown .code:n = \@@_error:n { Unknown-key-for-xdots }
489 }
```

```
490 \keys_define:nn { NiceMatrix / rules }
491 {
492   color .tl_set:N = \l_@@_rules_color_tl ,
493   color .value_required:n = true ,
494   width .dim_set:N = \arrayrulewidth ,
495   width .value_required:n = true
496 }
```

First, we define a set of keys “NiceMatrix / Global” which will be used (with the mechanism of `.inherit:n`) by other sets of keys.

```
497 \keys_define:nn { NiceMatrix / Global }
498 {
499   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
500   rules .value_required:n = true ,
501   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
502   standard-cline .default:n = true ,
503   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
504   cell-space-top-limit .value_required:n = true ,
505   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
506   cell-space-bottom-limit .value_required:n = true ,
507   cell-space-limits .meta:n =
508   {
509     cell-space-top-limit = #1 ,
510     cell-space-bottom-limit = #1 ,
511   } ,
512   cell-spaces-limits .value_required:n = true ,
513   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
514   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
515   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
516   light-syntax .default:n = true ,
517   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
518   end-of-row .value_required:n = true ,
519   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
520   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
521   last-row .int_set:N = \l_@@_last_row_int ,
522   last-row .default:n = -1 ,
523   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
524   code-for-first-col .value_required:n = true ,
525   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
526   code-for-last-col .value_required:n = true ,
527   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
528   code-for-first-row .value_required:n = true ,
529   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
530   code-for-last-row .value_required:n = true ,
531   hlines .bool_set:N = \l_@@_hlines_bool ,
532   vlines .bool_set:N = \l_@@_vlines_bool ,
533   vlines-in-sub-matrix .code:n =
534   {
535     \tl_if_single_token:nTF { #1 }
536     { \tl_set:Nn \l_@@_letter_vlism_tl { #1 } }
```

```

537     { \l_@@_error:n { One-letter-allowed } }
538   } ,
539   vlines-in-sub-matrix .value_required:n = true ,
540   hvlines .code:n =
541   {
542     \bool_set_true:N \l_@@_vlines_bool
543     \bool_set_true:N \l_@@_hlines_bool
544   } ,
545   parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool ,

```

With the option `renew-dots`, the command `\cdots`, `\ldots`, `\vdots`, `\ddots`, etc. are redefined and behave like the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots`, etc.

```

546   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
547   renew-dots .value_forbidden:n = true ,
548   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
549   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
550   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
551   create-extra-nodes .meta:n =
552   { create-medium-nodes , create-large-nodes } ,
553   left-margin .dim_set:N = \l_@@_left_margin_dim ,
554   left-margin .default:n = \arraycolsep ,
555   right-margin .dim_set:N = \l_@@_right_margin_dim ,
556   right-margin .default:n = \arraycolsep ,
557   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
558   margin .default:n = \arraycolsep ,
559   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
560   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
561   extra-margin .meta:n =
562   { extra-left-margin = #1 , extra-right-margin = #1 } ,
563   extra-margin .value_required:n = true ,
564 }

```

We define a set of keys used by the environments of `nicematrix` (but not by the command `\NiceMatrixOptions`).

```

565 \keys_define:nn { NiceMatrix / Env }
566 {
567   except-corners .clist_set:N = \l_@@_except_corners_clist ,
568   except-corners .default:n = { NW , SW , NE , SE } ,
569   hvlines-except-corners .code:n =
570   {
571     \clist_set:Nn \l_@@_except_corners_clist { #1 }
572     \bool_set_true:N \l_@@_vlines_bool
573     \bool_set_true:N \l_@@_hlines_bool
574   } ,
575   hvlines-except-corners .default:n = { NW , SW , NE , SE } ,
576   code-before .code:n =
577   {
578     \tl_if_empty:nF { #1 }
579     {
580       \tl_put_right:Nn \l_@@_code_before_tl { #1 }
581       \bool_set_true:N \l_@@_code_before_bool
582     }
583   } ,

```

The options `c`, `t` and `b` of the environment `{NiceArray}` have the same meaning as the option of the classical environment `{array}`.

```

584   c .code:n = \tl_set:Nn \l_@@_baseline_tl c ,
585   t .code:n = \tl_set:Nn \l_@@_baseline_tl t ,
586   b .code:n = \tl_set:Nn \l_@@_baseline_tl b ,
587   baseline .tl_set:N = \l_@@_baseline_tl ,
588   baseline .value_required:n = true ,

```

```

589 columns-width .code:n =
590   \tl_if_eq:nnTF { #1 } { auto }
591   { \bool_set_true:N \l_@@_auto_columns_width_bool }
592   { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
593 columns-width .value_required:n = true ,
594 name .code:n =

```

We test whether we are in the measuring phase of an environment of `amsmath` (always loaded by `nicematrix`) because we want to avoid a fallacious message of duplicate name in this case.

```

595   \legacy_if:nF { measuring@ }
596   {
597     \str_set:Nn \l_tmpa_str { #1 }
598     \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
599     { \@@_error:nn { Duplicate-name } { #1 } }
600     { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
601     \str_set_eq:NN \l_@@_name_str \l_tmpa_str
602   } ,
603 name .value_required:n = true ,
604 code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
605 code-after .value_required:n = true ,
606 colortbl-like .code:n =
607   \bool_set_true:N \l_@@_colortbl_like_bool
608   \bool_set_true:N \l_@@_code_before_bool ,
609 colortbl-like .value_forbidden:n = true
610 }
611 \keys_define:nn { NiceMatrix / notes }
612 {
613   para .bool_set:N = \l_@@_notes_para_bool ,
614   para .default:n = true ,
615   code-before .tl_set:N = \l_@@_notes_code_before_tl ,
616   code-before .value_required:n = true ,
617   code-after .tl_set:N = \l_@@_notes_code_after_tl ,
618   code-after .value_required:n = true ,
619   bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
620   bottomrule .default:n = true ,
621   style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
622   style .value_required:n = true ,
623   label-in-tabular .code:n =
624     \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
625   label-in-tabular .value_required:n = true ,
626   label-in-list .code:n =
627     \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
628   label-in-list .value_required:n = true ,
629   enumitem-keys .code:n =
630     {
631       \bool_if:NTF \c_@@_in_preamble_bool
632       {
633         \AtBeginDocument
634         {
635           \bool_if:NT \c_@@_enumitem_loaded_bool
636           { \setlist* [ tabularnotes ] { #1 } }
637         }
638       }
639       {
640         \bool_if:NT \c_@@_enumitem_loaded_bool
641         { \setlist* [ tabularnotes ] { #1 } }
642       }
643     } ,
644   enumitem-keys .value_required:n = true ,
645   enumitem-keys-para .code:n =
646     {
647       \bool_if:NTF \c_@@_in_preamble_bool
648       {
649         \AtBeginDocument

```



```

650         {
651             \bool_if:NT \c_@@_enumitem_loaded_bool
652             { \setlist* [ tabularnotes* ] { #1 } }
653         }
654     }
655     {
656         \bool_if:NT \c_@@_enumitem_loaded_bool
657         { \setlist* [ tabularnotes* ] { #1 } }
658     }
659 },
660 enumitem-keys-para .value_required:n = true ,
661 unknown .code:n = \@@_error:n { Unknown~key~for~notes }
662 }

```

We begin the construction of the major sets of keys (used by the different user commands and environments).

```

663 \keys_define:nn { NiceMatrix }
664 {
665     NiceMatrixOptions .inherit:n =
666     { NiceMatrix / Global } ,
667     NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
668     NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
669     NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
670     NiceMatrixOptions / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
671     SubMatrix / rules .inherit:n = NiceMatrix / rules ,
672     CodeAfter / xdots .inherit:n = NiceMatrix / xdots ,
673     NiceMatrix .inherit:n =
674     {
675         NiceMatrix / Global ,
676         NiceMatrix / Env ,
677     } ,
678     NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
679     NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
680     NiceTabular .inherit:n =
681     {
682         NiceMatrix / Global ,
683         NiceMatrix / Env
684     } ,
685     NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
686     NiceTabular / rules .inherit:n = NiceMatrix / rules ,
687     NiceArray .inherit:n =
688     {
689         NiceMatrix / Global ,
690         NiceMatrix / Env ,
691     } ,
692     NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
693     NiceArray / rules .inherit:n = NiceMatrix / rules ,
694     pNiceArray .inherit:n =
695     {
696         NiceMatrix / Global ,
697         NiceMatrix / Env ,
698     } ,
699     pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
700     pNiceArray / rules .inherit:n = NiceMatrix / rules ,
701 }

```

We finalise the definition of the set of keys “NiceMatrix / NiceMatrixOptions” with the options specific to \NiceMatrixOptions.

```

702 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
703 {
704     delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
705     delimiters / color .value_required:n = true ,

```

```

706 delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
707 delimiters-color .value_required:n = true ,
708 last-col .code:n = \tl_if_empty:nF { #1 }
709 { \@@_error:n { last-col~non~empty~for~NiceMatrixOptions } }
710 \int_zero:N \l_@@_last_col_int ,
711 small .bool_set:N = \l_@@_small_bool ,
712 small .value_forbidden:n = true ,

```

With the option `renew-matrix`, the environment `{matrix}` of `amsmath` and its variants are redefined to behave like the environment `{NiceMatrix}` and its variants.

```

713 renew-matrix .code:n = \@@_renew_matrix: ,
714 renew-matrix .value_forbidden:n = true ,

```

The key `transparent` is now considered as obsolete (because its name is ambiguous).

```

715 transparent .code:n =
716 {
717   \@@_renew_matrix:
718   \bool_set_true:N \l_@@_renew_dots_bool
719   \@@_error:n { Key~transparent }
720 } ,
721 transparent .value_forbidden:n = true ,

```

The option `exterior-arraycolsep` will have effect only in `{NiceArray}` for those who want to have for `{NiceArray}` the same behaviour as `{array}`.

```

722 exterior-arraycolsep .bool_set:N = \l_@@_exterior_arraycolsep_bool ,

```

If the option `columns-width` is used, all the columns will have the same width.

In `\NiceMatrixOptions`, the special value `auto` is not available.

```

723 columns-width .code:n =
724   \tl_if_eq:nnTF { #1 } { auto }
725   { \@@_error:n { Option~auto~for~columns~width } }
726   { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

Usually, an error is raised when the user tries to give the same name to two distinct environments of `nicematrix` (theses names are global and not local to the current TeX scope). However, the option `allow-duplicate-names` disables this feature.

```

727 allow-duplicate-names .code:n =
728   \@@_msg_redirect_name:nn { Duplicate~name } { none } ,
729 allow-duplicate-names .value_forbidden:n = true ,

```

By default, the specifier used in the preamble of the array (for example in `{pNiceArray}`) to draw a vertical dotted line between two columns is the colon “:”. However, it’s possible to change this letter with `letter-for-dotted-lines` and, by the way, the letter “:” will remain free for other packages (for example `arydshln`).

```

730 letter-for-dotted-lines .code:n =
731 {
732   \tl_if_single_token:nTF { #1 }
733   { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
734   { \@@_error:n { One~letter~allowed } }
735 } ,
736 letter-for-dotted-lines .value_required:n = true ,
737 notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
738 notes .value_required:n = true ,
739 sub-matrix .code:n =
740   \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
741 sub-matrix .value_required:n = true ,
742 unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
743 }
744 \str_new:N \l_@@_letter_for_dotted_lines_str
745 \str_set_eq:NN \l_@@_letter_for_dotted_lines_str \c_colon_str

```

`\NiceMatrixOptions` is the command of the `nicematrix` package to fix options at the document level. The scope of these specifications is the current TeX group.

```

746 \NewDocumentCommand \NiceMatrixOptions { m }
747 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

We finalise the definition of the set of keys “NiceMatrix / NiceMatrix” with the options specific to `{NiceMatrix}`.

```

748 \keys_define:nn { NiceMatrix / NiceMatrix }
749 {
750   last-col .code:n = \tl_if_empty:nTF {#1}
751     {
752       \bool_set_true:N \l_@@_last_col_without_value_bool
753       \int_set:Nn \l_@@_last_col_int { -1 }
754     }
755     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
756   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
757   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
758   small .bool_set:N = \l_@@_small_bool ,
759   small .value_forbidden:n = true ,
760   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
761   delimiters / color .value_required:n = true ,
762   delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
763   delimiters-color .value_required:n = true ,
764   unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
765 }

```

We finalise the definition of the set of keys “NiceMatrix / NiceArray” with the options specific to `{NiceArray}`.

```

766 \keys_define:nn { NiceMatrix / NiceArray }
767 {

```

In the environments `{NiceArray}` and its variants, the option `last-col` must be used without value because the number of columns of the array is read from the preamble of the array.

```

768   small .bool_set:N = \l_@@_small_bool ,
769   small .value_forbidden:n = true ,
770   last-col .code:n = \tl_if_empty:nF { #1 }
771     { \@@_error:n { last-col-non-empty-for-NiceArray } }
772     \int_zero:N \l_@@_last_col_int ,
773   notes / para .bool_set:N = \l_@@_notes_para_bool ,
774   notes / para .default:n = true ,
775   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
776   notes / bottomrule .default:n = true ,
777   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
778   tabularnote .value_required:n = true ,
779   delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
780   delimiters-color .value_required:n = true ,
781   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
782   delimiters / color .value_required:n = true ,
783   unknown .code:n = \@@_error:n { Unknown-option-for-NiceArray }
784 }

785 \keys_define:nn { NiceMatrix / pNiceArray }
786 {
787   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
788   last-col .code:n = \tl_if_empty:nF {#1}
789     { \@@_error:n { last-col-non-empty-for-NiceArray } }
790     \int_zero:N \l_@@_last_col_int ,
791   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
792   small .bool_set:N = \l_@@_small_bool ,
793   small .value_forbidden:n = true ,
794   unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
795 }

```

We finalise the definition of the set of keys “NiceMatrix / NiceTabular” with the options specific to {NiceTabular}.

```

796 \keys_define:nn { NiceMatrix / NiceTabular }
797 {
798   notes / para .bool_set:N = \l_@@_notes_para_bool ,
799   notes / para .default:n = true ,
800   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
801   notes / bottomrule .default:n = true ,
802   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
803   tabularnote .value_required:n = true ,
804   last-col .code:n = \tl_if_empty:nF {#1}
805     { \@@_error:n { last-col~non-empty~for~NiceArray } }
806     \int_zero:N \l_@@_last_col_int ,
807   unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
808 }

```

## Important code used by {NiceArrayWithDelims}

The pseudo-environment `\@@_Cell:-\@@_end_Cell:` will be used to format the cells of the array. In the code, the affectations are global because this pseudo-environment will be used in the cells of a `\halign` (via an environment {array}).

```

809 \cs_new_protected:Npn \@@_Cell:
810 {

```

At the beginning of the cell, we link `\CodeAfter` to a command which do *not* begin with `\omit` (whereas the standard version of `\CodeAfter` begins with `\omit`).

```

811   \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n

```

We increment `\c@jCol`, which is the counter of the columns.

```

812   \int_gincr:N \c@jCol

```

Now, we increment the counter of the rows. We don’t do this incrementation in the `\everycr` because some packages, like `arydshln`, create special rows in the `\halign` that we don’t want to take into account.

```

813   \int_compare:nNnT \c@jCol = 1
814     { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
815   \int_gset:Nn \g_@@_col_total_int { \int_max:nn \g_@@_col_total_int \c@jCol }

```

The content of the cell is composed in the box `\l_@@_cell_box` because we want to compute some dimensions of the box. The `\hbox_set_end:` corresponding to this `\hbox_set:Nw` will be in the `\@@_end_Cell:` (and the potential `\c_math_toggle_token` also).

```

816   \hbox_set:Nw \l_@@_cell_box
817   \bool_if:NF \l_@@_NiceTabular_bool
818   {
819     \c_math_toggle_token
820     \bool_if:NT \l_@@_small_bool \scriptstyle
821   }

```

We will call *corners* of the matrix the cases which are at the intersection of the exterior rows and exterior columns (of course, the four corners doesn’t always exist simultaneously).

The codes `\l_@@_code_for_first_row_tl` and `al` don’t apply in the corners of the matrix.

```

822   \int_compare:nNnTF \c@iRow = 0
823   {
824     \int_compare:nNnT \c@jCol > 0
825     {
826       \l_@@_code_for_first_row_tl
827       \xglobal \colorlet { nicematrix-first-row } { . }
828     }
829   }
830   {
831     \int_compare:nNnT \c@iRow = \l_@@_last_row_int

```

```

832     {
833         \l_@@_code_for_last_row_tl
834         \xglobal \colorlet { nicematrix-last-row } { . }
835     }
836 }
837 }

```

The following macro `\@@_begin_of_row` is usually used in the cell number 1 of the row. However, when the key `first-col` is used, `\@@_begin_of_row` is executed in the cell number 0 of the row.

```

838 \cs_new_protected:Npn \@@_begin_of_row:
839 {
840     \int_gincr:N \c@iRow
841     \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
842     \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
843     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
844     \pgfpicture
845     \pgfrememberpicturepositiononpagetrue
846     \pgfcoordinate
847     { \@@_env: - row - \int_use:N \c@iRow - base }
848     { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
849     \str_if_empty:NF \l_@@_name_str
850     {
851         \pgfnodealias
852         { \l_@@_name_str - row - \int_use:N \c@iRow - base }
853         { \@@_env: - row - \int_use:N \c@iRow - base }
854     }
855     \endpgfpicture
856 }

```

The following code is used in each cell of the array. It actualises quantities that, at the end of the array, will give informations about the vertical dimension of the two first rows and the two last rows. If the user uses the `last-row`, some lines of code will be dynamically added to this command.

```

857 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
858 {
859     \int_compare:nNnTF \c@iRow = 0
860     {
861         \dim_gset:Nn \g_@@_dp_row_zero_dim
862         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
863         \dim_gset:Nn \g_@@_ht_row_zero_dim
864         { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
865     }
866     {
867         \int_compare:nNnT \c@iRow = 1
868         {
869             \dim_gset:Nn \g_@@_ht_row_one_dim
870             { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
871         }
872     }
873 }
874 \cs_new_protected:Npn \@@_rotate_cell_box:
875 {
876     \box_rotate:Nn \l_@@_cell_box { 90 }
877     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
878     {
879         \vbox_set_top:Nn \l_@@_cell_box
880         {
881             \vbox_to_zero:n { }
882             \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
883             \box_use:N \l_@@_cell_box
884         }
885     }
886 }

```

```

885     }
886     \bool_gset_false:N \g_@@_rotate_bool
887 }
888 \cs_new_protected:Npn \@@_adjust_size_box:
889 {
890     \dim_compare:nNnT \g_@@_blocks_wd_dim > \c_zero_dim
891     {
892         \box_set_wd:Nn \l_@@_cell_box
893         { \dim_max:nn { \box_wd:N \l_@@_cell_box } \g_@@_blocks_wd_dim }
894         \dim_gzero:N \g_@@_blocks_wd_dim
895     }
896     \dim_compare:nNnT \g_@@_blocks_dp_dim > \c_zero_dim
897     {
898         \box_set_dp:Nn \l_@@_cell_box
899         { \dim_max:nn { \box_dp:N \l_@@_cell_box } \g_@@_blocks_dp_dim }
900         \dim_gzero:N \g_@@_blocks_dp_dim
901     }
902     \dim_compare:nNnT \g_@@_blocks_ht_dim > \c_zero_dim
903     {
904         \box_set_ht:Nn \l_@@_cell_box
905         { \dim_max:nn { \box_ht:N \l_@@_cell_box } \g_@@_blocks_ht_dim }
906         \dim_gzero:N \g_@@_blocks_ht_dim
907     }
908 }
909 \cs_new_protected:Npn \@@_end_Cell:
910 {
911     \@@_math_toggle_token:
912     \hbox_set_end:
913     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
914     \@@_adjust_size_box:
915     \box_set_ht:Nn \l_@@_cell_box
916     { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
917     \box_set_dp:Nn \l_@@_cell_box
918     { \box_dp:N \l_@@_cell_box + \l_@@_cell_space_bottom_limit_dim }

```

We want to compute in `\g_@@_max_cell_width_dim` the width of the widest cell of the array (except the cells of the “first column” and the “last column”).

```

919     \dim_gset:Nn \g_@@_max_cell_width_dim
920     { \dim_max:nn \g_@@_max_cell_width_dim { \box_wd:N \l_@@_cell_box } }

```

The following computations are for the “first row” and the “last row”.

```

921     \@@_update_for_first_and_last_row:

```

If the cell is empty, or may be considered as if, we must not create the PGF node, for two reasons:

- it’s a waste of time since such a node would be rather pointless;
- we test the existence of these nodes in order to determine whether a cell is empty when we search the extremities of a dotted line.

However, it’s very difficult to determine whether a cell is empty. As for now, we use the following technic:

- if the width of the box `\l_@@_cell_box` (created with the content of the cell) is equal to zero, we consider the cell as empty (however, this is not perfect since the user may have used a `\rlap`, a `\llap` or a `\mathclap` of `mathtools`).
- the cells with a command `\Ldots` or `\Cdots`, `\Vdots`, etc., should also be considered as empty; if `nullify-dots` is in force, there would be nothing to do (in this case the previous commands only write an instruction in a kind of `\CodeAfter`); however, if `nullify-dots` is not in force, a phantom of `\ldots`, `\cdots`, `\vdots` is inserted and its width is not equal to zero; that’s why these commands raise a boolean `\g_@@_empty_cell_bool` and we begin by testing this boolean.

```

922 \bool_if:NTF \g_@@_empty_cell_bool
923   { \box_use_drop:N \l_@@_cell_box }
924   {
925     \bool_lazy_or:nnTF
926       \g_@@_not_empty_cell_bool
927       { \dim_compare_p:nNn { \box_wd:N \l_@@_cell_box } > \c_zero_dim }
928       \@@_node_for_the_cell:
929       { \box_use_drop:N \l_@@_cell_box }
930   }
931 \bool_gset_false:N \g_@@_empty_cell_bool
932 \bool_gset_false:N \g_@@_not_empty_cell_bool
933 }

```

The following command creates the PGF name of the node with, of course, `\l_@@_cell_box` as the content.

```

934 \cs_new_protected:Npn \@@_node_for_the_cell:
935 {
936   \pgfpicture
937   \pgfsetbaseline \c_zero_dim
938   \pgfrememberpicturepositiononpagetrue
939   \pgfset
940   {
941     inner~sep = \c_zero_dim ,
942     minimum~width = \c_zero_dim
943   }
944   \pgfnode
945   { rectangle }
946   { base }
947   { \box_use_drop:N \l_@@_cell_box }
948   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
949   { }
950   \str_if_empty:NF \l_@@_name_str
951   {
952     \pgfnodealias
953     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
954     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
955   }
956   \endpgfpicture
957 }

```

The second argument of the following command `\@@_instruction_of_type:nnn` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). The third argument is the list of options. This command writes in the corresponding `\g_@@_type_lines_tl` the instruction which will actually draw the line after the construction of the matrix.

For example, for the following matrix,

```

\begin{pNiceMatrix}
1 & 2 & 3 & 4 \\
5 & \Cdots & & 6 \\
7 & \Cdots[color=red]
\end{pNiceMatrix}

```

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & \cdots & & 6 \\ 7 & \cdots & & \end{pmatrix}$$

the content of `\g_@@_Cdots_lines_tl` will be:

```

\@@_draw_Cdots:nnn {2}{2}{}
\@@_draw_Cdots:nnn {3}{2}{color=red}

```

The first argument is a boolean which indicates whether you must put the instruction on the left or on the right on the list of instructions.

```

958 \cs_new_protected:Npn \@@_instruction_of_type:nnn #1 #2 #3
959 {
960   \bool_if:NTF { #1 } \tl_gput_left:cx \tl_gput_right:cx
961   { g_@@_ #2 _ lines _ tl }

```

```

962 {
963   \use:c { @@ _ draw _ #2 : nnn }
964   { \int_use:N \c@iRow }
965   { \int_use:N \c@jCol }
966   { \exp_not:n { #3 } }
967 }
968 }

```

We want to use `\array` of `array`. However, if the class used is `revtex4-1` or `revtex4-2`, we have to do some tuning and use the command `\@array@array` instead of `\array` because these classes do a redefinition of `\array` incompatible with our use of `\array`.

```

969 \cs_new_protected:Npn \@@_revtex_array:
970 {
971   \cs_set_eq:NN \@acoll \@arrayacol
972   \cs_set_eq:NN \@acolr \@arrayacol
973   \cs_set_eq:NN \@acol \@arrayacol
974   \cs_set_nopar:Npn \@halignto { }
975   \@array@array
976 }
977 \cs_new_protected:Npn \@@_array:
978 {
979   \bool_if:NTF \c_@@_revtex_bool
980     \@@_revtex_array:
981   {
982     \bool_if:NTF \l_@@_NiceTabular_bool
983       { \dim_set_eq:NN \col@sep \tabcolsep }
984       { \dim_set_eq:NN \col@sep \arraycolsep }
985     \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim
986       { \cs_set_nopar:Npn \@halignto { } }
987       { \cs_set_nopar:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }

```

If `colortbl` is loaded, `\@tabarray` has been redefined to incorporate `\CT@start`.

```

988   \@tabarray
989 }

```

`\l_@@_baseline_tl` may have the value `t`, `c` or `b`. However, if the value is `b`, we compose the `\array` (of `array`) with the option `t` and the right translation will be done further. Remark that `\str_if_eq:VnTF` is fully expandable and you need something fully expandable here.

```

990   [ \str_if_eq:VnTF \l_@@_baseline_tl c c t ]
991 }

```

We keep in memory the standard version of `\ialign` because we will redefine `\ialign` in the environment `{NiceArrayWithDelims}` but restore the standard version for use in the cells of the array.

```

992 \cs_set_eq:NN \@@_old_ialign: \ialign

```

The following command creates a `row` node (and not a row of nodes!).

```

993 \cs_new_protected:Npn \@@_create_row_node:
994 {

```

The `\hbox:n` (or `\hbox`) is mandatory.

```

995   \hbox
996   {
997     \bool_if:NT \l_@@_code_before_bool
998     {
999       \vtop
1000       {
1001         \skip_vertical:N 0.5\arrayrulewidth
1002         \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
1003         \skip_vertical:N -0.5\arrayrulewidth
1004       }
1005     }
1006     \pgfpicture
1007     \pgfrememberpicturepositiononpagetrue

```



```

1008     \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
1009     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
1010     \str_if_empty:NF \l_@@_name_str
1011     {
1012         \pgfnodealias
1013         { \l_@@_name_str - row - \int_use:N \c@iRow }
1014         { \@@_env: - row - \int_use:N \c@iRow }
1015     }
1016     \endpgfpicture
1017 }
1018 }

```

The following must *not* be protected because it begins with `\noalign`.

```

1019 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
1020 \cs_new_protected:Npn \@@_everycr_i:
1021 {
1022     \int_gzero:N \c@jCol
1023     \bool_gset_false:N \g_@@_after_col_zero_bool
1024     \bool_if:NF \g_@@_row_of_col_done_bool
1025     {
1026         \@@_create_row_node:

```

We don't draw the rules of the key `hlines` (or `hvlines`) but we reserve the vertical space for theses rules.

```

1027         \bool_if:NT \l_@@_hlines_bool
1028         {

```

The counter `\c@iRow` has the value `-1` only if there is a “first row” and that we are before that “first row”, i.e. just before the beginning of the array.

```

1029         \int_compare:nNnT \c@iRow > { -1 }
1030         {
1031             \int_compare:nNnF \c@iRow = \l_@@_last_row_int

```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded. We use a TeX group in order to limit the scope of `\CT@arc@`.

```

1032             { \hrule height \arrayrulewidth width \c_zero_dim }
1033         }
1034     }
1035 }
1036 }

```

The command `\@@_newcolumntype` is the command `\newcolumntype` of `array` without the warnings for redefinitions of columns types (we will use it to redefine the columns types `w` and `W`).

```

1037 \cs_set_protected:Npn \@@_newcolumntype #1
1038 {
1039     \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
1040     \peek_meaning:NTF [
1041         { \newcol@ #1 }
1042         { \newcol@ #1 [ 0 ] }
1043     }

```

When the key `renew-dots` is used, the following code will be executed.

```

1044 \cs_set_protected:Npn \@@_renew_dots:
1045 {
1046     \cs_set_eq:NN \ldots \@@_Ldots
1047     \cs_set_eq:NN \cdots \@@_Cdots
1048     \cs_set_eq:NN \vdots \@@_Vdots
1049     \cs_set_eq:NN \ddots \@@_Ddots
1050     \cs_set_eq:NN \iddots \@@_Iddots
1051     \cs_set_eq:NN \dots \@@_Ldots
1052     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
1053 }

```

When the key `colortbl-like` is used, the following code will be executed.

```

1054 \cs_new_protected:Npn \@@_colortbl_like:
1055 {
1056   \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
1057   \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
1058   \cs_set_eq:NN \columncolor \@@_columncolor_preamble
1059 }

```

The following code `\@@_pre_array:` is used in `{NiceArrayWithDelims}`. It exists as a standalone macro only for legibility.

```

1060 \cs_new_protected:Npn \@@_pre_array:
1061 {

```

If `booktabs` is loaded, we have to patch the macro `\@BTnormal` which is a macro of `booktabs`. The macro `\@BTnormal` draws an horizontal rule but it occurs after a vertical skip done by a low level TeX command. When this macro `\@BTnormal` occurs, the `row` node has yet been inserted by `nicematrix` *before* the vertical skip (and thus, at a wrong place). That why we decide to create a new `row` node (for the same row). We patch the macro `\@BTnormal` to create this `row` node. This new `row` node will overwrite the previous definition of that `row` node and we have managed to avoid the error messages of that redefinition<sup>45</sup>.

```

1062   \bool_if:NT \c_@@_booktabs_loaded_bool
1063   { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
1064   \box_clear_new:N \l_@@_cell_box
1065   \cs_if_exist:NT \theiRow
1066   { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
1067   \int_gzero_new:N \c@iRow
1068   \cs_if_exist:NT \thejCol
1069   { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
1070   \int_gzero_new:N \c@jCol
1071   \normalbaselines

```

If the option `small` is used, we have to do some tuning. In particular, we change the value of `\arraystretch` (this parameter is used in the construction of `\@arstrutbox` in the beginning of `{array}`).

```

1072   \bool_if:NT \l_@@_small_bool
1073   {
1074     \cs_set_nopar:Npn \arraystretch { 0.47 }
1075     \dim_set:Nn \arraycolsep { 1.45 pt }
1076   }

```

The environment `{array}` uses internally the command `\ialign`. We change the definition of `\ialign` for several reasons. In particular, `\ialign` sets `\everycr` to `{ }` and we *need* to have to change the value of `\everycr`.

```

1077   \cs_set_nopar:Npn \ialign
1078   {
1079     \bool_if:NTF \c_@@_colortbl_loaded_bool
1080     {
1081       \CT@everycr
1082       {
1083         \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
1084         \@@_everycr:
1085       }
1086     }
1087     { \everycr { \@@_everycr: } }
1088     \tabskip = \c_zero_skip

```

The box `\@arstrutbox` is a box constructed in the beginning of the environment `{array}`. The construction of that box takes into account the current value of `\arraystretch`<sup>46</sup> and `\extrarowheight`

<sup>45</sup>cf. `\nicematrix@redefine@check@rerun`

<sup>46</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

(of array). That box is inserted (via `\@arstrut`) in the beginning of each row of the array. That's why we use the dimensions of that box to initialize the variables which will be the dimensions of the potential first and last row of the environment. This initialization must be done after the creation of `\@arstrutbox` and that's why we do it in the `\ialign`.

```

1089 \dim_gzero_new:N \g_@@_dp_row_zero_dim
1090 \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
1091 \dim_gzero_new:N \g_@@_ht_row_zero_dim
1092 \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
1093 \dim_gzero_new:N \g_@@_ht_row_one_dim
1094 \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
1095 \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
1096 \dim_gzero_new:N \g_@@_ht_last_row_dim
1097 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
1098 \dim_gzero_new:N \g_@@_dp_last_row_dim
1099 \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

After its first use, the definition of `\ialign` will revert automatically to its default definition. With this programming, we will have, in the cells of the array, a clean version of `\ialign`.

```

1100 \cs_set_eq:NN \ialign \@_old_ialign:
1101 \halign
1102 }

```

We keep in memory the old versions of `\ldots`, `\cdots`, etc. only because we use them inside `\phantom` commands in order that the new commands `\Ldots`, `\Cdots`, etc. give the same spacing (except when the option `nullify-dots` is used).

```

1103 \cs_set_eq:NN \@_old_ldots \ldots
1104 \cs_set_eq:NN \@_old_cdots \cdots
1105 \cs_set_eq:NN \@_old_vdots \vdots
1106 \cs_set_eq:NN \@_old_ddots \ddots
1107 \cs_set_eq:NN \@_old_iddots \iddots
1108 \bool_if:NTF \l_@@_standard_cline_bool
1109 { \cs_set_eq:NN \cline \@_standard_cline }
1110 { \cs_set_eq:NN \cline \@_cline }
1111 \cs_set_eq:NN \Ldots \@_Ldots
1112 \cs_set_eq:NN \Cdots \@_Cdots
1113 \cs_set_eq:NN \Vdots \@_Vdots
1114 \cs_set_eq:NN \Ddots \@_Ddots
1115 \cs_set_eq:NN \Iddots \@_Iddots
1116 \cs_set_eq:NN \hdottedline \@_hdottedline:
1117 \cs_set_eq:NN \Hline \@_Hline:
1118 \cs_set_eq:NN \Hspace \@_Hspace:
1119 \cs_set_eq:NN \Hdotsfor \@_Hdotsfor:
1120 \cs_set_eq:NN \Vdotsfor \@_Vdotsfor:
1121 \cs_set_eq:NN \multicolumn \@_multicolumn:n
1122 \cs_set_eq:NN \Block \@_Block:
1123 \cs_set_eq:NN \rotate \@_rotate:
1124 \cs_set_eq:NN \OnlyMainNiceMatrix \@_OnlyMainNiceMatrix:n
1125 \cs_set_eq:NN \dotfill \@_old_dotfill:
1126 \cs_set_eq:NN \CodeAfter \@_CodeAfter:
1127 \cs_set_eq:NN \diagbox \@_diagbox:nn
1128 \cs_set_eq:NN \NotEmpty \@_NotEmpty:
1129 \bool_if:NT \l_@@_colortbl_like_bool \@_colortbl_like:
1130 \bool_if:NT \l_@@_renew_dots_bool \@_renew_dots:

```

The sequence `\g_@@_multicolumn_cells_seq` will contain the list of the cells of the array where a command `\multicolumn{n}{...}{...}` with  $n > 1$  is issued. In `\g_@@_multicolumn_sizes_seq`, the “sizes” (that is to say the values of  $n$ ) correspondant will be stored. These lists will be used for the creation of the “medium nodes” (if they are created).

```

1131 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
1132 \seq_gclear_new:N \g_@@_multicolumn_sizes_seq

```

The counter `\c@iRow` will be used to count the rows of the array (its incrementation will be in the first cell of the row).

```

1133 \int_gset:Nn \c@iRow { \l_@@_first_row_int - 1 }

```

At the end of the environment `{array}`, `\c@iRow` will be the total number de rows.  
`\g_@@_row_total_int` will be the number or rows excepted the last row (if `\l_@@_last_row_bool` has been raised with the option `last-row`).

```
1134 \int_gzero_new:N \g_@@_row_total_int
```

The counter `\c@jCol` will be used to count the columns of the array. Since we want to know the total number of columns of the matrix, we also create a counter `\g_@@_col_total_int`. These counters are updated in the command `\@@_Cell:` executed at the beginning of each cell.

```
1135 \int_gzero_new:N \g_@@_col_total_int
1136 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1137 \@@_renew_NC@rewrite@S:
1138 \bool_gset_false:N \g_@@_last_col_found_bool
```

During the construction of the array, the instructions `\Cdots`, `\Ldots`, etc. will be written in token lists `\g_@@_Cdots_lines_tl`, etc. which will be executed after the construction of the array.

```
1139 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1140 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1141 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1142 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1143 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1144 \tl_gclear_new:N \g_@@_HVdotsfor_lines_tl

1145 \tl_gclear_new:N \g_nicematrix_code_before_tl
1146 }
```

This is the end of `\@@_pre_array:`.

## The environment `{NiceArrayWithDelims}`

```
1147 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
1148 {
1149 \@@_provide_pgfsyspdfmark:
1150 \bool_if:NT \c_@@_footnote_bool \savenotes
```

The aim of the following `\bgroup` (the corresponding `\egroup` is, of course, at the end of the environment) is to be able to put an exposant to a matrix in a mathematical formula.

```
1151 \bgroup

1152 \tl_set:Nn \l_@@_left_delim_tl { #1 }
1153 \tl_set:Nn \l_@@_right_delim_tl { #2 }
1154 \tl_gset:Nn \g_@@_preamble_tl { #4 }

1155 \int_gzero:N \g_@@_block_box_int
1156 \dim_zero:N \g_@@_width_last_col_dim
1157 \dim_zero:N \g_@@_width_first_col_dim
1158 \bool_gset_false:N \g_@@_row_of_col_done_bool
1159 \str_if_empty:NT \g_@@_name_env_str
1160 { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1161 \@@_adapt_S_column:
1162 \bool_if:NTF \l_@@_NiceTabular_bool
1163 \mode_leave_vertical:
1164 \@@_test_if_math_mode:
1165 \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
1166 \bool_set_true:N \l_@@_in_env_bool
```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>47</sup>. This command is used by `\CT@arc@` but we use it also for compatibility with `colortbl`. But we want also to be able to use color for the rules of the array when `colortbl` is *not* loaded. That's why we do the following

---

<sup>47</sup>e.g. `\color[rgb]{0.5,0.5,0}`

instruction which is in the patch of the beginning of arrays done by `colortbl`. Of course, we restore the value of `\CT@arc@` at the end of our environment.

```
1167 \cs_gset_eq:NN \@@_old_CT@arc@ \CT@arc@
```

We deactivate Tikz externalization because we will use PGF pictures with the options `overlay` and `remember picture` (or equivalent forms). We deactivate with `\tikzexternaldisable` and not with `\tikzset{external/export=false}` which is *not* equivalent.

```
1168 \cs_if_exist:NT \tikz@library@external@loaded
1169 {
1170   \tikzexternaldisable
1171   \cs_if_exist:NT \ifstandalone
1172   { \tikzset { external / optimize = false } }
1173 }
```

We increment the counter `\g_@@_env_int` which counts the environments of the package.

```
1174 \int_gincr:N \g_@@_env_int
1175 \bool_if:NF \l_@@_block_auto_columns_width_bool
1176 { \dim_gzero_new:N \g_@@_max_cell_width_dim }
```

The sequence `\g_@@_blocks_seq` will contain the characteristics of the blocks (specified by `\Block`) of the array. The sequence `\g_@@_pos_of_blocks_seq` will contain only the position of the blocks. Of course, this is redundant but it's for efficiency.

```
1177 \seq_gclear:N \g_@@_blocks_seq
1178 \seq_gclear:N \g_@@_pos_of_blocks_seq
```

In fact, the sequence `\g_@@_pos_of_blocks_seq` will also contain the positions of the cells with a `\diagbox`.

```
1179 \seq_gclear:N \g_@@_pos_of_stroken_blocks_seq
1180 \seq_gclear:N \g_@@_pos_of_xdots_seq

1181 \tl_if_exist:cT { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1182 {
1183   \bool_set_true:N \l_@@_code_before_bool
1184   \exp_args:NNv \tl_put_right:Nn \l_@@_code_before_tl
1185   { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1186 }
```

The set of keys is not exactly the same for `{NiceArray}` and for the variants of `{NiceArray}` (`{pNiceArray}`, `{bNiceArray}`, etc.) because, for `{NiceArray}`, we have the options `t`, `c`, `b` and `baseline`.

```
1187 \bool_if:NTF \l_@@_NiceArray_bool
1188 { \keys_set:nn { NiceMatrix / NiceArray } }
1189 { \keys_set:nn { NiceMatrix / pNiceArray } }
1190 { #3 , #5 }

1191 \tl_if_empty:NF \l_@@_rules_color_tl
1192 { \exp_after:wN \@@_set_CT@arc@: \l_@@_rules_color_tl \q_stop }
```

If the key `code-before` is used, we have to create the `col` nodes and the `row` nodes before the creation of the array. First, we have to test whether the size of the array has been written in the `aux` file in a previous run. In this case, a command `\@@_size_nb_of_env:` has been created.

```
1193 \bool_if:NT \l_@@_code_before_bool
1194 {
1195   \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1196   {
```

First, we give values to the LaTeX counters `iRow` and `jCol`. We remind that, in the `code-before` (and in the `\CodeAfter`) they represent the numbers of rows and columns of the array (without the potential last row and last column).

```
1197   \int_zero_new:N \c@iRow
1198   \int_set:Nn \c@iRow
1199   { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1200   \int_zero_new:N \c@jCol
1201   \int_set:Nn \c@jCol
1202   { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 }
```

We have to adjust the values of `\c@iRow` and `\c@jCol` to take into account the potential last row and last column. A value of `-2` for `\l_@@_last_row_int` means that there is no last row. Idem for the columns.

```

1203 \int_compare:nNnF \l_@@_last_row_int = { -2 }
1204 { \int_decr:N \c@iRow }
1205 \int_compare:nNnF \l_@@_last_col_int = { -2 }
1206 { \int_decr:N \c@jCol }

```

Now, we will create all the `col` nodes and `row` nodes with the informations written in the `aux` file. You use the technique described in the page 1229 of `pgfmanual.pdf`, version 3.1.4b.

```

1207 \pgfsys@markposition { \@@_env: - position }
1208 \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1209 \pgfpicture

```

First, the creation of the `row` nodes.

```

1210 \int_step_inline:nnn
1211 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1212 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1213 {
1214 \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1215 \pgfcoordinate { \@@_env: - row - ##1 }
1216 { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1217 }

```

Now, the creation of the `col` nodes.

```

1218 \int_step_inline:nnn
1219 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1220 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1221 {
1222 \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1223 \pgfcoordinate { \@@_env: - col - ##1 }
1224 { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1225 }
1226 \endpgfpicture

```

Now, you create the diagonal nodes by using the `row` nodes and the `col` nodes.

```

1227 \@@_create_diag_nodes:

```

Now, yet another settings before the execution of the `code-before`.

```

1228 \group_begin:
1229 \bool_if:NT \c_@@_tikz_loaded_bool
1230 {
1231 \tikzset
1232 {
1233 every~picture / .style =
1234 { overlay , name~prefix = \@@_env: - }
1235 }
1236 }
1237 \cs_set_eq:NN \cellcolor \@@_cellcolor
1238 \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
1239 \cs_set_eq:NN \rowcolor \@@_rowcolor
1240 \cs_set_eq:NN \rowcolors \@@_rowcolors
1241 \cs_set_eq:NN \columncolor \@@_columncolor
1242 \cs_set_eq:NN \chessboardcolors \@@_chessboardcolors

```

We compose the `code-before` in math mode in order to nullify the spaces put by the user between instructions in the `code-before`.

```

1243 \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1244 \seq_clear_new:N \l_@@_colors_seq
1245 \l_@@_code_before_tl
1246 \@@_actually_color:
1247 \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1248 \group_end:
1249 }
1250 }

```

A value of  $-1$  for the counter `\l_@@_last_row_int` means that the user has used the option `last-row` without value, that is to say without specifying the number of that last row. In this case, we try to read that value from the aux file (if it has been written on a previous run).

```

1251 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1252 {
1253   \tl_put_right:Nn \@@_update_for_first_and_last_row:
1254   {
1255     \dim_gset:Nn \g_@@_ht_last_row_dim
1256     { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1257     \dim_gset:Nn \g_@@_dp_last_row_dim
1258     { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1259   }
1260 }
1261 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1262 {
1263   \bool_set_true:N \l_@@_last_row_without_value_bool

```

A value based on the name is more reliable than a value based on the number of the environment.

```

1264 \str_if_empty:NTF \l_@@_name_str
1265 {
1266   \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1267   {
1268     \int_set:Nn \l_@@_last_row_int
1269     { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1270   }
1271 }
1272 {
1273   \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1274   {
1275     \int_set:Nn \l_@@_last_row_int
1276     { \use:c { @@_last_row_ \l_@@_name_str } }
1277   }
1278 }
1279 }

```

A value of  $-1$  for the counter `\l_@@_last_col_int` means that the user has used the option `last-col` without value, that is to say without specifying the number of that last column. In this case, we try to read that value from the aux file (if it has been written on a previous run).

```

1280 \int_compare:nNnT \l_@@_last_col_int = { -1 }
1281 {
1282   \str_if_empty:NTF \l_@@_name_str
1283   {
1284     \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1285     {
1286       \int_set:Nn \l_@@_last_col_int
1287       { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1288     }
1289   }
1290   {
1291     \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1292     {
1293       \int_set:Nn \l_@@_last_col_int
1294       { \use:c { @@_last_col_ \l_@@_name_str } }
1295     }
1296   }
1297 }

```

The code in `\@@_pre_array:` is used only by `{NiceArrayWithDelims}`.

```

1298 \@@_pre_array:

```

We compute the width of both delimiters.

```

1299 \dim_zero_new:N \l_@@_left_delim_dim
1300 \dim_zero_new:N \l_@@_right_delim_dim

```

```

1301 \bool_if:NTF \l_@@_NiceArray_bool
1302 {
1303   \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1304   \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1305 }
1306 {

```

The command `\bBigg@` is a command of `amsmath`.

```

1307   \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \l_@@_left_delim_tl $ }
1308   \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1309   \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \l_@@_right_delim_tl $ }
1310   \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1311 }

```

The array will be composed in a box (named `\l_@@_the_array_box`) because we have to do manipulations concerning the potential exterior rows.

```

1312 \box_clear_new:N \l_@@_the_array_box

```

If the user has loaded `nicematrix` with the option `define-L-C-R`, he will be able to use `L`, `C` and `R` instead of `l`, `c` and `r` in the preambles of the environments of `nicematrix` (it's a compatibility mode since `L`, `C` and `R` were mandatory before version 5.0).

```

1313 \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:

```

The preamble will be constructed in `\g_@@_preamble_tl`.

```

1314 \@@_construct_preamble:

```

Now, the preamble is constructed in `\g_@@_preamble_tl`

Here is the beginning of the box which will contain the array. The `\hbox_set_end:` corresponding to this `\hbox_set:Nw` will be in the second part of the environment (and the closing `\c_math_toggle_token` also).

```

1315 \hbox_set:Nw \l_@@_the_array_box
1316 \skip_horizontal:N \l_@@_left_margin_dim
1317 \skip_horizontal:N \l_@@_extra_left_margin_dim
1318 \c_math_toggle_token
1319 \bool_if:NTF \l_@@_light_syntax_bool
1320 { \use:c { @@-light-syntax } }
1321 { \use:c { @@-normal-syntax } }
1322 }
1323 {
1324   \bool_if:NTF \l_@@_light_syntax_bool
1325   { \use:c { end @@-light-syntax } }
1326   { \use:c { end @@-normal-syntax } }
1327   \c_math_toggle_token
1328   \skip_horizontal:N \l_@@_right_margin_dim
1329   \skip_horizontal:N \l_@@_extra_right_margin_dim
1330   \hbox_set_end:

```

End of the construction of the array (in the box `\l_@@_the_array_box`).

If the user has used the key `last-row` with a value, we control that the given value is correct (since we have just constructed the array, we know the real number of rows of the array).

```

1331 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1332 {
1333   \bool_if:NF \l_@@_last_row_without_value_bool
1334   {
1335     \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1336     {
1337       \@@_error:n { Wrong~last~row }
1338       \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1339     }
1340   }
1341 }

```



Now, the definition of `\c@jCol` and `\g_@@_col_total_int` change: `\c@jCol` will be the number of columns without the “last column”; `\g_@@_col_total_int` will be the number of columns with this “last column”.<sup>48</sup>

```

1342 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1343 \bool_if:nTF \g_@@_last_col_found_bool
1344 { \int_gdecr:N \c@jCol }
1345 {
1346   \int_compare:nNnT \l_@@_last_col_int > { -1 }
1347   { \@@_error:n { last~col~not~used } }
1348 }

```

We fix also the value of `\c@iRow` and `\g_@@_row_total_int` with the same principle.

```

1349 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1350 \int_compare:nNnT \l_@@_last_row_int > { -1 } { \int_gdecr:N \c@iRow }

```

Now, we begin the real construction in the output flow of TeX. First, we take into account a potential “first column” (we remind that this “first column” has been constructed in an overlapping position and that we have computed its width in `\g_@@_width_first_col_dim`: see p. 97).

```

1351 \int_compare:nNnT \l_@@_first_col_int = 0
1352 {
1353   \skip_horizontal:N \col@sep
1354   \skip_horizontal:N \g_@@_width_first_col_dim
1355 }

```

The construction of the real box is different when `\l_@@_NiceArray_bool` is true (`{NiceArray}` or `{NiceTabular}`) and in the other environments because, in `{NiceArray}` or `{NiceTabular}`, we have no delimiter to put (but we have tabular notes to put). We begin with this case.

Remark that, in all cases, `@@_use_arraybox_with_notes_c:` is used.

```

1356 \bool_if:nTF \l_@@_NiceArray_bool
1357 {
1358   \str_case:VnF \l_@@_baseline_tl
1359   {
1360     b \@@_use_arraybox_with_notes_b:
1361     c \@@_use_arraybox_with_notes_c:
1362   }
1363   \@@_use_arraybox_with_notes:
1364 }

```

Now, in the case of an environment `{pNiceArray}`, `{bNiceArray}`, etc. We compute `\l_tmpa_dim` which is the total height of the “first row” above the array (when the key `first-row` is used).

```

1365 {
1366   \int_compare:nNnTF \l_@@_first_row_int = 0
1367   {
1368     \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1369     \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1370   }
1371   { \dim_zero:N \l_tmpa_dim }

```

We compute `\l_tmpb_dim` which is the total height of the “last row” below the array (when the key `last-row` is used). A value of `-2` for `\l_@@_last_row_int` means that there is no “last row”.<sup>49</sup>

```

1372 \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1373 {
1374   \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1375   \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1376 }
1377 { \dim_zero:N \l_tmpb_dim }
1378 \hbox_set:Nn \l_tmpa_box
1379 {
1380   \c_math_toggle_token
1381   \tl_if_empty:NF \l_@@_delimiters_color_tl
1382   { \color { \l_@@_delimiters_color_tl } }

```

<sup>48</sup>We remind that the potential “first column” (exterior) has the number 0.

<sup>49</sup>A value of `-1` for `\l_@@_last_row_int` means that there is a “last row” but the the user have not set the value with the option `last row` (and we are in the first compilation).

```

1383         \exp_after:wN \left \l_@@_left_delim_tl
1384         \vcenter
1385         {

```

We take into account the “first row” (we have previously computed its total height in `\l_tmpa_dim`). The `\hbox:n` (or `\hbox`) is necessary here.

```

1386         \skip_vertical:N -\l_tmpa_dim
1387         \hbox
1388         {
1389             \bool_if:NTF \l_@@_NiceTabular_bool
1390             { \skip_horizontal:N -\tabcolsep }
1391             { \skip_horizontal:N -\arraycolsep }
1392             \@@_use_arraybox_with_notes_c:
1393             \bool_if:NTF \l_@@_NiceTabular_bool
1394             { \skip_horizontal:N -\tabcolsep }
1395             { \skip_horizontal:N -\arraycolsep }
1396         }

```

We take into account the “last row” (we have previously computed its total height in `\l_tmpb_dim`).

```

1397         \skip_vertical:N -\l_tmpb_dim
1398     }
1399     \exp_after:wN \right \l_@@_right_delim_tl
1400     \c_math_toggle_token
1401 }

```

Now, the box `\l_tmpa_box` is created with the correct delimiters.

We will put the box in the TeX flow. However, we have a small work to do when the option `max-delimiter-width` is used.

```

1402     \bool_if:NTF \l_@@_max_delimiter_width_bool
1403     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1404     \@@_put_box_in_flow:
1405 }

```

We take into account a potential “last column” (this “last column” has been constructed in an overlapping position and we have computed its width in `\g_@@_width_last_col_dim`: see p. 98).

```

1406     \bool_if:NT \g_@@_last_col_found_bool
1407     {
1408         \skip_horizontal:N \g_@@_width_last_col_dim
1409         \skip_horizontal:N \col@sep
1410     }
1411     \bool_if:NF \l_@@_Matrix_bool
1412     {
1413         \int_compare:nNt \c@jCol < \g_@@_static_num_of_col_int
1414         { \@@_error:n { columns-not-used } }
1415     }
1416     \group_begin:
1417     \globaldefs = 1
1418     \@@_msg_redirect_name:nn { columns-not-used } { error }
1419     \group_end:
1420     \@@_after_array:

```

The aim of the following `\egroup` (the corresponding `\bgroup` is, of course, at the beginning of the environment) is to be able to put an exposant to a matrix in a mathematical formula.

```

1421     \egroup
1422     \bool_if:NT \c_@@_footnote_bool \endsavenotes
1423 }

```

This is the end of the environment `{NiceArrayWithDelims}`.

## We construct the preamble of the array

The transformation of the preamble is an operation in several steps.

The preamble given by the final use is in `\g_@@_preamble_tl` and the modified version will be stored in `\g_@@_preamble_tl` also.

```

1424 \cs_new_protected:Npn \@@_construct_preamble:
1425 {

```

First, we will do an “expansion” of the preamble with the tools of the package `array` itself. This “expansion” will expand all the constructions with `*` and with all column types (defined by the user or by various packages using `\newcolumntype`).

Since we use the tools of `array` to do this expansion, we will have a programming which is not in the style of `expl3`.

We redefine the column types `w` and `W`. We use `\@@_newcolumntype` instead of `\newcolumntype` because we don’t want warnings for column types already defined. These redefinitions are in fact *protections* of the letters `w` and `W`. We don’t want these columns type expanded because we will do the patch ourselves after. We want to be able the standard column types `w` and `W` in potential `{tabular}` of `array` in some cells of our array. That’s why we do those redefinitions in a TeX group.

```

1426 \group_begin:

```

If we are in an environment without explicit preamble, we have nothing to do (excepted the treatment on both sides of the preamble which will be done at the end).

```

1427 \bool_if:NF \l_@@_Matrix_bool
1428 {
1429   \@@_newcolumntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1430   \@@_newcolumntype W [ 2 ] { \@@_W: { ##1 } { ##2 } }

```

First, we have to store our preamble in the token register `\@temptokena` (those “token registers” are not supported by `expl3`).

```

1431 \exp_args:NV \@temptokena \g_@@_preamble_tl

```

Initialisation of a flag used by `array` to detect the end of the expansion.

```

1432 \@tempswatrue

```

The following line actually does the expansion (it’s has been copied from `array.sty`).

```

1433 \@whilesw \if@tempswa \fi { \@tempswafalse \the \NC@list }

```

Now, we have to “patch” that preamble by transforming some columns. We will insert in the TeX flow the preamble in its actual form (that is to say after the “expansion”) following by a marker `\q_stop` and we will consume these tokens constructing the (new form of the) preamble in `\g_@@_preamble_tl`. This is done recursively with the command `\@@_patch_preamble:n`. In the same time, we will count the columns with the counter `\c@jCol`.

```

1434 \int_gzero_new:N \c@jCol
1435 \bool_if:NTF \l_@@_vlines_bool
1436 {
1437   \tl_gset:Nn \g_@@_preamble_tl
1438     { ! { \skip_horizontal:N \arrayrulewidth } }
1439 }
1440 { \tl_gclear:N \g_@@_preamble_tl }

```

The sequence `\g_@@_cols_vlsim_seq` will contain the numbers of the columns where you will to have to draw vertical lines in the potential sub-matrices (hence the name `vlsim`).

```

1441 \seq_clear:N \g_@@_cols_vlism_seq

```

The counter `\l_tmpa_int` will count the number of consecutive occurrences of the symbol `|`.

```

1442 \int_zero:N \l_tmpa_int

```

Now, we actually patch the preamble (and it is constructed in `\g_@@_preamble_tl`).

```

1443 \exp_after:wN \@@_patch_preamble:n \the \@temptokena \q_stop
1444 \int_gset_eq:NN \g_@@_static_num_of_col_int \c@jCol
1445 }

```

Now, we replace `\columncolor` by `\@@_columncolor_preamble`.

```

1446 \bool_if:NT \l_@@_colortbl_like_bool
1447 {
1448   \regex_replace_all:NnN
1449     \c_@@_columncolor_regex
1450     { \c { @@_columncolor_preamble } }
1451     \g_@@_preamble_tl
1452 }

```

We complete the preamble with the potential “exterior columns”.

```

1453 \int_compare:nNnTF \l_@@_first_col_int = 0
1454 { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1455 {
1456   \bool_lazy_all:nT
1457   {
1458     \l_@@_NiceArray_bool
1459     { \bool_not_p:n \l_@@_NiceTabular_bool }
1460     { \bool_not_p:n \l_@@_vlines_bool }
1461     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1462   }
1463   { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1464 }
1465 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1466 { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }
1467 {
1468   \bool_lazy_all:nT
1469   {
1470     \l_@@_NiceArray_bool
1471     { \bool_not_p:n \l_@@_NiceTabular_bool }
1472     { \bool_not_p:n \l_@@_vlines_bool }
1473     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1474   }
1475   { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1476 }

```

We add a last column to raise a good error message when the user put more columns than allowed by its preamble. However, for technical reasons, it’s not possible to do that in `{NiceTabular*}` (`\l_@@_tabular_width_dim=0pt`).

```

1477 \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1478 {
1479   \tl_gput_right:Nn \g_@@_preamble_tl
1480   { > { \@@_error_too_much_cols: } 1 }
1481 }

```

Now, we have to close the TeX group which was opened for the redefinition of the columns of type `w` and `W`.

```

1482 \group_end:
1483 }

1484 \cs_new_protected:Npn \@@_patch_preamble:n #1
1485 {
1486   \str_case:nnF { #1 }
1487   {
1488     c { \@@_patch_preamble_i:n #1 }
1489     l { \@@_patch_preamble_i:n #1 }
1490     r { \@@_patch_preamble_i:n #1 }
1491     > { \@@_patch_preamble_ii:nn #1 }
1492     ! { \@@_patch_preamble_ii:nn #1 }
1493     @ { \@@_patch_preamble_ii:nn #1 }
1494     | { \@@_patch_preamble_iii:n #1 }
1495     p { \@@_patch_preamble_iv:nnn t #1 }
1496     m { \@@_patch_preamble_iv:nnn c #1 }
1497     b { \@@_patch_preamble_iv:nnn b #1 }
1498     \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1499     \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1500     \@@_true_c: { \@@_patch_preamble_vi:n #1 }
1501     ( { \@@_patch_preamble_vii:n #1 }
1502     [ { \@@_patch_preamble_vii:n #1 }
1503     \{ { \@@_patch_preamble_vii:n #1 }
1504     ) { \@@_patch_preamble_viii:n #1 }
1505     ] { \@@_patch_preamble_viii:n #1 }
1506     \} { \@@_patch_preamble_viii:n #1 }

```

```

1507     C { \@@_error:nn { old~column~type } #1 }
1508     L { \@@_error:nn { old~column~type } #1 }
1509     R { \@@_error:nn { old~column~type } #1 }
1510     \q_stop { }
1511   }
1512   {
1513     \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1514     { \@@_patch_preamble_xi:n #1 }
1515     {
1516       \str_if_eq:VnTF \l_@@_letter_vlism_tl { #1 }
1517       {
1518         \seq_gput_right:Nx \g_@@_cols_vlism_seq
1519         { \int_eval:n { \c@jCol + 1 } } }
1520       \tl_gput_right:Nx \g_@@_preamble_tl
1521       { \exp_not:N ! { \skip_horizontal:N \arrayrulewidth } }
1522       \@@_patch_preamble:n
1523     }
1524     {
1525       \bool_lazy_and:nnTF
1526       { \str_if_eq_p:nn { : } { #1 } }
1527       \c_@@_arydshln_loaded_bool
1528       {
1529         \tl_gput_right:Nn \g_@@_preamble_tl { : }
1530         \@@_patch_preamble:n
1531       }
1532       { \@@_fatal:nn { unknown~column~type } { #1 } }
1533     }
1534   }
1535 }
1536 }

```

For c, l and r

```

1537 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1538 {
1539   \tl_gput_right:Nn \g_@@_preamble_tl
1540   {
1541     > { \@@_Cell: \tl_set:Nn \l_@@_cell_type_tl { #1 } }
1542     #1
1543     < \@@_end_Cell:
1544   }

```

We increment the counter of columns and then we test for the presence of a <.

```

1545     \int_gincr:N \c@jCol
1546     \@@_patch_preamble_x:n
1547   }

```

For >, ! and @

```

1548 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1549 {
1550   \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
1551   \@@_patch_preamble:n
1552 }

```

For |

```

1553 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1554 {

```

\l\_tmpa\_int is the number of successive occurrences of |

```

1555     \int_incr:N \l_tmpa_int
1556     \@@_patch_preamble_iii_i:n
1557   }

```

```

1558 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1559 {
1560   \str_if_eq:nnTF { #1 } |
1561   { \@@_patch_preamble_iii:n | }
1562   {
1563     \tl_gput_right:Nx \g_@@_preamble_tl
1564     {
1565       \exp_not:N !
1566       {
1567         \skip_horizontal:n
1568         {
1569           \dim_eval:n
1570           {
1571             \arrayrulewidth * \l_tmpa_int
1572             + \doublerulesep * ( \l_tmpa_int - 1)
1573           }
1574         }
1575       }
1576     }
1577     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1578     { \@@_vline:nn { \@@_succ:n \c@jCol } { \int_use:N \l_tmpa_int } }
1579     \int_zero:N \l_tmpa_int
1580     \@@_patch_preamble:n #1
1581   }
1582 }

```

For p, m and b

```

1583 \cs_new_protected:Npn \@@_patch_preamble_iv:nnn #1 #2 #3
1584 {
1585   \tl_gput_right:Nn \g_@@_preamble_tl
1586   {
1587     > {
1588       \@@_Cell:
1589       \begin { minipage } [ #1 ] { #3 }
1590       \mode_leave_vertical:
1591       \arraybackslash
1592       \vrule height \box_ht:N \@arstrutbox depth 0 pt width 0 pt % v. 5.11
1593     }
1594     c
1595     < {
1596       \vrule height 0 pt depth \box_dp:N \@arstrutbox width 0 pt % v. 5.11
1597       \end { minipage }
1598       \@@_end_Cell:
1599     }
1600   }

```

We increment the counter of columns, and then we test for the presence of a <.

```

1601   \int_gincr:N \c@jCol
1602   \@@_patch_preamble_x:n
1603 }

```

For w and W

```

1604 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1605 {
1606   \tl_gput_right:Nn \g_@@_preamble_tl
1607   {
1608     > {
1609       \hbox_set:Nw \l_@@_cell_box
1610       \@@_Cell:
1611       \tl_set:Nn \l_@@_cell_type_tl { #3 }
1612     }
1613     c
1614     < {
1615       \@@_end_Cell:

```

```

1616         #1
1617         \hbox_set_end:
1618         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1619         \@@_adjust_size_box:
1620         \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
1621     }
1622 }

```

We increment the counter of columns and then we test for the presence of a <.

```

1623     \int_gincr:N \c@jCol
1624     \@@_patch_preamble_x:n
1625 }

```

For \@@\_true\_c: which will appear in our redefinition of the columns of type S (of siunitx).

```

1626 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1627 {
1628     \tl_gput_right:Nn \g_@@_preamble_tl { c }

```

We increment the counter of columns and then we test for the presence of a <.

```

1629     \int_gincr:N \c@jCol
1630     \@@_patch_preamble_x:n
1631 }

```

For (, [ and \{.

```

1632 \cs_new_protected:Npn \@@_patch_preamble_vii:n #1
1633 {
1634     \bool_if:NT \l_@@_small_bool
1635     { \@@_fatal:n { Delimiter-with-small } }

```

If we are before the column 1, we reserve space for the left delimiter.

```

1636     \int_compare:nNnT \c@jCol = \c_zero_int
1637     { \tl_gput_right:Nx \g_@@_preamble_tl { ! { \enskip } } }
1638     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1639     { \@@_delimiter:nnn #1 { \@@_succ:n \c@jCol } \c_true_bool }
1640     \@@_patch_preamble:n
1641 }

```

For ), ] and \}.

```

1642 \cs_new_protected:Npn \@@_patch_preamble_viii:n #1
1643 {
1644     \bool_if:NT \l_@@_small_bool
1645     { \@@_fatal:n { Delimiter-with-small } }
1646     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1647     { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }

```

After this closing delimiter, we have to test whether there is a opening delimiter (we consider that there can't be a second closing delimiter) in order to add an horizontal space (equal to 0.5 em).

```

1648     \@@_patch_preamble_viii_i:n
1649 }

1650 \cs_new_protected:Npn \@@_patch_preamble_viii_i:n #1
1651 {
1652     \bool_lazy_any:nT
1653     {
1654         { \str_if_eq_p:nn { #1 } ( }
1655         { \str_if_eq_p:nn { #1 } [ }
1656         { \str_if_eq_p:nn { #1 } \{ }
1657         { \str_if_eq_p:nn { #1 } \q_stop }
1658     }
1659     { \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } } }
1660     \@@_patch_preamble:n #1
1661 }

```

```

1662 \cs_new_protected:Npn \@@_patch_preamble_xi:n #1
1663 {
1664   \tl_gput_right:Nn \g_@@_preamble_tl
1665   { ! { \skip_horizontal:N 2\l_@@_radius_dim } }

```

The command `\@@_vdottedline:n` is protected, and, therefore, won't be expanded before writing on `\g_@@_internal_code_after_tl`.

```

1666   \tl_gput_right:Nx \g_@@_internal_code_after_tl
1667   { \@@_vdottedline:n { \int_use:N \c@jCol } }
1668   \@@_patch_preamble:n
1669 }

```

After a specifier of column, we have to test whether there is one or several `<{...}` because, after those potential `<{...}`, we have to insert `!\skip_horizontal:N ...` when the key `vlines` is used.

```

1670 \cs_new_protected:Npn \@@_patch_preamble_x:n #1
1671 {
1672   \str_if_eq:nnTF { #1 } { < }
1673   \@@_patch_preamble_ix:n
1674   {
1675     \bool_if:NT \l_@@_vlines_bool
1676     {
1677       \tl_gput_right:Nn \g_@@_preamble_tl
1678       { ! { \skip_horizontal:N \arrayrulewidth } }
1679     }
1680     \@@_patch_preamble:n { #1 }
1681   }
1682 }
1683 \cs_new_protected:Npn \@@_patch_preamble_ix:n #1
1684 {
1685   \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
1686   \@@_patch_preamble_x:n
1687 }

```

The command `\@@_put_box_in_flow:` puts the box `\l_tmpa_box` (which contains the array) in the flow. It is used for the environments with delimiters. First, we have to modify the height and the depth to take back into account the potential exterior rows (the total height of the first row has been computed in `\l_tmpa_dim` and the total height of the potential last row in `\l_tmpb_dim`).

```

1688 \cs_new_protected:Npn \@@_put_box_in_flow:
1689 {
1690   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1691   \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1692   \tl_if_eq:NnTF \l_@@_baseline_tl { c }
1693   { \box_use_drop:N \l_tmpa_box }
1694   \@@_put_box_in_flow_i:
1695 }

```

The command `\@@_put_box_in_flow_i:` is used when the value of `\l_@@_baseline_tl` is different of `c` (which is the initial value and the most used).

```

1696 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1697 {
1698   \pgfpicture
1699   \@@_qpoint:n { row - 1 }
1700   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1701   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1702   \dim_gadd:Nn \g_tmpa_dim \pgf@y
1703   \dim_gset:Nn \g_tmpa_dim { 0.5 \g_tmpa_dim }

```

Now, `\g_tmpa_dim` contains the  $y$ -value of the center of the array (the delimiters are centered in relation with this value).

```

1704   \str_if_in:NnTF \l_@@_baseline_tl { line- }
1705   {
1706     \int_set:Nn \l_tmpa_int
1707     {

```



```

1708         \str_range:Nnn
1709         \l_@@_baseline_tl
1710         6
1711         { \tl_count:V \l_@@_baseline_tl }
1712     }
1713     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1714 }
1715 {
1716     \str_case:VnF \l_@@_baseline_tl
1717     {
1718         { t } { \int_set:Nn \l_tmpa_int 1 }
1719         { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1720     }
1721     { \int_set:Nn \l_tmpa_int \l_@@_baseline_tl }
1722     \bool_lazy_or:nnT
1723     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1724     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1725     {
1726         \@@_error:n { bad~value~for~baseline }
1727         \int_set:Nn \l_tmpa_int 1
1728     }
1729     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

We take into account the position of the mathematical axis.

```

1730     \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
1731 }
1732 \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

Now, `\g_tmpa_dim` contains the value of the  $y$  translation we have to to.

```

1733     \endpgfpicture
1734     \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1735     \box_use_drop:N \l_tmpa_box
1736 }

```

The following command is *always* used by `{NiceArrayWithDelims}` (even if, in fact, there is no tabular notes: in fact, it's not possible to know whether there is tabular notes or not before the composition of the blocks).

```

1737 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
1738 {

```

We need a `{minipage}` because we will insert a LaTeX list for the tabular notes (that means that a `\vtop{\hsize=...}` is not enough).

```

1739     \begin { minipage } [ t ] { \box_wd:N \l_@@_the_array_box }
1740     \box_use_drop:N \l_@@_the_array_box

```

We have to draw the blocks right now because there may be tabular notes in some blocks (which are not mono-column: the blocks which are mono-column have been composed in boxes yet)... and we have to create (potentially) the extra nodes before creating the blocks since there are `medium` nodes to create for the blocks.

```

1741     \@@_create_extra_nodes:
1742     \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
1743     \bool_lazy_or:nnT
1744     { \int_compare_p:nNn \c@tabularnote > 0 }
1745     { ! \tl_if_empty_p:V \l_@@_tabularnote_tl }
1746     \@@_insert_tabularnotes:
1747     \end { minipage }
1748 }

```

```

1749 \cs_new_protected:Npn \@@_insert_tabularnotes:
1750 {
1751     \skip_vertical:N 0.65ex

```

The TeX group is for potential specifications in the `\l_@@_notes_code_before_tl`.

```

1752     \group_begin:
1753     \l_@@_notes_code_before_tl
1754     \tl_if_empty:NF \l_@@_tabularnote_tl { \l_@@_tabularnote_tl \par }

```

We compose the tabular notes with a list of `enumitem`. The `\strut` and the `\unskip` are designed to give the ability to put a `\bottomrule` at the end of the notes with a good vertical space.

```

1755 \int_compare:nNnT \c@tabularnote > 0
1756 {
1757   \bool_if:NTF \l_@@_notes_para_bool
1758   {
1759     \begin { tabularnotes* }
1760     \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1761     \end { tabularnotes* }

```

The following `\par` is mandatory for the event that the user has put `\footnotesize` (for example) in the notes/code-before.

```

1762     \par
1763   }
1764 {
1765   \tabularnotes
1766   \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1767   \endtabularnotes
1768 }
1769 }
1770 \unskip
1771 \group_end:
1772 \bool_if:NT \l_@@_notes_bottomrule_bool
1773 {
1774   \bool_if:NTF \c_@@_booktabs_loaded_bool
1775   {

```

The two dimensions `\aboverulesep` et `\heavyrulewidth` are parameters defined by `booktabs`.

```

1776     \skip_vertical:N \aboverulesep
\CT@arc@ is the specification of color defined by colortbl but you use it even if colortbl is not loaded.
1777     { \CT@arc@ \hrule height \heavyrulewidth }
1778   }
1779   { @@_error:n { bottomrule-without-booktabs } }
1780 }
1781 \l_@@_notes_code_after_tl
1782 \seq_gclear:N \g_@@_tabularnotes_seq
1783 \int_gzero:N \c@tabularnote
1784 }

```

The case of `baseline` equal to `b`. Remember that, when the key `b` is used, the `{array}` (of array) is constructed with the option `t` (and not `b`). Now, we do the translation to take into account the option `b`.

```

1785 \cs_new_protected:Npn @@_use_arraybox_with_notes_b:
1786 {
1787   \pgfpicture
1788   @@_qpoint:n { row - 1 }
1789   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1790   @@_qpoint:n { row - \int_use:N \c@iRow - base }
1791   \dim_gsub:Nn \g_tmpa_dim \pgf@y
1792   \endpgfpicture
1793   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1794   \int_compare:nNnT \l_@@_first_row_int = 0
1795   {
1796     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1797     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1798   }
1799   \box_move_up:nn \g_tmpa_dim { \hbox { @@_use_arraybox_with_notes_c: } }
1800 }

```

Now, the general case.

```

1801 \cs_new_protected:Npn @@_use_arraybox_with_notes:
1802 {

```

We convert a value of `t` to a value of 1.

```
1803 \tl_if_eq:NnT \l_@@_baseline_tl { t }
1804 { \tl_set:Nn \l_@@_baseline_tl { 1 } }
```

Now, we convert the value of `\l_@@_baseline_tl` (which should represent an integer) to an integer stored in `\l_tmpa_int`.

```
1805 \pgfpicture
1806 \@@_qpoint:n { row - 1 }
1807 \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1808 \str_if_in:NnTF \l_@@_baseline_tl { line- }
1809 {
1810   \int_set:Nn \l_tmpa_int
1811   {
1812     \str_range:Nnn
1813     \l_@@_baseline_tl
1814     6
1815     { \tl_count:V \l_@@_baseline_tl }
1816   }
1817   \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1818 }
1819 {
1820   \int_set:Nn \l_tmpa_int \l_@@_baseline_tl
1821   \bool_lazy_or:nnT
1822   { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1823   { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1824   {
1825     \@@_error:n { bad~value~for~baseline }
1826     \int_set:Nn \l_tmpa_int 1
1827   }
1828   \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1829 }
1830 \dim_gsub:Nn \g_tmpa_dim \pgf@y
1831 \endpgfpicture
1832 \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1833 \int_compare:nNnT \l_@@_first_row_int = 0
1834 {
1835   \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1836   \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1837 }
1838 \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
1839 }
```

The command `\@@_put_box_in_flow_bis:` is used when the option `max-delimiter-width` is used because, in this case, we have to adjust the widths of the delimiters. The arguments `#1` and `#2` are the delimiters specified by the user.

```
1840 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1841 {
```

We will compute the real width of both delimiters used.

```
1842 \dim_zero_new:N \l_@@_real_left_delim_dim
1843 \dim_zero_new:N \l_@@_real_right_delim_dim
1844 \hbox_set:Nn \l_tmpb_box
1845 {
1846   \c_math_toggle_token
1847   \left #1
1848   \vcenter
1849   {
1850     \vbox_to_ht:nn
1851     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1852     { }
1853   }
1854   \right .
1855   \c_math_toggle_token
```

```

1856     }
1857     \dim_set:Nn \l_@@_real_left_delim_dim
1858     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
1859     \hbox_set:Nn \l_tmpb_box
1860     {
1861       \c_math_toggle_token
1862       \left .
1863       \vbox_to_ht:nn
1864       { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1865       { }
1866       \right #2
1867       \c_math_toggle_token
1868     }
1869     \dim_set:Nn \l_@@_real_right_delim_dim
1870     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }

```

Now, we can put the box in the TeX flow with the horizontal adjustments on both sides.

```

1871     \skip_horizontal:N \l_@@_left_delim_dim
1872     \skip_horizontal:N -\l_@@_real_left_delim_dim
1873     \@@_put_box_in_flow:
1874     \skip_horizontal:N \l_@@_right_delim_dim
1875     \skip_horizontal:N -\l_@@_real_right_delim_dim
1876   }

```

The construction of the array in the environment `{NiceArrayWithDelims}` is, in fact, done by the environment `{@@-light-syntax}` or by the environment `{@@-normal-syntax}` (whether the option `light-syntax` is in force or not). When the key `light-syntax` is not used, the construction is a standard environment (and, thus, it's possible to use verbatim in the array).

```

1877 \NewDocumentEnvironment { @@-normal-syntax } { }

```

First, we test whether the environment is empty. If it is empty, we raise a fatal error (it's only a security). In order to detect whether it is empty, we test whether the next token is `\end` and, if it's the case, we test if this is the end of the environment (if it is not, an standard error will be raised by LaTeX for incorrect nested environments).

```

1878 {
1879   \peek_meaning_ignore_spaces:NTF \end \@@_analyze_end:Nn

```

Here is the call to `\array` (we have a dedicated macro `\@@_array:` because of compatibility with the classes `revtex4-1` and `revtex4-2`).

```

1880     { \exp_args:NV \@@_array: \g_@@_preamble_tl }
1881   }
1882   {
1883     \@@_create_col_nodes:
1884     \endarray
1885   }

```

When the key `light-syntax` is in force, we use an environment which takes its whole body as an argument (with the specifier `b` of `xparse`).

```

1886 \NewDocumentEnvironment { @@-light-syntax } { b }
1887 {

```

First, we test whether the environment is empty. It's only a security. Of course, this test is more easy than the similar test for the “normal syntax” because we have the whole body of the environment in #1.

```

1888   \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1889   \tl_map_inline:nn { #1 }
1890   {
1891     \str_if_eq:nnT { ##1 } { & }
1892     { \@@_fatal:n { ampersand-in-light-syntax } }
1893     \str_if_eq:nnT { ##1 } { \ }
1894     { \@@_fatal:n { double-backslash-in-light-syntax } }
1895   }

```

Now, you extract the `\CodeAfter` of the body of the environment. Maybe, there is no command `\CodeAfter` in the body. That's why you put a marker `\CodeAfter` after `#1`. If there is yet a `\CodeAfter` in `#1`, this second (or third...) `\CodeAfter` will be caught in the value of `\g_nicematrix_code_after_tl`. That doesn't matter because `\CodeAfter` will be set to *no-op* before the execution of `\g_nicematrix_code_after_tl`.

```
1896     \@@_light_syntax_i #1 \CodeAfter \q_stop
1897 }
```

Now, the second part of the environment. It is empty. That's not surprising because we have caught the whole body of the environment with the specifier `b` provided by `xparse`.

```
1898 { }
1899 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1900 {
1901     \tl_gput_right:Nn \g_nicematrix_code_after_tl { #2 }
```

The body of the array, which is stored in the argument `#1`, is now splitted into items (and *not* tokens).

```
1902     \seq_gclear_new:N \g_@@_rows_seq
1903     \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1904     \exp_args:NNV \seq_gset_split:Nnn \g_@@_rows_seq \l_@@_end_of_row_tl { #1 }
```

If the environment uses the option `last-row` without value (i.e. without saying the number of the rows), we have now the opportunity to know that value. We do it, and so, if the token list `\l_@@_code_for_last_row_tl` is not empty, we will use directly where it should be.

```
1905     \int_compare:nNnT \l_@@_last_row_int = { -1 }
1906         { \int_set:Nn \l_@@_last_row_int { \seq_count:N \g_@@_rows_seq } }
```

Here is the call to `\array` (we have a dedicated macro `\@@_array`: because of compatibility with the classes `revtex4-1` and `revtex4-2`).

```
1907     \exp_args:NV \@@_array: \g_@@_preamble_tl
```

We need a global affectation because, when executing `\l_tmpa_tl`, we will exit the first cell of the array.

```
1908     \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1909     \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1910     \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1911     \@@_create_col_nodes:
1912     \endarray
1913 }
1914 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1915 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }
1916 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1917 {
1918     \seq_gclear_new:N \g_@@_cells_seq
1919     \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1920     \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1921     \l_tmpa_tl
1922     \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1923 }
```

The following command is used by the code which detects whether the environment is empty (we raise a fatal error in this case: it's only a security).

```
1924 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1925 {
1926     \str_if_eq:VnT \g_@@_name_env_str { #2 }
1927         { \@@_fatal:n { empty-environment } } }
```

We reup in the stream the `\end{...}` we have extracted and the user will have an error for incorrect nested environments.

```
1928     \end { #2 }
1929 }
```

The command `\@@_create_col_nodes:` will construct a special last row. That last row is a false row used to create the col nodes and to fix the width of the columns (when the array is constructed with an option which specify the width of the columns).

```

1930 \cs_new:Npn \@@_create_col_nodes:
1931 {
1932   \crrc
1933   \int_compare:nNnT \l_@@_first_col_int = 0
1934   {
1935     \omit
1936     \hbox_overlap_left:n
1937     {
1938       \bool_if:NT \l_@@_code_before_bool
1939       { \pgfsys@markposition { \@@_env: - col - 0 } }
1940       \pgfpicture
1941       \pgfrememberpicturerepositiononpagetrue
1942       \pgfcoordinate { \@@_env: - col - 0 } \pgfpintorigin
1943       \str_if_empty:NF \l_@@_name_str
1944       { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1945       \endpgfpicture
1946       \skip_horizontal:N 2\col@sep
1947       \skip_horizontal:N \g_@@_width_first_col_dim
1948     }
1949     &
1950   }
1951   \omit

```

The following instruction must be put after the instruction `\omit`.

```

1952   \bool_gset_true:N \g_@@_row_of_col_done_bool

```

First, we put a col node on the left of the first column (of course, we have to do that *after* the `\omit`).

```

1953   \int_compare:nNnTF \l_@@_first_col_int = 0
1954   {
1955     \bool_if:NT \l_@@_code_before_bool
1956     {
1957       \hbox
1958       {
1959         \skip_horizontal:N -0.5\arrayrulewidth
1960         \pgfsys@markposition { \@@_env: - col - 1 }
1961         \skip_horizontal:N 0.5\arrayrulewidth
1962       }
1963     }
1964     \pgfpicture
1965     \pgfrememberpicturerepositiononpagetrue
1966     \pgfcoordinate { \@@_env: - col - 1 }
1967     { \pgfpint { - 0.5 \arrayrulewidth } \c_zero_dim }
1968     \str_if_empty:NF \l_@@_name_str
1969     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1970     \endpgfpicture
1971   }
1972   {
1973     \bool_if:NT \l_@@_code_before_bool
1974     {
1975       \hbox
1976       {
1977         \skip_horizontal:N 0.5\arrayrulewidth
1978         \pgfsys@markposition { \@@_env: - col - 1 }
1979         \skip_horizontal:N -0.5\arrayrulewidth
1980       }
1981     }
1982     \pgfpicture
1983     \pgfrememberpicturerepositiononpagetrue
1984     \pgfcoordinate { \@@_env: - col - 1 }
1985     { \pgfpint { 0.5 \arrayrulewidth } \c_zero_dim }

```

```

1986     \str_if_empty:NF \l_@@_name_str
1987     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1988     \endpgfpicture
1989 }

```

We compute in `\g_tmpa_skip` the common width of the columns (it's a skip and not a dimension). We use a global variable because we are in a cell of an `\halign` and because we have to use this variable in other cells (of the same row). The affectation of `\g_tmpa_skip`, like all the affectations, must be done after the `\omit` of the cell.

We give a default value for `\g_tmpa_skip` (0 pt plus 1 fill) but it will just after be erased by a fixed value in the concerned cases.

```

1990     \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1991     \bool_if:NF \l_@@_auto_columns_width_bool
1992     { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
1993     {
1994         \bool_lazy_and:nnTF
1995         \l_@@_auto_columns_width_bool
1996         { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1997         { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1998         { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1999         \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
2000     }
2001     \skip_horizontal:N \g_tmpa_skip
2002     \hbox
2003     {
2004         \bool_if:NT \l_@@_code_before_bool
2005         {
2006             \hbox
2007             {
2008                 \skip_horizontal:N -0.5\arrayrulewidth
2009                 \pgfsys@markposition { \@@_env: - col - 2 }
2010                 \skip_horizontal:N 0.5\arrayrulewidth
2011             }
2012         }
2013         \pgfpicture
2014         \pgfrememberpicturepositiononpagetrue
2015         \pgfcoordinate { \@@_env: - col - 2 }
2016         { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2017         \str_if_empty:NF \l_@@_name_str
2018         { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
2019         \endpgfpicture
2020     }

```

We begin a loop over the columns. The integer `\g_tmpa_int` will be the number of the current column. This integer is used for the Tikz nodes.

```

2021     \int_gset:Nn \g_tmpa_int 1
2022     \bool_if:NTF \g_@@_last_col_found_bool
2023     { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
2024     { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
2025     {
2026         &
2027         \omit
2028         \int_gincr:N \g_tmpa_int

```

The incrementation of the counter `\g_tmpa_int` must be done after the `\omit` of the cell.

```

2029         \skip_horizontal:N \g_tmpa_skip
2030         \bool_if:NT \l_@@_code_before_bool
2031         {
2032             \hbox
2033             {
2034                 \skip_horizontal:N -0.5\arrayrulewidth
2035                 \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2036                 \skip_horizontal:N 0.5\arrayrulewidth
2037             }

```

2038 }

We create the col node on the right of the current column.

```

2039 \pgfpicture
2040 \pgfrememberpicturepositiononpagetrue
2041 \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2042 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2043 \str_if_empty:NF \l_@@_name_str
2044 {
2045 \pgfnodealias
2046 { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
2047 { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2048 }
2049 \endpgfpicture
2050 }
2051 \bool_if:NT \g_@@_last_col_found_bool
2052 {
2053 \hbox_overlap_right:n
2054 {
2055 % \skip_horizontal:N \col@sep
2056 \skip_horizontal:N \g_@@_width_last_col_dim
2057 \bool_if:NT \l_@@_code_before_bool
2058 {
2059 \pgfsys@markposition
2060 { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2061 }
2062 \pgfpicture
2063 \pgfrememberpicturepositiononpagetrue
2064 \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2065 \pgfpointorigin
2066 \str_if_empty:NF \l_@@_name_str
2067 {
2068 \pgfnodealias
2069 { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
2070 { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2071 }
2072 \endpgfpicture
2073 }
2074 }
2075 \cr
2076 }

```

Here is the preamble for the “first column” (if the user uses the key `first-col`)

```

2077 \tl_const:Nn \c_@@_preamble_first_col_tl
2078 {
2079 >
2080 {

```

At the beginning of the cell, we link `\CodeAfter` to a command which do *not* begin with `\omit` (whereas the standard version of `\CodeAfter` begins with `\omit`).

```

2081 \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n
2082 \bool_gset_true:N \g_@@_after_col_zero_bool
2083 \@@_begin_of_row:

```

The contents of the cell is constructed in the box `\l_@@_cell_box` because we have to compute some dimensions of this box.

```

2084 \hbox_set:Nw \l_@@_cell_box
2085 \@@_math_toggle_token:
2086 \bool_if:NT \l_@@_small_bool \scriptstyle

```

We insert `\l_@@_code_for_first_col_tl...` but we don’t insert it in the potential “first row” and in the potential “last row”.

```

2087 \bool_lazy_and:nnT
2088 { \int_compare_p:nNn \c@iRow > 0 }

```



```

2089     {
2090         \bool_lazy_or_p:nn
2091         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2092         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2093     }
2094     {
2095         \l_@@_code_for_first_col_tl
2096         \xglobal \colorlet { nicematrix-first-col } { . }
2097     }
2098 }

```

Be careful: despite this letter `l` the cells of the “first column” are composed in a `R` manner since they are composed in a `\hbox_overlap_left:n`.

```

2099     l
2100     <
2101     {
2102         \@@_math_toggle_token:
2103         \hbox_set_end:
2104         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2105         \@@_adjust_size_box:
2106         \@@_update_for_first_and_last_row:

```

We actualise the width of the “first column” because we will use this width after the construction of the array.

```

2107         \dim_gset:Nn \g_@@_width_first_col_dim
2108         { \dim_max:nn \g_@@_width_first_col_dim { \box_wd:N \l_@@_cell_box } }

```

The content of the cell is inserted in an overlapping position.

```

2109         \hbox_overlap_left:n
2110         {
2111             \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2112             \@@_node_for_the_cell:
2113             { \box_use_drop:N \l_@@_cell_box }
2114             \skip_horizontal:N \l_@@_left_delim_dim
2115             \skip_horizontal:N \l_@@_left_margin_dim
2116             \skip_horizontal:N \l_@@_extra_left_margin_dim
2117         }
2118         \bool_gset_false:N \g_@@_empty_cell_bool
2119         \skip_horizontal:N -2\col@sep
2120     }
2121 }

```

Here is the preamble for the “last column” (if the user uses the key `last-col`).

```

2122 \tl_const:Nn \c_@@_preamble_last_col_tl
2123 {
2124     >
2125     {

```

At the beginning of the cell, we link `\CodeAfter` to a command which do *not* begin with `\omit` (whereas the standard version of `\CodeAfter` begins with `\omit`).

```

2126         \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n

```

With the flag `\g_@@_last_col_found_bool`, we will know that the “last column” is really used.

```

2127         \bool_gset_true:N \g_@@_last_col_found_bool
2128         \int_gincr:N \c@jCol
2129         \int_gset_eq:NN \g_@@_col_total_int \c@jCol

```

The contents of the cell is constructed in the box `\l_tmpa_box` because we have to compute some dimensions of this box.

```

2130         \hbox_set:Nw \l_@@_cell_box
2131         \@@_math_toggle_token:
2132         \bool_if:NT \l_@@_small_bool \scriptstyle

```

We insert `\l_@@_code_for_last_col_tl...` but we don't insert it in the potential “first row” and in the potential “last row”.

```

2133     \int_compare:nNnT \c@iRow > 0
2134     {
2135         \bool_lazy_or:nnT
2136         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2137         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2138         {
2139             \l_@@_code_for_last_col_tl
2140             \xglobal \colorlet { nicematrix-last-col } { . }
2141         }
2142     }
2143 }
2144 1
2145 <
2146 {
2147     \@@_math_toggle_token:
2148     \hbox_set_end:
2149     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2150     \@@_adjust_size_box:
2151     \@@_update_for_first_and_last_row:

```

We actualise the width of the “last column” because we will use this width after the construction of the array.

```

2152     \dim_gset:Nn \g_@@_width_last_col_dim
2153     { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
2154     \skip_horizontal:N -2\col@sep

```

The content of the cell is inserted in an overlapping position.

```

2155     \hbox_overlap_right:n
2156     {
2157         \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2158         {
2159             \skip_horizontal:N \l_@@_right_delim_dim
2160             \skip_horizontal:N \l_@@_right_margin_dim
2161             \skip_horizontal:N \l_@@_extra_right_margin_dim
2162             \@@_node_for_the_cell:
2163         }
2164     }
2165     \bool_gset_false:N \g_@@_empty_cell_bool
2166 }
2167 }

```

The environment `{NiceArray}` is constructed upon the environment `{NiceArrayWithDelims}` but, in fact, there is a flag `\l_@@_NiceArray_bool`. In `{NiceArrayWithDelims}`, some special code will be executed if this flag is raised.

```

2168 \NewDocumentEnvironment { NiceArray } { }
2169 {
2170     \bool_set_true:N \l_@@_NiceArray_bool
2171     \str_if_empty:NT \g_@@_name_env_str
2172     { \str_gset:Nn \g_@@_name_env_str { NiceArray } }

```

We put `.` and `.` for the delimiters but, in fact, that doesn't matter because these arguments won't be used in `{NiceArrayWithDelims}` (because the flag `\l_@@_NiceArray_bool` is raised).

```

2173     \NiceArrayWithDelims . .
2174 }
2175 { \endNiceArrayWithDelims }

```

We create the variants of the environment `{NiceArrayWithDelims}`.

```

2176 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
2177 {

```

```

2178 \NewDocumentEnvironment { #1 NiceArray } { }
2179 {
2180   \str_if_empty:NT \g_@@_name_env_str
2181     { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
2182   \@@_test_if_math_mode:
2183   \NiceArrayWithDelims #2 #3
2184 }
2185 { \endNiceArrayWithDelims }
2186 }
2187 \@@_def_env:nnn p ( )
2188 \@@_def_env:nnn b [ ]
2189 \@@_def_env:nnn B \{ \}
2190 \@@_def_env:nnn v | |
2191 \@@_def_env:nnn V \| \|

```

## The environment {NiceMatrix} and its variants

```

2192 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
2193 {
2194   \bool_set_true:N \l_@@_Matrix_bool
2195   \use:c { #1 NiceArray }
2196   {
2197     *
2198     {
2199       \int_compare:nNnTF \l_@@_last_col_int < 0
2200         \c@MaxMatrixCols
2201         { \@@_pred:n \l_@@_last_col_int }
2202     }
2203     { > \@@_Cell: #2 < \@@_end_Cell: }
2204   }
2205 }
2206 \clist_map_inline:nn { { } , p , b , B , v , V }
2207 {
2208   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
2209   {
2210     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
2211     \tl_set:Nn \l_@@_type_of_col_tl c
2212     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
2213     \exp_args:Nne \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
2214   }
2215   { \use:c { end #1 NiceArray } }
2216 }

```

The following command will be linked to \NotEmpty in the environments of nicematrix.

```

2217 \cs_new_protected:Npn \@@_NotEmpty:
2218 { \bool_gset_true:N \g_@@_not_empty_cell_bool }

```

## The environments {NiceTabular} and {NiceTabular\*}

```

2219 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
2220 {
2221   \str_gset:Nn \g_@@_name_env_str { NiceTabular }
2222   \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
2223   \bool_set_true:N \l_@@_NiceTabular_bool
2224   \NiceArray { #2 }
2225 }
2226 { \endNiceArray }
2227 \NewDocumentEnvironment { NiceTabular* } { m 0 { } m ! 0 { } }

```

```

2228 {
2229   \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
2230   \dim_set:Nn \l_@@_tabular_width_dim { #1 }
2231   \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2232   \bool_set_true:N \l_@@_NiceTabular_bool
2233   \NiceArray { #3 }
2234 }
2235 { \endNiceArray }

```

## After the construction of the array

```

2236 \cs_new_protected:Npn \@@_after_array:
2237 {
2238   \group_begin:

```

When the option `last-col` is used in the environments with explicit preambles (like `{NiceArray}`, `{pNiceArray}`, etc.) a special type of column is used at the end of the preamble in order to compose the cells in an overlapping position (with `\hbox_overlap_right:n`) but (if `last-col` has been used), we don't have the number of that last column. However, we have to know that number for the color of the potential `\Vdots` drawn in that last column. That's why we fix the correct value of `\l_@@_last_col_int` in that case.

```

2239   \bool_if:NT \g_@@_last_col_found_bool
2240   { \int_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int }

```

If we are in an environment without preamble (like `{NiceMatrix}` or `{pNiceMatrix}`) and if the option `last-col` has been used without value we fix the real value of `\l_@@_last_col_int`.

```

2241   \bool_if:NT \l_@@_last_col_without_value_bool
2242   {
2243     \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
2244     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2245     \iow_shipout:Nx \@mainaux
2246     {
2247       \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
2248       { \int_use:N \g_@@_col_total_int }
2249     }
2250     \str_if_empty:NF \l_@@_name_str
2251     {
2252       \iow_shipout:Nx \@mainaux
2253       {
2254         \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
2255         { \int_use:N \g_@@_col_total_int }
2256       }
2257     }
2258     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2259   }

```

It's also time to give to `\l_@@_last_row_int` its real value. But, if the user had used the option `last-row` without value, we write in the aux file the number of that last row for the next run.

```

2260   \bool_if:NT \l_@@_last_row_without_value_bool
2261   {
2262     \dim_set_eq:NN \l_@@_last_row_int \g_@@_row_total_int

```

If the option `light-syntax` is used, we have nothing to write since, in this case, the number of rows is directly determined.

```

2263   \bool_if:NF \l_@@_light_syntax_bool
2264   {
2265     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2266     \iow_shipout:Nx \@mainaux
2267     {
2268       \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
2269       { \int_use:N \g_@@_row_total_int }
2270     }

```

If the environment has a name, we also write a value based on the name because it's more reliable than a value based on the number of the environment.

```

2271 \str_if_empty:NF \l_@@_name_str
2272 {
2273   \iow_shipout:Nx \@mainaux
2274   {
2275     \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
2276     { \int_use:N \g_@@_row_total_int }
2277   }
2278 }
2279 \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2280 }
2281 }

```

If the key code-before is used, we have to write on the aux file the actual size of the array.

```

2282 \bool_if:NT \l_@@_code_before_bool
2283 {
2284   \iow_now:Nn \@mainaux \ExplSyntaxOn
2285   \iow_now:Nx \@mainaux
2286   { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
2287   \iow_now:Nx \@mainaux
2288   {
2289     \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
2290     {
2291       \int_use:N \l_@@_first_row_int ,
2292       \int_use:N \g_@@_row_total_int ,
2293       \int_use:N \l_@@_first_col_int ,

```

If the user has used a key last-row in an environment with preamble (like {pNiceArray}) and that that last row has not been found, we have to increment the value because it will be decreased when used in the code-before.

```

2294 \bool_lazy_and:nnTF
2295 { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
2296 { \bool_not_p:n \g_@@_last_col_found_bool }
2297 \@@_succ:n
2298 \int_use:N
2299 \g_@@_col_total_int
2300 }

```

We write also the potential content of \g\_@@\_pos\_of\_blocks\_seq (it will be useful if the commands \rowcolors is used with the key respect-blocks).

```

2301 \seq_gset_from_clist:cn
2302 { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
2303 { \seq_use:Nnnn \g_@@_pos_of_blocks_seq , , , }
2304 }
2305 \iow_now:Nn \@mainaux \ExplSyntaxOff
2306 }
2307
2307 \@@_create_diag_nodes:
2308 \str_if_empty:NF \l_@@_name_str
2309 {
2310   \pgfpicture
2311   \pgfrememberpicturepositiononpagetrue
2312   \pgfnodealias { \l_@@_name_str - ##1 } { \@@_env: - ##1 }
2313   \endpgfpicture
2314 }

```

By default, the diagonal lines will be parallelized<sup>50</sup>. There are two types of diagonals lines: the \Ddots diagonals and the \iddots diagonals. We have to count both types in order to know whether a diagonal is the first of its type in the current {NiceArray} environment.

```

2315 \bool_if:NT \l_@@_parallelize_diags_bool
2316 {
2317   \int_gzero_new:N \g_@@_ddots_int
2318   \int_gzero_new:N \g_@@_iddots_int

```

The dimensions \g\_@@\_delta\_x\_one\_dim and \g\_@@\_delta\_y\_one\_dim will contain the  $\Delta_x$  and  $\Delta_y$  of the first \Ddots diagonal. We have to store these values in order to draw the others \Ddots

<sup>50</sup>It's possible to use the option parallelize-diags to disable this parallelization.

diagonals parallel to the first one. Similarly `\g_@@_delta_x_two_dim` and `\g_@@_delta_y_two_dim` are the  $\Delta_x$  and  $\Delta_y$  of the first `\Iddots` diagonal.

```

2319     \dim_gzero_new:N \g_@@_delta_x_one_dim
2320     \dim_gzero_new:N \g_@@_delta_y_one_dim
2321     \dim_gzero_new:N \g_@@_delta_x_two_dim
2322     \dim_gzero_new:N \g_@@_delta_y_two_dim
2323   }
2324   \int_zero_new:N \l_@@_initial_i_int
2325   \int_zero_new:N \l_@@_initial_j_int
2326   \int_zero_new:N \l_@@_final_i_int
2327   \int_zero_new:N \l_@@_final_j_int
2328   \bool_set_false:N \l_@@_initial_open_bool
2329   \bool_set_false:N \l_@@_final_open_bool

```

If the option `small` is used, the values `\l_@@_radius_dim` and `\l_@@_inter_dots_dim` (used to draw the dotted lines created by `\hdottedline` and `\vdotteline` and also for all the other dotted lines when `line-style` is equal to `standard`, which is the initial value) are changed.

```

2330   \bool_if:NT \l_@@_small_bool
2331   {
2332     \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2333     \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em }

```

The dimension `\l_@@_xdots_shorten_dim` corresponds to the option `xdots/shorten` available to the user. That's why we give a new value according to the current value, and not an absolute value.

```

2334     \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2335   }

```

Now, we actually draw the dotted lines (specified by `\Cdots`, `\Vdots`, etc.).

```

2336   \@@_draw_dotted_lines:

```

The following computes the “corners” (made up of empty cells) but if there is no corner to compute, it will do nothing.

```

2337   \@@_compute_corners:

```

The sequence `\g_@@_pos_of_blocks_seq` must be “adjusted” (for the case where the user have written something like `\Block{1-}`).

```

2338   \@@_adjust_pos_of_blocks_seq:

```

The following code is only for efficiency. We determine whether the potential horizontal and vertical rules are “complete”, that is to say drawn in the whole array. We are sure that all the rules will be complete when there is no block, no virtual block (determined by a command such as `\Cdots`, `\Vdots`, etc.) and no corners. In that case, we switch to a shortcut version of `\@@_vline_i:nn` and `\@@_hline:nn`.

```

2339   \bool_lazy_all:nT
2340   {
2341     { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
2342     { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
2343     { \seq_if_empty_p:N \l_@@_empty_corner_cells_seq }
2344   }
2345   {
2346     \cs_set_eq:NN \@@_vline_i:nn \@@_vline_i_complete:nn
2347     \cs_set_eq:NN \@@_hline_i:nn \@@_hline_i_complete:nn
2348   }
2349   \bool_if:NT \l_@@_hlines_bool \@@_draw_hlines:
2350   \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
2351   \g_@@_internal_code_after_tl
2352   \tl_gclear:N \g_@@_internal_code_after_tl

```

Now, the `\CodeAfter`.

```

2353   \bool_if:NT \c_@@_tikz_loaded_bool
2354   {

```

```

2355     \tikzset
2356     {
2357         every~picture / .style =
2358         {
2359             overlay ,
2360             remember~picture ,
2361             name-prefix = \@@_env: -
2362         }
2363     }
2364 }
2365 \cs_set_eq:NN \line \@@_line
2366 \cs_set_eq:NN \SubMatrix \@@_SubMatrix

```

When `light-syntax` is used, we insert systematically a `\CodeAfter` in the flow. Thus, it's possible to have two instructions `\CodeAfter` and the second may be in `\g_nicematrix_code_after_tl`. That's why we set `\Code-after` to be *no-op* now.

```

2367 \cs_set_eq:NN \CodeAfter \prg_do_nothing:

```

We clear the list of the names of the potential `\SubMatrix` that will appear in the `\CodeAfter` (unfortunately, that list has to be global).

```

2368 \seq_gclear:N \g_@@_submatrix_names_seq

```

And here's the `\CodeAfter`. Since the `\CodeAfter` may begin with an “argument” between square brackets of the options, we extract and treat that potential “argument” with the command `\@@_CodeAfter_keys:`.

```

2369 \exp_last_unbraced:NV \@@_CodeAfter_keys: \g_nicematrix_code_after_tl
2370 \scan_stop:
2371 \tl_gclear:N \g_nicematrix_code_after_tl
2372 \group_end:

```

`\g_nicematrix_code_before_tl` is for instructions in the cells of the array such as `\rowcolor` and `\cellcolor` (when the key `colortbl-like` is in force). These instructions will be written on the `aux` file to be added to the `code-before` in the next run.

```

2373 \tl_if_empty:NF \g_nicematrix_code_before_tl
2374 {

```

The command `\rowcolor` in `tabular` will in fact use `\rectanglecolor` in order to follow the behaviour of `\rowcolor` of `colortbl`. That's why there may be a command `\rectanglecolor` in `\g_nicematrix_code_before_tl`. In order to avoid an error during the expansion, we define a protected version of `\rectanglecolor`.

```

2375     \cs_set_protected:Npn \rectanglecolor { }
2376     \cs_set_protected:Npn \columncolor { }
2377     \iow_now:Nn \@mainaux \ExplSyntaxOn
2378     \iow_now:Nx \@mainaux
2379     {
2380         \tl_gset:cn
2381         { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
2382         { \exp_not:V \g_nicematrix_code_before_tl }
2383     }
2384     \iow_now:Nn \@mainaux \ExplSyntaxOff
2385     \bool_set_true:N \l_@@_code_before_bool
2386 }

2387 \str_gclear:N \g_@@_name_env_str
2388 \@@_restore_iRow_jCol:

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>51</sup>. This command is used by `\CT@arc@` but we use it also for compatibility with `colortbl`. But we want also to be able to use color for the rules of the array when `colortbl` is *not* loaded. That's why we do the following instruction which is in the patch of the end of arrays done by `colortbl`.

```

2389 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
2390 }

```

---

<sup>51</sup>e.g. `\color[rgb]{0.5,0.5,0}`

The following command will extract the potential options (between square brackets) at the beginning of the `\CodeAfter` (that is to say, when `\CodeAfter` is used, the options of that “command” `\CodeAfter`).

```
2391 \NewDocumentCommand \@@_CodeAfter_keys: { 0 { } }
2392 { \keys_set:nn { NiceMatrix / CodeAfter } { #1 } }
```

We remind that the first mandatory argument of the command `\Block` is the size of the block with the special format  $i-j$ . However, the user is allowed to omit  $i$  or  $j$  (or both). This will be interpreted as: the last row (resp. column) of the block will be the last row (resp. column) of the block (without the potential exterior row—resp. column—of the array). By convention, this is stored in `\g_@@_pos_of_blocks_seq` (and `\g_@@_blocks_seq`) as a number of rows (resp. columns) for the block equal to 100. It’s possible, after the construction of the array, to replace these values by the correct ones (since we know the number of rows and columns of the array).

```
2393 \cs_new_protected:Npn \@@_adjust_pos_of_blocks_seq:
2394 {
2395     \seq_gset_map_x:NNn \g_@@_pos_of_blocks_seq \g_@@_pos_of_blocks_seq
2396     { \@@_adjust_pos_of_blocks_seq_i:nnnn #1 }
2397 }
```

The following command must *not* be protected.

```
2398 \cs_new:Npn \@@_adjust_pos_of_blocks_seq_i:nnnn #1 #2 #3 #4
2399 {
2400     { #1 }
2401     { #2 }
2402     {
2403         \int_compare:nNnTF { #3 } > { 99 }
2404         { \int_use:N \c@iRow }
2405         { #3 }
2406     }
2407     {
2408         \int_compare:nNnTF { #4 } > { 99 }
2409         { \int_use:N \c@jCol }
2410         { #4 }
2411     }
2412 }
```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”. That’s why we have to define the adequate version of `\@@_draw_dotted_lines`: whether Tikz is loaded or not (in that case, only PGF is loaded).

```
2413 \AtBeginDocument
2414 {
2415     \cs_new_protected:Npx \@@_draw_dotted_lines:
2416     {
2417         \c_@@_pgfortikzpicture_tl
2418         \@@_draw_dotted_lines_i:
2419         \c_@@_endpgfortikzpicture_tl
2420     }
2421 }
```

The following command *must* be protected because it will appear in the construction of the command `\@@_draw_dotted_lines`.

```
2422 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2423 {
2424     \pgfrememberpicturepositiononpagetrue
2425     \pgf@relevantforpicturesizefalse
2426     \g_@@_HVdotsfor_lines_tl
2427     \g_@@_Vdots_lines_tl
2428     \g_@@_Ddots_lines_tl
2429     \g_@@_Iddots_lines_tl
2430     \g_@@_Cdots_lines_tl
2431     \g_@@_Ldots_lines_tl
2432 }
```



```

2433 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2434 {
2435   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2436   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2437 }

```

The following command creates the diagonal nodes (in fact, if the matrix is not a square matrix, not all the nodes are on the diagonal).

```

2438 \cs_new_protected:Npn \@@_create_diag_nodes:
2439 {
2440   \int_step_inline:nn { \int_max:nn \c@iRow \c@jCol + 1 }
2441   {
2442     \pgfpicture
2443     \pgfrememberpicturepositiononpagetrue
2444     \@@_qpoint:n { row - \int_min:nn { ##1 } { \c@iRow + 1 } }
2445     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2446     \@@_qpoint:n { col - \int_min:nn { ##1 } { \c@jCol + 1 } }
2447     \pgfcoordinate { \@@_env: - ##1 } { \pgfpoint \pgf@x \l_tmpa_dim }
2448     \endpgfpicture
2449   }
2450 }

```

## We draw the dotted lines

A dotted line will be said *open* in one of its extremities when it stops on the edge of the matrix and *closed* otherwise. In the following matrix, the dotted line is closed on its left extremity and open on its right.

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

The command `\@@_find_extremities_of_line:nnnn` takes four arguments:

- the first argument is the row of the cell where the command was issued;
- the second argument is the column of the cell where the command was issued;
- the third argument is the  $x$ -value of the orientation vector of the line;
- the fourth argument is the  $y$ -value of the orientation vector of the line.

This command computes:

- `\l_@@_initial_i_int` and `\l_@@_initial_j_int` which are the coordinates of one extremity of the line;
- `\l_@@_final_i_int` and `\l_@@_final_j_int` which are the coordinates of the other extremity of the line;
- `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` to indicate whether the extremities are open or not.

```

2451 \cs_new_protected:Npn \@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2452 {

```

First, we declare the current cell as “dotted” because we forbid intersections of dotted lines.

```

2453   \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }

```

Initialization of variables.

```

2454   \int_set:Nn \l_@@_initial_i_int { #1 }
2455   \int_set:Nn \l_@@_initial_j_int { #2 }
2456   \int_set:Nn \l_@@_final_i_int { #1 }
2457   \int_set:Nn \l_@@_final_j_int { #2 }

```

We will do two loops: one when determinating the initial cell and the other when determinating the final cell. The boolean `\l_@@_stop_loop_bool` will be used to control these loops. In the first loop, we search the “final” extremity of the line.

```

2458     \bool_set_false:N \l_@@_stop_loop_bool
2459     \bool_do_until:Nn \l_@@_stop_loop_bool
2460     {
2461         \int_add:Nn \l_@@_final_i_int { #3 }
2462         \int_add:Nn \l_@@_final_j_int { #4 }

```

We test if we are still in the matrix.

```

2463     \bool_set_false:N \l_@@_final_open_bool
2464     \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
2465     {
2466         \int_compare:nNnTF { #3 } = 1
2467         { \bool_set_true:N \l_@@_final_open_bool }
2468         {
2469             \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2470             { \bool_set_true:N \l_@@_final_open_bool }
2471         }
2472     }
2473     {
2474         \int_compare:nNnTF \l_@@_final_j_int < 1
2475         {
2476             \int_compare:nNnT { #4 } = { -1 }
2477             { \bool_set_true:N \l_@@_final_open_bool }
2478         }
2479         {
2480             \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2481             {
2482                 \int_compare:nNnT { #4 } = 1
2483                 { \bool_set_true:N \l_@@_final_open_bool }
2484             }
2485         }
2486     }
2487     \bool_if:NTF \l_@@_final_open_bool

```

If we are outside the matrix, we have found the extremity of the dotted line and it’s an *open* extremity.

```

2488     {

```

We do a step backwards.

```

2489         \int_sub:Nn \l_@@_final_i_int { #3 }
2490         \int_sub:Nn \l_@@_final_j_int { #4 }
2491         \bool_set_true:N \l_@@_stop_loop_bool
2492     }

```

If we are in the matrix, we test whether the cell is empty. If it’s not the case, we stop the loop because we have found the correct values for `\l_@@_final_i_int` and `\l_@@_final_j_int`.

```

2493     {
2494         \cs_if_exist:cTF
2495         {
2496             @@ _ dotted _
2497             \int_use:N \l_@@_final_i_int -
2498             \int_use:N \l_@@_final_j_int
2499         }
2500         {
2501             \int_sub:Nn \l_@@_final_i_int { #3 }
2502             \int_sub:Nn \l_@@_final_j_int { #4 }
2503             \bool_set_true:N \l_@@_final_open_bool
2504             \bool_set_true:N \l_@@_stop_loop_bool
2505         }
2506     }
2507     \cs_if_exist:cTF
2508     {
2509         pgf @ sh @ ns @ \@@_env:
2510         - \int_use:N \l_@@_final_i_int

```

```

2511         - \int_use:N \l_@@_final_j_int
2512     }
2513     { \bool_set_true:N \l_@@_stop_loop_bool }

```

If the case is empty, we declare that the cell as non-empty. Indeed, we will draw a dotted line and the cell will be on that dotted line. All the cells of a dotted line have to be marked as “dotted” because we don’t want intersections between dotted lines. We recall that the research of the extremities of the lines are all done in the same TeX group (the group of the environnement), even though, when the extremities are found, each line is drawn in a TeX group that we will open for the options of the line.

```

2514     {
2515         \cs_set:cpn
2516         {
2517             @@ _ dotted _
2518             \int_use:N \l_@@_final_i_int -
2519             \int_use:N \l_@@_final_j_int
2520         }
2521         { }
2522     }
2523 }
2524 }
2525 }

```

For `\l_@@_initial_i_int` and `\l_@@_initial_j_int` the programming is similar to the previous one.

```

2526 \bool_set_false:N \l_@@_stop_loop_bool
2527 \bool_do_until:Nn \l_@@_stop_loop_bool
2528 {
2529     \int_sub:Nn \l_@@_initial_i_int { #3 }
2530     \int_sub:Nn \l_@@_initial_j_int { #4 }
2531     \bool_set_false:N \l_@@_initial_open_bool
2532     \int_compare:nNnTF \l_@@_initial_i_int < 1
2533     {
2534         \int_compare:nNnTF { #3 } = 1
2535         { \bool_set_true:N \l_@@_initial_open_bool }
2536         {
2537             \int_compare:nNnT \l_@@_initial_j_int = 0
2538             { \bool_set_true:N \l_@@_initial_open_bool }
2539         }
2540     }
2541     {
2542         \int_compare:nNnTF \l_@@_initial_j_int < 1
2543         {
2544             \int_compare:nNnT { #4 } = 1
2545             { \bool_set_true:N \l_@@_initial_open_bool }
2546         }
2547         {
2548             \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
2549             {
2550                 \int_compare:nNnT { #4 } = { -1 }
2551                 { \bool_set_true:N \l_@@_initial_open_bool }
2552             }
2553         }
2554     }
2555     \bool_if:NTF \l_@@_initial_open_bool
2556     {
2557         \int_add:Nn \l_@@_initial_i_int { #3 }
2558         \int_add:Nn \l_@@_initial_j_int { #4 }
2559         \bool_set_true:N \l_@@_stop_loop_bool
2560     }
2561     {
2562         \cs_if_exist:cTF

```

```

2563     {
2564         @@ _ dotted _
2565         \int_use:N \l_@@_initial_i_int -
2566         \int_use:N \l_@@_initial_j_int
2567     }
2568     {
2569         \int_add:Nn \l_@@_initial_i_int { #3 }
2570         \int_add:Nn \l_@@_initial_j_int { #4 }
2571         \bool_set_true:N \l_@@_initial_open_bool
2572         \bool_set_true:N \l_@@_stop_loop_bool
2573     }
2574     {
2575         \cs_if_exist:cTF
2576         {
2577             pgf @ sh @ ns @ \@@_env:
2578             - \int_use:N \l_@@_initial_i_int
2579             - \int_use:N \l_@@_initial_j_int
2580         }
2581         { \bool_set_true:N \l_@@_stop_loop_bool }
2582         {
2583             \cs_set:cpn
2584             {
2585                 @@ _ dotted _
2586                 \int_use:N \l_@@_initial_i_int -
2587                 \int_use:N \l_@@_initial_j_int
2588             }
2589             { }
2590         }
2591     }
2592 }
2593 }

```

We remind the rectangle described by all the dotted lines in order to respect the corresponding virtual “block” when drawing the horizontal and vertical rules.

```

2594     \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2595     {
2596         { \int_use:N \l_@@_initial_i_int }
2597         { \int_use:N \l_@@_initial_j_int }
2598         { \int_use:N \l_@@_final_i_int }
2599         { \int_use:N \l_@@_final_j_int }
2600     }
2601 }
2602 \cs_new_protected:Npn \@@_set_initial_coords:
2603 {
2604     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2605     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2606 }
2607 \cs_new_protected:Npn \@@_set_final_coords:
2608 {
2609     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2610     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2611 }
2612 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2613 {
2614     \pgfpointanchor
2615     {
2616         \@@_env:
2617         - \int_use:N \l_@@_initial_i_int
2618         - \int_use:N \l_@@_initial_j_int
2619     }
2620     { #1 }
2621     \@@_set_initial_coords:
2622 }

```

```

2623 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2624 {
2625   \pgfpointanchor
2626   {
2627     \@@_env:
2628     - \int_use:N \l_@@_final_i_int
2629     - \int_use:N \l_@@_final_j_int
2630   }
2631   { #1 }
2632   \@@_set_final_coords:
2633 }

```

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

2634 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
2635 {
2636   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2637   {
2638     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 0 1

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

2639   \group_begin:
2640   \int_compare:nNnTF { #1 } = 0
2641     { \color { nicematrix-first-row } }
2642     {

```

We remind that, when there is a “last row” `\l_@@_last_row_int` will always be (after the construction of the array) the number of that “last row” even if the option `last-row` has been used without value.

```

2643       \int_compare:nNnT { #1 } = \l_@@_last_row_int
2644       { \color { nicematrix-last-row } }
2645     }
2646     \keys_set:nn { NiceMatrix / xdots } { #3 }
2647     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2648     \@@_actually_draw_Ldots:
2649   \group_end:
2650 }
2651 }

```

The command `\@@_actually_draw_Ldots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool`.

The following function is also used by `\Hdotsfor`.

```

2652 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2653 {
2654   \bool_if:NTF \l_@@_initial_open_bool
2655   {
2656     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2657     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2658     \dim_add:Nn \l_@@_x_initial_dim \col@sep
2659     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2660     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2661   }

```

```

2662     { \@@_set_initial_coords_from_anchor:n { base-east } }
2663   \bool_if:NTF \l_@@_final_open_bool
2664   {
2665     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2666     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2667     \dim_sub:Nn \l_@@_x_final_dim \col@sep
2668     \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2669     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2670   }
2671   { \@@_set_final_coords_from_anchor:n { base-west } }

```

We raise the line of a quantity equal to the radius of the dots because we want the dots really “on” the line of texte. Of course, maybe we should not do that when the option `line-style` is used (?).

```

2672     \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2673     \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2674     \@@_draw_line:
2675   }

```

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

2676 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2677 {
2678   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2679   {
2680     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 0 1

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

2681     \group_begin:
2682     \int_compare:nNnTF { #1 } = 0
2683     { \color { nicematrix-first-row } }
2684     {

```

We remind that, when there is a “last row” `\l_@@_last_row_int` will always be (after the construction of the array) the number of that “last row” even if the option `last-row` has been used without value.

```

2685         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2686         { \color { nicematrix-last-row } }
2687     }
2688     \keys_set:nn { NiceMatrix / xdots } { #3 }
2689     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2690     \@@_actually_draw_Cdots:
2691   \group_end:
2692 }
2693 }

```

The command `\@@_actually_draw_Cdots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool`.

```

2694 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2695 {
2696   \bool_if:NTF \l_@@_initial_open_bool
2697   {
2698     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2699     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x

```

```

2700     \dim_add:Nn \l_@@_x_initial_dim \col@sep
2701   }
2702   { \@@_set_initial_coords_from_anchor:n { mid~east } }
2703   \bool_if:NTF \l_@@_final_open_bool
2704   {
2705     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2706     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2707     \dim_sub:Nn \l_@@_x_final_dim \col@sep
2708   }
2709   { \@@_set_final_coords_from_anchor:n { mid~west } }
2710   \bool_lazy_and:nnTF
2711     \l_@@_initial_open_bool
2712     \l_@@_final_open_bool
2713   {
2714     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2715     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2716     \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2717     \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2718     \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2719   }
2720   {
2721     \bool_if:NT \l_@@_initial_open_bool
2722     { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2723     \bool_if:NT \l_@@_final_open_bool
2724     { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2725   }
2726   \@@_draw_line:
2727 }

```

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

2728 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2729 {
2730   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2731   {
2732     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 0

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

2733   \group_begin:
2734     \int_compare:nNnTF { #2 } = 0
2735     { \color { nicematrix-first-col } }
2736     {
2737       \int_compare:nNnT { #2 } = \l_@@_last_col_int
2738       { \color { nicematrix-last-col } }
2739     }
2740     \keys_set:nn { NiceMatrix / xdots } { #3 }
2741     \tl_if_empty:VF \l_@@_xdots_color_tl
2742     { \color { \l_@@_xdots_color_tl } }
2743     \@@_actually_draw_Vdots:
2744   \group_end:
2745 }
2746 }

```

The command `\@@_actually_draw_Vdots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`

- \l\_@@\_final\_open\_bool.

The following function is also used by \Vdotsfor.

```
2747 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2748 {
```

The boolean \l\_tmpa\_bool indicates whether the column is of type l or may be considered as if.

```
2749   \bool_set_false:N \l_tmpa_bool
2750   \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2751   {
2752     \@@_set_initial_coords_from_anchor:n { south~west }
2753     \@@_set_final_coords_from_anchor:n { north~west }
2754     \bool_set:Nn \l_tmpa_bool
2755       { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2756   }
```

Now, we try to determine whether the column is of type c or may be considered as if.

```
2757   \bool_if:NTF \l_@@_initial_open_bool
2758   {
2759     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2760     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2761   }
2762   { \@@_set_initial_coords_from_anchor:n { south } }
2763   \bool_if:NTF \l_@@_final_open_bool
2764   {
2765     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2766     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2767   }
2768   { \@@_set_final_coords_from_anchor:n { north } }
2769   \bool_if:NTF \l_@@_initial_open_bool
2770   {
2771     \bool_if:NTF \l_@@_final_open_bool
2772     {
2773       \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2774       \dim_set_eq:NN \l_tmpa_dim \pgf@x
2775       \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2776       \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2777       \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
```

We may think that the final user won't use a "last column" which contains only a command \Vdots. However, if the \Vdots is in fact used to draw, not a dotted line, but an arrow (to indicate the number of rows of the matrix), it may be really encountered.

```
2778       \int_compare:nNnT \l_@@_last_col_int > { -2 }
2779       {
2780         \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2781         {
2782           \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2783           \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2784           \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2785           \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2786         }
2787       }
2788     }
2789     { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2790   }
2791   {
2792     \bool_if:NTF \l_@@_final_open_bool
2793     { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2794     {
```

Now the case where both extremities are closed. The first conditional tests whether the column is of type c (C of {NiceArray}) or may be considered as if.

```
2795       \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2796       {
2797         \dim_set:Nn \l_@@_x_initial_dim
```



```

2798         {
2799             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2800             \l_@@_x_initial_dim \l_@@_x_final_dim
2801         }
2802         \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2803     }
2804 }
2805 }
2806 \@@_draw_line:
2807 }

```

For the diagonal lines, the situation is a bit more complicated because, by default, we parallelize the diagonals lines. The first diagonal line is drawn and then, all the other diagonal lines are drawn parallel to the first one.

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

2808 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2809 {
2810     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2811     {
2812         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 1

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

2813         \group_begin:
2814         \keys_set:nn { NiceMatrix / xdots } { #3 }
2815         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2816         \@@_actually_draw_Ddots:
2817     \group_end:
2818 }
2819 }

```

The command `\@@_actually_draw_Ddots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool`.

```

2820 \cs_new_protected:Npn \@@_actually_draw_Ddots:
2821 {
2822     \bool_if:NTF \l_@@_initial_open_bool
2823     {
2824         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2825         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2826         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2827         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2828     }
2829     { \@@_set_initial_coords_from_anchor:n { south-east } }
2830     \bool_if:NTF \l_@@_final_open_bool
2831     {
2832         \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2833         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2834         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2835         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2836     }
2837     { \@@_set_final_coords_from_anchor:n { north-west } }

```

We have retrieved the coordinates in the usual way (they are stored in `\l_@@_x_initial_dim`, etc.). If the parallelization of the diagonals is set, we will have (maybe) to adjust the fourth coordinate.

```

2838     \bool_if:NT \l_@@_parallelize_diags_bool
2839     {
2840         \int_gincr:N \g_@@_ddots_int

```

We test if the diagonal line is the first one (the counter `\g_@@_ddots_int` is created for this usage).

```

2841         \int_compare:nNnTF \g_@@_ddots_int = 1

```

If the diagonal line is the first one, we have no adjustment of the line to do but we store the  $\Delta_x$  and the  $\Delta_y$  of the line because these values will be used to draw the others diagonal lines parallels to the first one.

```

2842         {
2843             \dim_gset:Nn \g_@@_delta_x_one_dim
2844             { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2845             \dim_gset:Nn \g_@@_delta_y_one_dim
2846             { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2847         }

```

If the diagonal line is not the first one, we have to adjust the second extremity of the line by modifying the coordinate `\l_@@_x_initial_dim`.

```

2848         {
2849             \dim_set:Nn \l_@@_y_final_dim
2850             {
2851                 \l_@@_y_initial_dim +
2852                 ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2853                 \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
2854             }
2855         }
2856     }
2857     \@@_draw_line:
2858 }

```

We draw the `\Iddots` diagonals in the same way.

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

2859 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2860 {
2861     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2862     {
2863         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

2864         \group_begin:
2865         \keys_set:nn { NiceMatrix / xdots } { #3 }
2866         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2867         \@@_actually_draw_Iddots:
2868         \group_end:
2869     }
2870 }

```

The command `\@@_actually_draw_Iddots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`

- `\l_@@_final_open_bool`.

```

2871 \cs_new_protected:Npn \@@_actually_draw_iddots:
2872 {
2873   \bool_if:NTF \l_@@_initial_open_bool
2874   {
2875     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2876     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2877     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2878     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2879   }
2880   { \@@_set_initial_coords_from_anchor:n { south-west } }
2881   \bool_if:NTF \l_@@_final_open_bool
2882   {
2883     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2884     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2885     \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2886     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2887   }
2888   { \@@_set_final_coords_from_anchor:n { north-east } }
2889   \bool_if:NT \l_@@_parallelize_diags_bool
2890   {
2891     \int_gincr:N \g_@@_iddots_int
2892     \int_compare:nNnTF \g_@@_iddots_int = 1
2893     {
2894       \dim_gset:Nn \g_@@_delta_x_two_dim
2895       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2896       \dim_gset:Nn \g_@@_delta_y_two_dim
2897       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2898     }
2899     {
2900       \dim_set:Nn \l_@@_y_final_dim
2901       {
2902         \l_@@_y_initial_dim +
2903         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2904         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2905       }
2906     }
2907   }
2908   \@@_draw_line:
2909 }

```

## The actual instructions for drawing the dotted line with Tikz

The command `\@@_draw_line:` should be used in a `{pgfpicture}`. It has six implicit arguments:

- `\l_@@_x_initial_dim`
- `\l_@@_y_initial_dim`
- `\l_@@_x_final_dim`
- `\l_@@_y_final_dim`
- `\l_@@_initial_open_bool`
- `\l_@@_final_open_bool`

```

2910 \cs_new_protected:Npn \@@_draw_line:
2911 {
2912   \pgfrememberpicturepositiononpagetrue
2913   \pgf@relevantforpicturesizefalse
2914   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl

```

```

2915 \@@_draw_standard_dotted_line:
2916 \@@_draw_non_standard_dotted_line:
2917 }

```

We have to do a special construction with `\exp_args:NV` to be able to put in the list of options in the correct place in the Tikz instruction.

```

2918 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2919 {
2920   \begin { scope }
2921   \exp_args:No \@@_draw_non_standard_dotted_line:n
2922     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2923 }

```

We have used the fact that, in PGF, un color name can be put directly in a list of options (that's why we have put directly `\l_@@_xdots_color_tl`).

The argument of `\@@_draw_non_standard_dotted_line:n` is, in fact, the list of options.

```

2924 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2925 {
2926   \@@_draw_non_standard_dotted_line:nVV
2927   { #1 }
2928   \l_@@_xdots_up_tl
2929   \l_@@_xdots_down_tl
2930 }
2931 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:nnn #1 #2 #3
2932 {
2933   \draw
2934   [
2935     #1 ,
2936     shorten-> = \l_@@_xdots_shorten_dim ,
2937     shorten-< = \l_@@_xdots_shorten_dim ,
2938   ]
2939   ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )

```

Be careful: We can't put `\c_math_toggle_token` instead of `$` in the following lines because we are in the contents of Tikz nodes (and they will be *rescanned* if the Tikz library `babel` is loaded).

```

2940   -- node [ sloped , above ] { $ \scriptstyle #2 $ }
2941   node [ sloped , below ] { $ \scriptstyle #3 $ }
2942   ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2943   \end { scope }
2944 }
2945 \cs_generate_variant:Nn \@@_draw_non_standard_dotted_line:nnn { n V V }

```

The command `\@@_draw_standard_dotted_line:` draws the line with our system of points (which gives a dotted line with real round points).

```

2946 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2947 {
2948   \bool_lazy_and:nnF
2949     { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2950     { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2951   {
2952     \pgfscope
2953     \pgftransformshift
2954     {
2955       \pgfpointlineattime { 0.5 }
2956       { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2957       { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2958     }
2959     \pgftransformrotate
2960     {
2961       \fp_eval:n
2962       {
2963         atand

```

```

2964         (
2965             \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2966             \l_@@_x_final_dim - \l_@@_x_initial_dim
2967         )
2968     }
2969 }
2970 \pgfnode
2971 { rectangle }
2972 { south }
2973 {
2974     \c_math_toggle_token
2975     \scriptstyle \l_@@_xdots_up_tl
2976     \c_math_toggle_token
2977 }
2978 { }
2979 { \pgfusepath { } }
2980 \pgfnode
2981 { rectangle }
2982 { north }
2983 {
2984     \c_math_toggle_token
2985     \scriptstyle \l_@@_xdots_down_tl
2986     \c_math_toggle_token
2987 }
2988 { }
2989 { \pgfusepath { } }
2990 \endpgfscope
2991 }
2992 \group_begin:

```

The dimension `\l_@@_l_dim` is the length  $\ell$  of the line to draw. We use the floating point reals of `expl3` to compute this length.

```

2993     \dim_zero_new:N \l_@@_l_dim
2994     \dim_set:Nn \l_@@_l_dim
2995     {
2996         \fp_to_dim:n
2997         {
2998             sqrt
2999             (
3000                 ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
3001                 +
3002                 ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
3003             )
3004         }
3005     }

```

It seems that, during the first compilations, the value of `\l_@@_l_dim` may be erroneous (equal to zero or very large). We must detect these cases because they would cause errors during the drawing of the dotted line. Maybe we should also write something in the `aux` file to say that one more compilation should be done.

```

3006     \bool_lazy_or:nnF
3007     { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
3008     { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
3009     \@@_draw_standard_dotted_line_i:
3010 \group_end:
3011 }
3012 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
3013 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
3014 {

```

The integer `\l_tmpa_int` is the number of dots of the dotted line.

```

3015     \bool_if:NTF \l_@@_initial_open_bool
3016     {

```

```

3017 \bool_if:NTF \l_@@_final_open_bool
3018 {
3019   \int_set:Nn \l_tmpa_int
3020   { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
3021 }
3022 {
3023   \int_set:Nn \l_tmpa_int
3024   {
3025     \dim_ratio:nn
3026     { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
3027     \l_@@_inter_dots_dim
3028   }
3029 }
3030 }
3031 {
3032   \bool_if:NTF \l_@@_final_open_bool
3033   {
3034     \int_set:Nn \l_tmpa_int
3035     {
3036       \dim_ratio:nn
3037       { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
3038       \l_@@_inter_dots_dim
3039     }
3040   }
3041   {
3042     \int_set:Nn \l_tmpa_int
3043     {
3044       \dim_ratio:nn
3045       { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
3046       \l_@@_inter_dots_dim
3047     }
3048   }
3049 }

```

The dimensions `\l_tmpa_dim` and `\l_tmpb_dim` are the coordinates of the vector between two dots in the dotted line.

```

3050 \dim_set:Nn \l_tmpa_dim
3051 {
3052   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3053   \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
3054 }
3055 \dim_set:Nn \l_tmpb_dim
3056 {
3057   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
3058   \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
3059 }

```

The length  $\ell$  is the length of the dotted line. We note  $\Delta$  the length between two dots and  $n$  the number of intervals between dots. We note  $\delta = \frac{1}{2}(\ell - n\Delta)$ . The distance between the initial extremity of the line and the first dot will be equal to  $k \cdot \delta$  where  $k = 0, 1$  or  $2$ . We first compute this number  $k$  in `\l_tmpb_int`.

```

3060 \int_set:Nn \l_tmpb_int
3061 {
3062   \bool_if:NTF \l_@@_initial_open_bool
3063   { \bool_if:NTF \l_@@_final_open_bool 1 0 }
3064   { \bool_if:NTF \l_@@_final_open_bool 2 1 }
3065 }

```

In the loop over the dots, the dimensions `\l_@@_x_initial_dim` and `\l_@@_y_initial_dim` will be used for the coordinates of the dots. But, before the loop, we must move until the first dot.

```

3066 \dim_gadd:Nn \l_@@_x_initial_dim
3067 {
3068   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3069   \dim_ratio:nn

```

```

3070         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3071         { 2 \l_@@_l_dim }
3072     * \l_tmpb_int
3073 }
3074 \dim_gadd:Nn \l_@@_y_initial_dim
3075 {
3076     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
3077     \dim_ratio:nn
3078     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3079     { 2 \l_@@_l_dim }
3080     * \l_tmpb_int
3081 }
3082 \pgf@relevantforpicturesizefalse
3083 \int_step_inline:nnn 0 \l_tmpa_int
3084 {
3085     \pgfpathcircle
3086     { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
3087     { \l_@@_radius_dim }
3088     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
3089     \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
3090 }
3091 \pgfusepathqfill
3092 }

```

## User commands available in the new environments

The commands `\@@_Ldots`, `\@@_Cdots`, `\@@_Vdots`, `\@@_Ddots` and `\@@_Iddots` will be linked to `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` and `\Iddots` in the environments `{NiceArray}` (the other environments of `nicematrix` rely upon `{NiceArray}`).

The starred versions of these commands are deprecated since version 3.1 but, as of now, they are still available with an error.

The syntax of these commands uses the character `_` as embellishment and that's why we have to insert a character `_` in the *arg spec* of these commands. However, we don't know the future catcode of `_` in the main document (maybe the user will use `underscore`, and, in that case, the catcode is 13 because `underscore` activates `_`). That's why these commands will be defined in a `\AtBeginDocument` and the *arg spec* will be rescanned.

```

3093 \AtBeginDocument
3094 {
3095     \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
3096     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3097     \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
3098     {
3099         \int_compare:nNnTF \c@jCol = 0
3100         { \@@_error:nn { in~first~col } \Ldots }
3101         {
3102             \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3103             { \@@_error:nn { in~last~col } \Ldots }
3104             {
3105                 \@@_instruction_of_type:nnn \c_false_bool { \Ldots }
3106                 { #1 , down = #2 , up = #3 }
3107             }
3108         }
3109         \bool_if:NF \l_@@_nullify_dots_bool
3110         { \phantom { \ensuremath { \@@_old_ldots } } }
3111         \bool_gset_true:N \g_@@_empty_cell_bool
3112     }

```

```

3113 \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
3114 {
3115   \int_compare:nNnTF \c@jCol = 0
3116   { \@@_error:nn { in~first~col } \Cdots }
3117   {
3118     \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3119     { \@@_error:nn { in~last~col } \Cdots }
3120     {
3121       \@@_instruction_of_type:nnn \c_false_bool { Cdots }
3122       { #1 , down = #2 , up = #3 }
3123     }
3124   }
3125   \bool_if:NF \l_@@_nullify_dots_bool
3126   { \phantom { \ensuremath { \@@_old_cdots } } }
3127   \bool_gset_true:N \g_@@_empty_cell_bool
3128 }

3129 \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
3130 {
3131   \int_compare:nNnTF \c@iRow = 0
3132   { \@@_error:nn { in~first~row } \Vdots }
3133   {
3134     \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
3135     { \@@_error:nn { in~last~row } \Vdots }
3136     {
3137       \@@_instruction_of_type:nnn \c_false_bool { Vdots }
3138       { #1 , down = #2 , up = #3 }
3139     }
3140   }
3141   \bool_if:NF \l_@@_nullify_dots_bool
3142   { \phantom { \ensuremath { \@@_old_vdots } } }
3143   \bool_gset_true:N \g_@@_empty_cell_bool
3144 }

3145 \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
3146 {
3147   \int_case:nnF \c@iRow
3148   {
3149     0 { \@@_error:nn { in~first~row } \Ddots }
3150     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
3151   }
3152   {
3153     \int_case:nnF \c@jCol
3154     {
3155       0 { \@@_error:nn { in~first~col } \Ddots }
3156       \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
3157     }
3158     {
3159       \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3160       \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Ddots }
3161       { #1 , down = #2 , up = #3 }
3162     }
3163   }
3164 }
3165 \bool_if:NF \l_@@_nullify_dots_bool
3166 { \phantom { \ensuremath { \@@_old_ddots } } }
3167 \bool_gset_true:N \g_@@_empty_cell_bool
3168 }

3169 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
3170 {

```



```

3171 \int_case:nnF \c@iRow
3172 {
3173     0 { \@@_error:nn { in~first~row } \Iddots }
3174     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
3175 }
3176 {
3177     \int_case:nnF \c@jCol
3178     {
3179         0 { \@@_error:nn { in~first~col } \Iddots }
3180         \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
3181     }
3182     {
3183         \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3184         \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Iddots }
3185         { #1 , down = #2 , up = #3 }
3186     }
3187 }
3188 \bool_if:NF \l_@@_nullify_dots_bool
3189 { \phantom { \ensuremath { \@@_old_iddots } } }
3190 \bool_gset_true:N \g_@@_empty_cell_bool
3191 }
3192 }

```

End of the `\AtBeginDocument`.

Despite its name, the following set of keys will be used for `\Ddots` but also for `\Iddots`.

```

3193 \keys_define:nn { NiceMatrix / Ddots }
3194 {
3195     draw-first .bool_set:N = \l_@@_draw_first_bool ,
3196     draw-first .default:n = true ,
3197     draw-first .value_forbidden:n = true
3198 }

```

The command `\@@_Hspace:` will be linked to `\hspace` in `{NiceArray}`.

```

3199 \cs_new_protected:Npn \@@_Hspace:
3200 {
3201     \bool_gset_true:N \g_@@_empty_cell_bool
3202     \hspace
3203 }

```

In the environment `{NiceArray}`, the command `\multicolumn` will be linked to the following command `\@@_multicolumn:nnn`.

```

3204 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
3205 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
3206 {

```

We have to act in an expandable way since it will begin by a `\multicolumn`.

```

3207 \exp_args:NNe
3208 \@@_old_multicolumn
3209 { #1 }

```

We will have to replace `\tl_lower_case:n` in the future since it seems to be deprecated.

```

3210 {
3211     \exp_args:Ne \str_case:nn { \str_foldcase:n { #2 } }
3212     {
3213         l { > \@@_Cell: l < \@@_end_Cell: }
3214         r { > \@@_Cell: r < \@@_end_Cell: }
3215         c { > \@@_Cell: c < \@@_end_Cell: }
3216         { l | } { > \@@_Cell: l < \@@_end_Cell: | }
3217         { r | } { > \@@_Cell: r < \@@_end_Cell: | }
3218         { c | } { > \@@_Cell: c < \@@_end_Cell: | }
3219         { | l } { | > \@@_Cell: l < \@@_end_Cell: }
3220         { | r } { | > \@@_Cell: r < \@@_end_Cell: }

```

```

3221         { | c } { | > \@@_Cell: c < \@@_end_Cell: }
3222         { | l | } { | > \@@_Cell: l < \@@_end_Cell: | }
3223         { | r | } { | > \@@_Cell: r < \@@_end_Cell: | }
3224         { | c | } { | > \@@_Cell: c < \@@_end_Cell: | }
3225     }
3226 }
3227 { #3 }

```

The `\peek_remove_spaces:n` is mandatory.

```

3228 \peek_remove_spaces:n
3229 {
3230     \int_compare:nNnT #1 > 1
3231     {
3232         \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
3233         { \int_use:N \c@iRow - \int_use:N \c@jCol }
3234         \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
3235         \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
3236         {
3237             { \int_use:N \c@iRow }
3238             { \int_use:N \c@jCol }
3239             { \int_use:N \c@iRow }
3240             { \int_eval:n { \c@jCol + #1 - 1 } }
3241         }
3242     }
3243     \int_gadd:Nn \c@jCol { #1 - 1 }
3244     \int_compare:nNnT \c@jCol > \g_@@_col_total_int
3245     { \int_gset_eq:NN \g_@@_col_total_int \c@jCol }
3246 }
3247 }

```

The command `\@@_Hdotsfor` will be linked to `\Hdotsfor` in `{NiceArrayWithDelims}`. Tikz nodes are created also in the implicit cells of the `\Hdotsfor` (maybe we should modify that point).

This command must *not* be protected since it begins with `\multicolumn`.

```

3248 \cs_new:Npn \@@_Hdotsfor:
3249 {
3250     \bool_lazy_and:nnTF
3251     { \int_compare_p:nNn \c@jCol = 0 }
3252     { \int_compare_p:nNn \l_@@_first_col_int = 0 }
3253     {
3254         \bool_if:NTF \g_@@_after_col_zero_bool
3255         {
3256             \multicolumn { 1 } { c } { }
3257             \@@_Hdotsfor_i
3258         }
3259         { \@@_fatal:n { Hdotsfor~in~col~0 } }
3260     }
3261     {
3262         \multicolumn { 1 } { c } { }
3263         \@@_Hdotsfor_i
3264     }
3265 }

```

The command `\@@_Hdotsfor_i` is defined with `\NewDocumentCommand` because it has an optional argument. Note that such a command defined by `\NewDocumentCommand` is protected and that's why we have put the `\multicolumn` before (in the definition of `\@@_Hdotsfor:`).

```

3266 \AtBeginDocument
3267 {
3268     \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3269     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl

```

We don't put `!` before the last optionnal argument for homogeneity with `\Cdots`, etc. which have only one optional argument.

```

3270 \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl

```

```

3271 {
3272   \tl_gput_right:Nx \g_@@_HVDotsfor_lines_tl
3273   {
3274     \@@_Hdotsfor:nnnn
3275     { \int_use:N \c@iRow }
3276     { \int_use:N \c@jCol }
3277     { #2 }
3278     {
3279       #1 , #3 ,
3280       down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3281     }
3282   }
3283   \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }
3284 }
3285 }

```

Enf of \AtBeginDocument.

```

3286 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
3287 {
3288   \bool_set_false:N \l_@@_initial_open_bool
3289   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

3290   \int_set:Nn \l_@@_initial_i_int { #1 }
3291   \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

For the column, it's a bit more complicated.

```

3292   \int_compare:nNnTF #2 = 1
3293   {
3294     \int_set:Nn \l_@@_initial_j_int 1
3295     \bool_set_true:N \l_@@_initial_open_bool
3296   }
3297   {
3298     \cs_if_exist:cTF
3299     {
3300       pgf @ sh @ ns @ \@@_env:
3301       - \int_use:N \l_@@_initial_i_int
3302       - \int_eval:n { #2 - 1 }
3303     }
3304     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
3305     {
3306       \int_set:Nn \l_@@_initial_j_int { #2 }
3307       \bool_set_true:N \l_@@_initial_open_bool
3308     }
3309   }
3310   \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
3311   {
3312     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3313     \bool_set_true:N \l_@@_final_open_bool
3314   }
3315   {
3316     \cs_if_exist:cTF
3317     {
3318       pgf @ sh @ ns @ \@@_env:
3319       - \int_use:N \l_@@_final_i_int
3320       - \int_eval:n { #2 + #3 }
3321     }
3322     { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
3323     {
3324       \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3325       \bool_set_true:N \l_@@_final_open_bool
3326     }
3327   }

```

```

3328 \group_begin:
3329 \int_compare:nNnTF { #1 } = 0
3330 { \color { nicematrix-first-row } }
3331 {
3332   \int_compare:nNnT { #1 } = \g_@@_row_total_int
3333   { \color { nicematrix-last-row } }
3334 }
3335 \keys_set:nn { NiceMatrix / xdots } { #4 }
3336 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3337 \@@_actually_draw_Ldots:
3338 \group_end:

```

We declare all the cells concerned by the `\Hdotsfor` as “dotted” (for the dotted lines created by `\Cdots`, `\Ldots`, etc., this job is done by `\@@_find_extremities_of_line:nnnn`). This declaration is done by defining a special control sequence (to nil).

```

3339 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
3340 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
3341 }

3342 \AtBeginDocument
3343 {
3344   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3345   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3346   \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
3347   {
3348     \tl_gput_right:Nx \g_@@_HVDotsfor_lines_tl
3349     {
3350       \@@_Vdotsfor:nnnn
3351       { \int_use:N \c@iRow }
3352       { \int_use:N \c@jCol }
3353       { #2 }
3354       {
3355         #1 , #3 ,
3356         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3357       }
3358     }
3359   }
3360 }

```

End of `\AtBeginDocument`.

```

3361 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
3362 {
3363   \bool_set_false:N \l_@@_initial_open_bool
3364   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it’s easy.

```

3365 \int_set:Nn \l_@@_initial_j_int { #2 }
3366 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

For the row, it’s a bit more complicated.

```

3367 \int_compare:nNnTF #1 = 1
3368 {
3369   \int_set:Nn \l_@@_initial_i_int 1
3370   \bool_set_true:N \l_@@_initial_open_bool
3371 }
3372 {
3373   \cs_if_exist:cTF
3374   {
3375     pgf @ sh @ ns @ \@@_env:
3376     - \int_eval:n { #1 - 1 }
3377     - \int_use:N \l_@@_initial_j_int
3378   }
3379   { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }

```

```

3380         {
3381             \int_set:Nn \l_@@_initial_i_int { #1 }
3382             \bool_set_true:N \l_@@_initial_open_bool
3383         }
3384     }
3385     \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
3386     {
3387         \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3388         \bool_set_true:N \l_@@_final_open_bool
3389     }
3390     {
3391         \cs_if_exist:cTF
3392         {
3393             pgf @ sh @ ns @ \@@_env:
3394             - \int_eval:n { #1 + #3 }
3395             - \int_use:N \l_@@_final_j_int
3396         }
3397         { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
3398         {
3399             \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3400             \bool_set_true:N \l_@@_final_open_bool
3401         }
3402     }
3403     \group_begin:
3404     \int_compare:nNnTF { #2 } = 0
3405     { \color { nicematrix-first-col } }
3406     {
3407         \int_compare:nNnT { #2 } = \g_@@_col_total_int
3408         { \color { nicematrix-last-col } }
3409     }
3410     \keys_set:nn { NiceMatrix / xdots } { #4 }
3411     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3412     \@@_actually_draw_Vdots:
3413     \group_end:

```

We declare all the cells concerned by the `\Vdotsfor` as “dotted” (for the dotted lines created by `\Cdots`, `\Ldots`, etc., this job is done by `\@@_find_extremities_of_line:nnnn`). This declaration is done by defining a special control sequence (to nil).

```

3414     \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
3415     { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
3416 }

```

The command `\@@_rotate:` will be linked to `\rotate` in `{NiceArrayWithDelims}`.

```

3417 \cs_new_protected:Npn \@@_rotate: { \bool_gset_true:N \g_@@_rotate_bool }

```

## The command `\line` accessible in code-after

In the `\CodeAfter`, the command `\@@_line:nn` will be linked to `\line`. This command takes two arguments which are the specifications of two cells in the array (in the format  $i-j$ ) and draws a dotted line between these cells.

First, we write a command with an argument of the format  $i-j$  and applies the command `\int_eval:n` to  $i$  and  $j$ ; this must *not* be protected (and is, of course fully expandable).<sup>52</sup>

```

3418 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
3419 { \int_eval:n { #1 } - \int_eval:n { #2 } }

```

---

<sup>52</sup>Indeed, we want that the user may use the command `\line` in `\CodeAfter` with LaTeX counters in the arguments — with the command `\value`.

With the following construction, the command `\@@_double_int_eval:n` is applied to both arguments before the application of `\@@_line_i:nn` (the construction uses the fact the `\@@_line_i:nn` is protected and that `\@@_double_int_eval:n` is fully expandable).

```

3420 \AtBeginDocument
3421 {
3422   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
3423   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3424   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
3425     {
3426       \group_begin:
3427       \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
3428       \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3429       \use:e
3430       {
3431         \@@_line_i:nn
3432         { \@@_double_int_eval:n #2 \q_stop }
3433         { \@@_double_int_eval:n #3 \q_stop }
3434       }
3435       \group_end:
3436     }
3437 }
3438 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3439 {
3440   \bool_set_false:N \l_@@_initial_open_bool
3441   \bool_set_false:N \l_@@_final_open_bool
3442   \bool_if:nTF
3443     {
3444       \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3445       ||
3446       \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3447     }
3448     {
3449       \@@_error:nnn { unknown~cell~for~line~in~CodeAfter } { #1 } { #2 }
3450     }
3451     { \@@_draw_line_ii:nn { #1 } { #2 } }
3452 }
3453 \AtBeginDocument
3454 {
3455   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3456   {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible” and that why we do this static construction of the command `\@@_draw_line_ii:`.

```

3457   \c_@@_pgfortikzpicture_tl
3458   \@@_draw_line_iii:nn { #1 } { #2 }
3459   \c_@@_endpgfortikzpicture_tl
3460 }
3461 }

```

The following command *must* be protected (it’s used in the construction of `\@@_draw_line_ii:nn`).

```

3462 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3463 {
3464   \pgfrememberpicturepositiononpagetrue
3465   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3466   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3467   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3468   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3469   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3470   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3471   \@@_draw_line:
3472 }

```

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots` don't use this command because they have to do other settings (for example, the diagonal lines must be parallelized).

## Colors of cells, rows and columns

We want to avoid the thin white lines that are shown in some PDF viewers (eg: with the engine MuPDF used by SumatraPDF). That's why we try to draw rectangles of the same color in the same instruction `\pgfusepathqfill` (and they will be in the same instruction `fill`—coded `f`— in the resulting PDF).

The commands `\@@_rowcolor`, `\@@_columncolor` and `\@@_rectanglecolor` (which are linked to `\rowcolor`, `\columncolor` and `\rectanglecolor` before the execution of the `code-before`) don't directly draw the corresponding rectangles. Instead, they store their instructions color by color:

- A sequence `\l_@@_colors_seq` will be built containing all the colors used by at least one of these instructions. Each *color* may be prefixed by its color model (eg: `[gray]{0.5}`).
- For the color whose index in `\l_@@_colors_seq` is equal to *i*, a list of instructions which use that color will be constructed in the token list `\l_@@_color_i_tl`. In that token list, the instructions will `\@@_rowcolor:n`, `\@@_columncolor:n` and `\@@_rectanglecolor:nn` (corresponding of `\@@_rowcolor`, `\@@_columncolor` and `\@@_rectanglecolor`).

`bigskip #1` is the color and `#2` is an instruction using that color. Despite its name, the command `\@@_add_to_color_seq` doesn't only add a color to `\l_@@_colors_seq`: it also updates the corresponding token list `\l_@@_color_i_tl`.

```
3473 \cs_new_protected:Npn \@@_add_to_colors_seq:nn #1 #2
3474 {
```

First, we look for the number of the color and, if it's found, we store it in `\l_tmpa_int`. If the color is not present in `\l_@@_colors_seq`, `\l_tmpa_int` will remain equal to 0.

```
3475 \int_zero:N \l_tmpa_int
3476 \seq_map_indexed_inline:Nn \l_@@_colors_seq
3477 { \tl_if_eq:nnT { #1 } { ##2 } { \int_set:Nn \l_tmpa_int { ##1 } } }
3478 \int_compare:nNnTF \l_tmpa_int = \c_zero_int
```

First, the case where the color is a *new* color (not in the sequence).

```
3479 {
3480 \seq_put_right:Nn \l_@@_colors_seq { #1 }
3481 \tl_set:cn { l_@@_color _ \seq_count:N \l_@@_colors_seq _ tl } { #2 }
3482 }
```

Now, the case where the color is *not* a new color (the color is in the sequence at the position `\l_tmpa_int`).

```
3483 { \tl_put_right:cn { l_@@_color _ \int_use:N \l_tmpa_int _tl } { #2 } }
3484 }
```

```
3485 \cs_generate_variant:Nn \@@_add_to_colors_seq:nn { x n }
```

The macro `\@@_actually_color:` will actually fill all the rectangles, color by color (using the sequence `\l_@@_colors_seq` and all the token lists of the form `\l_@@_color_i_tl`).

```
3486 \cs_new_protected:Npn \@@_actually_color:
3487 {
3488 \pgfpicture
3489 \pgf@relevantforpicturesizefalse
3490 \seq_map_indexed_inline:Nn \l_@@_colors_seq
3491 {
3492 \color ##2
3493 \use:c { l_@@_color _ ##1 _tl }
3494 \pgfusepathqfill
3495 }
3496 \endpgfpicture
3497 }
```

```

3498 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3499 {
3500   \tl_set:Nn \l_tmpa_tl { #1 }
3501   \tl_set:Nn \l_tmpb_tl { #2 }
3502 }

```

Here is an example : `\@@_rowcolor {red!15} {1,3,5-7,10-}`

```

3503 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3504 {
3505   \tl_if_blank:nF { #2 }
3506   {
3507     \@@_add_to_colors_seq:xn
3508     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3509     { \@@_rowcolor:n { #3 } }
3510   }
3511 }
3512 \cs_new_protected:Npn \@@_rowcolor:n #1
3513 {
3514   \tl_set:Nn \l_@@_rows_tl { #1 }
3515   \tl_set:Nn \l_@@_cols_tl { - }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3516   \@@_cartesian_path:
3517 }

```

Here an example : `\@@_columncolor:nn {red!15} {1,3,5-7,10-}`

```

3518 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3519 {
3520   \tl_if_blank:nF { #2 }
3521   {
3522     \@@_add_to_colors_seq:xn
3523     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3524     { \@@_columncolor:n { #3 } }
3525   }
3526 }
3527 \cs_new_protected:Npn \@@_columncolor:n #1
3528 {
3529   \tl_set:Nn \l_@@_rows_tl { - }
3530   \tl_set:Nn \l_@@_cols_tl { #1 }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3531   \@@_cartesian_path:
3532 }

```

Here is an example : `\@@_rectanglecolor{red!15}{2-3}{5-6}`

```

3533 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3534 {
3535   \tl_if_blank:nF { #2 }
3536   {
3537     \@@_add_to_colors_seq:xn
3538     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3539     { \@@_rectanglecolor:nn { #3 } { #4 } }
3540   }
3541 }

```



```

3542 \cs_new_protected:Npn \@@_rectanglecolor:nn #1 #2
3543 {
3544   \@@_cut_on_hyphen:w #1 \q_stop
3545   \tl_clear_new:N \l_tmpc_tl
3546   \tl_clear_new:N \l_tmpd_tl
3547   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
3548   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
3549   \@@_cut_on_hyphen:w #2 \q_stop
3550   \tl_set:Nx \l_@@_rows_tl { \l_tmpc_tl - \l_tmpa_tl }
3551   \tl_set:Nx \l_@@_cols_tl { \l_tmpd_tl - \l_tmpb_tl }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3552   \@@_cartesian_path:
3553 }

```

Here is an example : `\@@_cellcolor[rgb]{0.5,0.5,0}{2-3,3-4,4-5,5-6}`

```

3554 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3555 {
3556   \clist_map_inline:nn { #3 }
3557   { \@@_rectanglecolor [ #1 ] { #2 } { ##1 } { ##1 } }
3558 }

```

```

3559 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3560 {
3561   \int_step_inline:nn { \int_use:N \c@iRow }
3562   {
3563     \int_step_inline:nn { \int_use:N \c@jCol }
3564     {
3565       \int_if_even:nTF { ####1 + ##1 }
3566       { \@@_cellcolor [ #1 ] { #2 } }
3567       { \@@_cellcolor [ #1 ] { #3 } }
3568       { ##1 - ####1 }
3569     }
3570   }
3571 }

```

```

3572 \keys_define:nn { NiceMatrix / rowcolors }
3573 {
3574   respect-blocks .bool_set:N = \l_@@_respect_blocks_bool ,
3575   respect-blocks .default:n = true ,
3576   cols .tl_set:N = \l_@@_cols_tl ,
3577   restart .bool_set:N = \l_@@_rowcolors_restart_bool ,
3578   restart .default:n = true ,
3579   unknown .code:n = \@@_error:n { Unknown-key-for-rowcolors }
3580 }

```

The command `\rowcolors` (accessible in the code-before) is inspired by the command `\rowcolors` of the package `xcolor` (with the option `table`). However, the command `\rowcolors` of `nicematrix` has *not* the optional argument of the command `\rowcolors` of `xcolor`. Here is an example: `\rowcolors{1}{blue!10}{}[respect-blocks]`.

**#1** (optional) is the color space ; **#2** is a list of intervals of rows ; **#3** is the first color ; **#4** is the second color ; **#5** is for the optional list of pairs key-value.

```

3581 \NewDocumentCommand \@@_rowcolors { 0 { } m m m 0 { } }
3582 {

```

The group is for the options.

```

3583   \group_begin:
3584   \tl_clear_new:N \l_@@_cols_tl
3585   \tl_set:Nn \l_@@_cols_tl { - }
3586   \keys_set:nn { NiceMatrix / rowcolors } { #5 }

```

The boolean `\l_tmpa_bool` will indicate whereas we are in a row of the first color or of the second color.

```

3587 \bool_set_true:N \l_tmpa_bool
3588 \bool_lazy_and:nnT
3589 \l_@@_respect_blocks_bool
3590 {
3591   \cs_if_exist_p:c
3592   { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq } }
3593 {

```

We don't want to take into account a block which is completely in the “first column” of (number 0) or in the “last column” and that's why we filter the sequence of the blocks (in a the sequence `\l_tmpa_seq`).

```

3594 \seq_set_eq:Nc \l_tmpb_seq
3595 { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3596 \seq_set_filter:Nn \l_tmpa_seq \l_tmpb_seq
3597 { \@@_not_in_exterior_p:nnnn ##1 }
3598 }
3599 \pgfpicture
3600 \pgf@relevantforpicturesizefalse
3601 \clist_map_inline:nn { #2 }
3602 {
3603   \tl_set:Nn \l_tmpa_tl { ##1 }
3604   \tl_if_in:NnTF \l_tmpa_tl { - }
3605   { \@@_cut_on_hyphen:w ##1 \q_stop }
3606   { \tl_set:Nx \l_tmpb_tl { \int_use:N \c{iRow } } }

```

The counter `\l_tmpa_int` will be the index of the loop.

```

3607 \int_set:Nn \l_tmpa_int \l_tmpa_tl
3608 \bool_if:NNTF \l_@@_rowcolors_restart_bool
3609 { \bool_set_true:N \l_tmpa_bool }
3610 { \bool_set:Nn \l_tmpa_bool { \int_if_odd_p:n { \l_tmpa_tl } } }
3611 \int_zero_new:N \l_tmppc_int
3612 \int_set:Nn \l_tmppc_int \l_tmpb_tl
3613 \int_do_until:nNnn \l_tmpa_int > \l_tmppc_int
3614 {

```

We will compute in `\l_tmppb_int` the last row of the “block”.

```

3615 \int_set_eq:NN \l_tmppb_int \l_tmpa_int

```

If the key `respect-blocks` is in force, we have to adjust that value (of course).

```

3616 \bool_lazy_and:nnT
3617 \l_@@_respect_blocks_bool
3618 {
3619   \cs_if_exist_p:c
3620   { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq } }
3621 }
3622 {
3623   \seq_set_filter:Nn \l_tmppb_seq \l_tmpa_seq
3624   { \@@_intersect_our_row_p:nnnn #####1 }
3625   \seq_map_inline:Nn \l_tmppb_seq { \@@_rowcolors_i:nnnn #####1 }

```

Now, the last row of the block is computed in `\l_tmppb_int`.

```

3626 }
3627 \tl_set:Nx \l_@@_rows_tl
3628 { \int_use:N \l_tmpa_int - \int_use:N \l_tmppb_int }
3629 \bool_if:NNTF \l_tmpa_bool
3630 {
3631   \tl_if_blank:nF { #3 }
3632   {
3633     \tl_if_empty:nTF { #1 }
3634     \color
3635     { \color [ #1 ] }
3636     { #3 }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3637         \@@_cartesian_path:
3638         \pgfusepathqfill
3639     }
3640     \bool_set_false:N \l_tmpa_bool
3641 }
3642 {
3643     \tl_if_blank:nF { #4 }
3644     {
3645         \tl_if_empty:nTF { #1 }
3646         \color
3647         { \color [ #1 ] }
3648         { #4 }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3649         \@@_cartesian_path:
3650         \pgfusepathqfill
3651     }
3652     \bool_set_true:N \l_tmpa_bool
3653 }
3654     \int_set:Nn \l_tmpa_int { \l_tmpb_int + 1 }
3655 }
3656 }
3657 \endpgfpicture
3658 \group_end:
3659 }

3660 \cs_new_protected:Npn \@@_rowcolors_i:nnnn #1 #2 #3 #4
3661 {
3662     \int_compare:nNnT { #3 } > \l_tmpb_int
3663     { \int_set:Nn \l_tmpb_int { #3 } }
3664 }

3665 \prg_new_conditional:Nnn \@@_not_in_exterior:nnnn p
3666 {
3667     \bool_lazy_or:nnTF
3668     { \int_compare_p:nNn { #4 } = \c_zero_int }
3669     { \int_compare_p:nNn { #2 } = { \@@_succ:n { \c_jCol } } }
3670     \prg_return_false:
3671     \prg_return_true:
3672 }

```

The following command return true when the block intersects the row `\l_tmpa_int`.

```

3673 \prg_new_conditional:Nnn \@@_intersect_our_row:nnnn p
3674 {
3675     \bool_if:nTF
3676     {
3677         \int_compare_p:n { #1 <= \l_tmpa_int }
3678         &&
3679         \int_compare_p:n { \l_tmpa_int <= #3 }
3680     }
3681     \prg_return_true:
3682     \prg_return_false:
3683 }

```

The following command uses two implicit arguments : `\l_@@_rows_tl` and `\l_@@_cols_tl` which are specifications for a set of rows and a set of columns. It creates a path but does *not* fill it. It must be filled by another command after.

```

3684 \cs_new_protected:Npn \@@_cartesian_path:
3685 {

```

We begin the loop over the columns.

```

3686 \clist_map_inline:Nn \l_@@_cols_tl
3687 {
3688   \tl_set:Nn \l_tmpa_tl { ##1 }
3689   \tl_if_in:NnTF \l_tmpa_tl { - }
3690     { \@@_cut_on_hyphen:w ##1 \q_stop }
3691     { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3692   \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3693   \tl_if_empty:NT \l_tmpb_tl
3694     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3695   \int_compare:nNnT \l_tmpb_tl > \c@jCol
3696     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3697   \@@_qpoint:n { col - \l_tmpa_tl }
3698   \int_compare:nNnTF \l_@@_first_col_int = \l_tmpa_tl
3699     { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3700     { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3701   \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3702   \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }

```

We begin the loop over the rows

```

3703 \clist_map_inline:Nn \l_@@_rows_tl
3704 {
3705   \tl_set:Nn \l_tmpa_tl { ####1 }
3706   \tl_if_in:NnTF \l_tmpa_tl { - }
3707     { \@@_cut_on_hyphen:w ####1 \q_stop }
3708     { \@@_cut_on_hyphen:w ####1 - ####1 \q_stop }
3709   \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3710   \tl_if_empty:NT \l_tmpb_tl
3711     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3712   \int_compare:nNnT \l_tmpb_tl > \c@iRow
3713     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }

```

Now, the numbers of both rows are in \l\_tmpa\_tl and \l\_tmpb\_tl.

```

3714   \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3715   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3716   \@@_qpoint:n { row - \l_tmpa_tl }
3717   \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3718   \pgfpathrectanglecorners
3719     { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3720     { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3721   }
3722 }
3723 }

```

When the user uses the key colortbl-like, the following command will be linked to \cellcolor in the tabular.

```

3724 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
3725 {
3726   \tl_gput_right:Nx \g_nicematrix_code_before_tl
3727     { \cellcolor [ #1 ] { #2 } { \int_use:N \c@iRow - \int_use:N \c@jCol } }
3728 }

```

When the user uses the key colortbll-like, the following command will be linked to \rowcolor in the tabular.

```

3729 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
3730 {
3731   \tl_gput_right:Nx \g_nicematrix_code_before_tl
3732   {
3733     \exp_not:N \rectanglecolor [ #1 ] { #2 }
3734     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3735     { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
3736   }
3737 }

```

```

3738 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
3739 {
3740   \int_compare:nNt \c@iRow = 1
3741   {

```

You use `gput_left` because we want the specification of colors for the columns drawn before the specifications of color for the rows (and the cells).

```

3742     \tl_gput_left:Nx \g_nicematrix_code_before_tl
3743     { \exp_not:N \columncolor [ #1 ] { #2 } { \int_use:N \c@jCol } }
3744   }
3745 }

```

## The vertical rules

We give to the user the possibility to define new types of columns (with `\newcolumnntype` of `array`) for special vertical rules (*e.g.* rules thicker than the standard ones) which will not extend in the potential exterior rows of the array.

We provide the command `\OnlyMainNiceMatrix` in that goal. However, that command must be no-op outside the environments of `nicematrix` (and so the user will be allowed to use the same new type of column in the environments of `nicematrix` and in the standard environments of `array`).

That's why we provide first a global definition of `\OnlyMainNiceMatrix`.

```

3746 \cs_set_eq:NN \OnlyMainNiceMatrix \use:n

```

Another definition of `\OnlyMainNiceMatrix` will be linked to the command in the environments of `nicematrix`. Here is that definition, called `\@@_OnlyMainNiceMatrix:n`.

```

3747 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
3748 {
3749   \int_compare:nNtF \l_@@_first_col_int = 0
3750   { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3751   {
3752     \int_compare:nNtF \c@jCol = 0
3753     {
3754       \int_compare:nNtF \c@iRow = { -1 }
3755       { \int_compare:nNtF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
3756     }
3757     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3758   }
3759 }

```

This definition may seem complicated by we must remind that the number of row `\c@iRow` is incremented in the first cell of the row, *after* a potential vertical rule on the left side of the first cell.

The command `\@@_OnlyMainNiceMatrix_i:n` is only a short-cut which is used twice in the above command. This command must *not* be protected.

```

3760 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
3761 {
3762   \int_compare:nNtF \c@iRow = 0
3763   { \int_compare:nNtF \c@iRow = \l_@@_last_row_int { #1 } }
3764 }

```

Remember that `\c@iRow` is not always inferior to `\l_@@_last_row_int` because `\l_@@_last_row_int` may be equal to `-2` or `-1` (we can't write `\int_compare:nNt \c@iRow < \l_@@_last_row_int`).

The following command will be executed in the `internal-code-after`. The rule will be drawn *before* the column `#1` (that is to say on the left side). `#2` is the number of consecutive occurrences of `|`.

```

3765 \cs_new_protected:Npn \@@_vline:nn #1 #2
3766 {

```

The following test is for the case where the user don't use all the columns specified in the preamble of the environment (for instance, a preamble of `|c|c|c|` but only two columns used).

```

3767 \int_compare:nNtT { #1 } < { \c@jCol + 2 }
3768 {
3769   \pgfpicture
3770   \@@_vline_i:nn { #1 } { #2 }
3771   \endpgfpicture
3772 }
3773 }
3774 \cs_new_protected:Npn \@@_vline_i:nn #1 #2
3775 {

```

`\l_tmpa_tl` is the number of row and `\l_tmpb_tl` the number of column. When we have found a row corresponding to a rule to draw, we note its number in `\l_tmpc_tl`.

```

3776 \tl_set:Nx \l_tmpb_tl { #1 }
3777 \tl_clear_new:N \l_tmpc_tl
3778 \int_step_variable:nNn \c@iRow \l_tmpa_tl
3779 {

```

The boolean `\g_tmpa_bool` indicates whether the small vertical rule will be drawn. If we find that it is in a block (a real block, created by `\Block` or a virtual block corresponding to a dotted line, created by `\Cdots`, `\Vdots`, etc.), we will set `\g_tmpa_bool` to `false` and the small vertical rule won't be drawn.

```

3780 \bool_gset_true:N \g_tmpa_bool
3781 \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3782 { \@@_test_vline_in_block:nnnn ##1 }
3783 \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3784 { \@@_test_vline_in_block:nnnn ##1 }
3785 \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
3786 { \@@_test_vline_in_stroken_block:nnnn ##1 }
3787 \clist_if_empty:NF \l_@@_except_corners_clist
3788 \@@_test_in_corner_v:
3789 \bool_if:NTF \g_tmpa_bool
3790 {
3791   \tl_if_empty:NT \l_tmpc_tl

```

We keep in memory that we have a rule to draw.

```

3792 { \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl }
3793 }
3794 {
3795   \tl_if_empty:NF \l_tmpc_tl
3796   {
3797     \@@_vline_ii:nnnn
3798     { #1 }
3799     { #2 }
3800     \l_tmpc_tl
3801     { \int_eval:n { \l_tmpa_tl - 1 } }
3802     \tl_clear:N \l_tmpc_tl
3803   }
3804 }
3805 }
3806 \tl_if_empty:NF \l_tmpc_tl
3807 {
3808   \@@_vline_ii:nnnn
3809   { #1 }
3810   { #2 }
3811   \l_tmpc_tl
3812   { \int_use:N \c@iRow }
3813   \tl_clear:N \l_tmpc_tl
3814 }
3815 }

```

```

3816 \cs_new_protected:Npn \@@_test_in_corner_v:

```

```

3817 {
3818   \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
3819   {
3820     \seq_if_in:NxT
3821     \l_@@_empty_corner_cells_seq
3822     { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3823     { \bool_set_false:N \g_tmpa_bool }
3824   }
3825   {
3826     \seq_if_in:NxT
3827     \l_@@_empty_corner_cells_seq
3828     { \l_tmpa_tl - \l_tmpb_tl }
3829     {
3830       \int_compare:nNnTF \l_tmpb_tl = 1
3831       { \bool_set_false:N \g_tmpa_bool }
3832       {
3833         \seq_if_in:NxT
3834         \l_@@_empty_corner_cells_seq
3835         { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3836         { \bool_set_false:N \g_tmpa_bool }
3837       }
3838     }
3839   }
3840 }

```

#1 is the number of the column; #2 is the number of vertical rules to draw (with potentially a color between); #3 and #4 are the numbers of the rows between which the rule has to be drawn.

```

3841 \cs_new_protected:Npn \@@_vline_ii:nnnn #1 #2 #3 #4
3842 {
3843   \pgfrememberpicturepositiononpagetrue
3844   \pgf@relevantforpicturesizefalse
3845   \@@_qpoint:n { row - #3 }
3846   \dim_set_eq:NN \l_tmpa_dim \pgf@y
3847   \@@_qpoint:n { col - #1 }
3848   \dim_set_eq:NN \l_tmpb_dim \pgf@x
3849   \@@_qpoint:n { row - \@@_succ:n { #4 } }
3850   \dim_set_eq:NN \l_tmpc_dim \pgf@y
3851   \bool_lazy_and:nnT
3852   { \int_compare_p:nNn { #2 } > 1 }
3853   { ! \tl_if_blank_p:V \CT@drsc@ }
3854   {
3855     \group_begin:
3856     \CT@drsc@
3857     \dim_add:Nn \l_tmpa_dim { 0.5 \arrayrulewidth }
3858     \dim_sub:Nn \l_tmpc_dim { 0.5 \arrayrulewidth }
3859     \dim_set:Nn \l_tmpd_dim
3860     { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3861     \pgfpathrectanglecorners
3862     { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3863     { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
3864     \pgfusepathqfill
3865     \group_end:
3866   }
3867   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3868   \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3869   \prg_replicate:nn { #2 - 1 }
3870   {
3871     \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3872     \dim_sub:Nn \l_tmpb_dim \doublerulesep
3873     \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3874     \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3875   }
3876   \CT@arc@

```

```

3877 \pgfsetlinewidth { 1.1 \arrayrulewidth }
3878 \pgfsetrectcap
3879 \pgfusepathqstroke
3880 }

```

The following command draws a complete vertical rule in the column #1 (#2 is the number of consecutive rules specified by the number of | in the preamble). This command will be used if there is no block in the array (and the key `except-corners` is not used).

```

3881 \cs_new_protected:Npn \@@_vline_i_complete:nn #1 #2
3882 { \@@_vline_ii:nnnn { #1 } { #2 } 1 { \int_use:N \c@iRow } }

```

The command `\@@_draw_hlines:` draws all the vertical rules excepted in the blocks, in the virtual blocks (determined by a command such as `\Cdots`) and in the corners (if the key `except-corners` is used).

```

3883 \cs_new_protected:Npn \@@_draw_vlines:
3884 {
3885   \int_step_inline:nnn
3886     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3887     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3888     { \@@_vline:nn { ##1 } 1 }
3889 }

```

## The horizontal rules

The following command will be executed in the `internal-code-after`. The rule will be drawn *before* the row #1. #2 is the number of consecutive occurrences of `\Hline`.

```

3890 \cs_new_protected:Npn \@@_hline:nn #1 #2
3891 {
3892   \pgfpicture
3893     \@@_hline_i:nn { #1 } { #2 }
3894   \endpgfpicture
3895 }
3896 \cs_new_protected:Npn \@@_hline_i:nn #1 #2
3897 {

```

`\l_tmpa_tl` is the number of row and `\l_tmpb_tl` the number of column. When we have found a column corresponding to a rule to draw, we note its number in `\l_tmpc_tl`.

```

3898   \tl_set:Nn \l_tmpa_tl { #1 }
3899   \tl_clear_new:N \l_tmpc_tl
3900   \int_step_variable:nNn \c@jCol \l_tmpb_tl
3901   {

```

The boolean `\g_tmpa_bool` indicates whether the small horizontal rule will be drawn. If we find that it is in a block (a real block, created by `\Block` or a virtual block corresponding to a dotted line, created by `\Cdots`, `\Vdots`, etc.), we will set `\g_tmpa_bool` to `false` and the small horizontal rule won't be drawn.

```

3902     \bool_gset_true:N \g_tmpa_bool
3903     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3904       { \@@_test_hline_in_block:nnnn ##1 }
3905     \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3906       { \@@_test_hline_in_block:nnnn ##1 }
3907     \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
3908       { \@@_test_hline_in_stroken_block:nnnn ##1 }
3909     \clist_if_empty:NF \l_@@_except_corners_clist \@@_test_in_corner_h:
3910     \bool_if:NTF \g_tmpa_bool
3911     {
3912       \tl_if_empty:NT \l_tmpc_tl

```



We keep in memory that we have a rule to draw.

```

3913         { \tl_set_eq:NN \l_tmpc_tl \l_tmpb_tl }
3914     }
3915     {
3916         \tl_if_empty:NF \l_tmpc_tl
3917         {
3918             \@@_hline_ii:nnnn
3919             { #1 }
3920             { #2 }
3921             \l_tmpc_tl
3922             { \int_eval:n { \l_tmpb_tl - 1 } }
3923             \tl_clear:N \l_tmpc_tl
3924         }
3925     }
3926 }
3927 \tl_if_empty:NF \l_tmpc_tl
3928 {
3929     \@@_hline_ii:nnnn
3930     { #1 }
3931     { #2 }
3932     \l_tmpc_tl
3933     { \int_use:N \c@jCol }
3934     \tl_clear:N \l_tmpc_tl
3935 }
3936 }

3937 \cs_new_protected:Npn \@@_test_in_corner_h:
3938 {
3939     \int_compare:nNnTF \l_tmpa_tl = { \@@_succ:n \c@iRow }
3940     {
3941         \seq_if_in:NxT
3942         \l_@@_empty_corner_cells_seq
3943         { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3944         { \bool_set_false:N \g_tmpa_bool }
3945     }
3946     {
3947         \seq_if_in:NxT
3948         \l_@@_empty_corner_cells_seq
3949         { \l_tmpa_tl - \l_tmpb_tl }
3950         {
3951             \int_compare:nNnTF \l_tmpa_tl = 1
3952             { \bool_set_false:N \g_tmpa_bool }
3953             {
3954                 \seq_if_in:NxT
3955                 \l_@@_empty_corner_cells_seq
3956                 { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3957                 { \bool_set_false:N \g_tmpa_bool }
3958             }
3959         }
3960     }
3961 }

```

#1 is the number of the row; #2 is the number of horizontal rules to draw (with potentially a color between); #3 and #4 are the number of the columns between which the rule has to be drawn.

```

3962 \cs_new_protected:Npn \@@_hline_ii:nnnn #1 #2 #3 #4
3963 {
3964     \pgfrememberpicturepositiononpagetrue
3965     \pgf@relevantforpicturesizefalse
3966     \@@_qpoint:n { col - #3 }
3967     \dim_set_eq:NN \l_tmpa_dim \pgf@x
3968     \@@_qpoint:n { row - #1 }
3969     \dim_set_eq:NN \l_tmpb_dim \pgf@y

```

```

3970 \@@_qpoint:n { col - \@@_succ:n { #4 } }
3971 \dim_set_eq:NN \l_tmpc_dim \pgf@x
3972 \bool_lazy_and:nnT
3973   { \int_compare_p:nNn { #2 } > 1 }
3974   { ! \tl_if_blank_p:V \CT@drsc@ }
3975   {
3976     \group_begin:
3977     \CT@drsc@
3978     \dim_set:Nn \l_tmpd_dim
3979       { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3980     \pgfpathrectanglecorners
3981       { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3982       { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3983     \pgfusepathqfill
3984     \group_end:
3985   }
3986 \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3987 \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3988 \prg_replicate:nn { #2 - 1 }
3989   {
3990     \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3991     \dim_sub:Nn \l_tmpb_dim \doublerulesep
3992     \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3993     \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3994   }
3995 \CT@arc@
3996 \pgfsetlinewidth { 1.1 \arrayrulewidth }
3997 \pgfsetrectcap
3998 \pgfusepathqstroke
3999 }

4000 \cs_new_protected:Npn \@@_hline_i_complete:nn #1 #2
4001   { \@@_hline_ii:nnnn { #1 } { #2 } 1 { \int_use:N \c@jCol } }

```

The command `\@@_draw_hlines:` draws all the horizontal rules excepted in the blocks (even the virtual drawn determined by commands such as `\Cdots` and in the corners (if the key `except-corners` is used).

```

4002 \cs_new_protected:Npn \@@_draw_hlines:
4003   {
4004     \int_step_inline:nnn
4005       { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
4006       { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
4007       { \@@_hline:nn { ##1 } 1 }
4008   }

```

The command `\@@_Hline:` will be linked to `\Hline` in the environments of `nicematrix`.

```

4009 \cs_set:Npn \@@_Hline: { \noalign { \ifnum 0 = ` } \fi \@@_Hline_i:n { 1 } }

```

The argument of the command `\@@_Hline_i:n` is the number of successive `\Hline` found.

```

4010 \cs_set:Npn \@@_Hline_i:n #1
4011   {
4012     \peek_meaning_ignore_spaces:NTF \Hline
4013       { \@@_Hline_ii:nn { #1 + 1 } }
4014       { \@@_Hline_iii:n { #1 } }
4015   }
4016 \cs_set:Npn \@@_Hline_ii:nn #1 #2 { \@@_Hline_i:n { #1 } }
4017 \cs_set:Npn \@@_Hline_iii:n #1
4018   {
4019     \skip_vertical:n
4020     {
4021       \arrayrulewidth * ( #1 )

```

```

4022     + \doublerulesep * ( \int_max:nn 0 { #1 - 1 } )
4023   }
4024   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4025   { \@@_hline:nn { \@@_succ:n { \c@iRow } } { #1 } }
4026   \ifnum 0 = `{ \fi }
4027 }

```

## The key hvlines

The following command tests whether the current position in the array (given by `\l_tmpa_tl` for the row and `\l_tmpb_tl` for the column) would provide an horizontal rule towards the right in the block delimited by the four arguments `#1`, `#2`, `#3` and `#4`. If this rule would be in the block (it must not be drawn), the boolean `\l_tmpa_bool` is set to false.

```

4028 \cs_new_protected:Npn \@@_test_hline_in_block:nnnn #1 #2 #3 #4
4029 {
4030   \bool_lazy_all:nT
4031   {
4032     { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
4033     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4034     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4035     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4036   }
4037   { \bool_gset_false:N \g_tmpa_bool }
4038 }

```

The same for vertical rules.

```

4039 \cs_new_protected:Npn \@@_test_vline_in_block:nnnn #1 #2 #3 #4
4040 {
4041   \bool_lazy_all:nT
4042   {
4043     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4044     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4045     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
4046     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4047   }
4048   { \bool_gset_false:N \g_tmpa_bool }
4049 }

4050 \cs_new_protected:Npn \@@_test_hline_in_stroken_block:nnnn #1 #2 #3 #4
4051 {
4052   \bool_lazy_all:nT
4053   {
4054     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4055     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 2 } }
4056     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4057     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4058   }
4059   { \bool_gset_false:N \g_tmpa_bool }
4060 }

4061 \cs_new_protected:Npn \@@_test_vline_in_stroken_block:nnnn #1 #2 #3 #4
4062 {
4063   \bool_lazy_all:nT
4064   {
4065     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4066     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4067     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4068     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 2 } }
4069   }
4070   { \bool_gset_false:N \g_tmpa_bool }
4071 }

```

## The key except-corners

When the key `except-corners` is raised, the rules are not drawn in the corners. Of course, we have to compute the corners before we begin to draw the rules.

```
4072 \cs_new_protected:Npn \@@_compute_corners:
4073 {
```

The sequence `\l_@@_empty_corner_cells_seq` will be the sequence of all the empty cells (and not in a block) considered in the corners of the array.

```
4074   \seq_clear_new:N \l_@@_empty_corner_cells_seq
4075   \clist_map_inline:Nn \l_@@_except_corners_clist
4076   {
4077     \str_case:nnF { ##1 }
4078     {
4079       { NW }
4080       { \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol }
4081       { NE }
4082       { \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1 }
4083       { SW }
4084       { \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol }
4085       { SE }
4086       { \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1 }
4087     }
4088     { \@@_error:nn { bad~corner } { ##1 } }
4089   }
4090 }
```

“Computing a corner” is determining all the empty cells (which are not in a block) that belong to that corner. These cells will be added to the sequence `\l_@@_empty_corner_cells_seq`.

The six arguments of `\@@_compute_a_corner:nnnnnn` are as follow:

- #1 and #2 are the number of row and column of the cell which is actually in the corner;
- #3 and #4 are the steps in rows and the step in columns when moving from the corner;
- #5 is the number of the final row when scanning the rows from the corner;
- #6 is the number of the final column when scanning the columns from the corner.

```
4091 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
4092 {
```

For the explanations and the name of the variables, we consider that we are computing the left-upper corner.

First, we try to determine which is the last empty cell (and not in a block: we won’t add that precision any longer) in the column of number 1. The flag `\l_tmpa_bool` will be raised when a non-empty cell is found.

```
4093   \bool_set_false:N \l_tmpa_bool
4094   \int_zero_new:N \l_@@_last_empty_row_int
4095   \int_set:Nn \l_@@_last_empty_row_int { #1 }
4096   \int_step_inline:nnnn { #1 } { #3 } { #5 }
4097   {
4098     \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
4099     \bool_lazy_or:nnTF
4100     {
4101       \cs_if_exist_p:c
4102       { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
4103     }
4104     \l_tmpb_bool
4105     { \bool_set_true:N \l_tmpa_bool }
4106     {
4107       \bool_if:NF \l_tmpa_bool
4108       { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
4109     }
4110   }
```

Now, you determine the last empty cell in the row of number 1.

```

4111 \bool_set_false:N \l_tmpa_bool
4112 \int_zero_new:N \l_@@_last_empty_column_int
4113 \int_set:Nn \l_@@_last_empty_column_int { #2 }
4114 \int_step_inline:nnnn { #2 } { #4 } { #6 }
4115 {
4116   \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
4117   \bool_lazy_or:nnTF
4118     \l_tmpb_bool
4119     {
4120       \cs_if_exist_p:c
4121         { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
4122     }
4123     { \bool_set_true:N \l_tmpa_bool }
4124     {
4125       \bool_if:NF \l_tmpa_bool
4126       { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
4127     }
4128 }

```

Now, we loop over the rows.

```

4129 \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
4130 {

```

We treat the row number ##1 with another loop.

```

4131 \bool_set_false:N \l_tmpa_bool
4132 \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
4133 {
4134   \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
4135   \bool_lazy_or:nnTF
4136     \l_tmpb_bool
4137     {
4138       \cs_if_exist_p:c
4139         { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
4140     }
4141     { \bool_set_true:N \l_tmpa_bool }
4142     {
4143       \bool_if:NF \l_tmpa_bool
4144       {
4145         \int_set:Nn \l_@@_last_empty_column_int { #####1 }
4146         \seq_put_right:Nn
4147           \l_@@_empty_corner_cells_seq
4148           { ##1 - #####1 }
4149       }
4150     }
4151 }
4152 }
4153 }

```

The following macro tests whether a cell is in (at least) one of the blocks of the array (or in a cell with a `\diagbox`).

The flag `\l_tmpb_bool` will be raised if the cell #1-#2 is in a block (or in a cell with a `\diagbox`).

```

4154 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
4155 {
4156   \int_set:Nn \l_tmpa_int { #1 }
4157   \int_set:Nn \l_tmpb_int { #2 }
4158   \bool_set_false:N \l_tmpb_bool
4159   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4160     { \@@_test_if_cell_in_block:nnnnnn \l_tmpa_int \l_tmpb_int ##1 }
4161 }
4162 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnn #1 #2 #3 #4 #5 #6
4163 {
4164   \int_compare:nNnT { #3 } < { \@@_succ:n { #1 } }

```

```

4165     {
4166       \int_compare:nNnT { #1 } < { \@@_succ:n { #5 } }
4167       {
4168         \int_compare:nNnT { #4 } < { \@@_succ:n { #2 } }
4169         {
4170           \int_compare:nNnT { #2 } < { \@@_succ:n { #6 } }
4171           { \bool_set_true:N \l_tmpb_bool }
4172         }
4173       }
4174     }
4175   }

```

## The commands to draw dotted lines to separate columns and rows

These commands don't use the normal nodes, the medium nor the large nodes. They only use the col nodes and the row nodes.

### Horizontal dotted lines

The following command must *not* be protected because it's meant to be expanded in a `\noalign`.

```

4176 \cs_new:Npn \@@_hdottedline:
4177 {
4178   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
4179   \@@_hdottedline_i:
4180 }

```

On the other side, the following command should be protected.

```

4181 \cs_new_protected:Npn \@@_hdottedline_i:
4182 {

```

We write in the code-after the instruction that will actually draw the dotted line. It's not possible to draw this dotted line now because we don't know the length of the line (we don't even know the number of columns).

```

4183   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4184   { \@@_hdottedline:n { \int_use:N \c@iRow } }
4185 }

```

The command `\@@_hdottedline:n` is the command written in the `\CodeAfter` that will actually draw the dotted line. Its argument is the number of the row *before* which we will draw the row.

```

4186 \AtBeginDocument
4187 {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”. That's why we construct now a version of `\@@_hdottedline:n` with the right environment (`\begin{pgfpicture}\end{pgfpicture}` or `\begin{tikzpicture}...\end{tikzpicture}`).

```

4188   \cs_new_protected:Npx \@@_hdottedline:n #1
4189   {
4190     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
4191     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
4192     \c_@@_pgfortikzpicture_tl
4193     \@@_hdottedline_i:n { #1 }
4194     \c_@@_endpgfortikzpicture_tl
4195   }
4196 }

```

The following command *must* be protected since it is used in the construction of `\@@_hdottedline:n`.

```

4197 \cs_new_protected:Npn \@@_hdottedline_i:n #1
4198 {
4199   \pgfrememberpicturepositiononpagetrue
4200   \@@_qpoint:n { row - #1 }

```

We do a translation par `-l_@@_radius_dim` because we want the dotted line to have exactly the same position as a vertical rule drawn by “|” (considering the rule having a width equal to the diameter of the dots).

```

4201 \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
4202 \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
4203 \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

The dotted line will be extended if the user uses `margin` (or `left-margin` and `right-margin`).

The aim is that, by standard the dotted line fits between square brackets (`\hline` doesn’t).

```

\begin{bNiceMatrix}
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & 4 \\
\hdottedline
1 & 2 & 3 & 4
\end{bNiceMatrix}

```

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

But, if the user uses `margin`, the dotted line extends to have the same width as a `\hline`.

```

\begin{bNiceMatrix}[margin]
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & 4 \\
\hdottedline
1 & 2 & 3 & 4
\end{bNiceMatrix}

```

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

```

4204 \@@_qpoint:n { col - 1 }
4205 \dim_set:Nn \l_@@_x_initial_dim
4206 {
4207 \pgf@x +

```

We do a reduction by `\arraycolsep` for the environments with delimiters (and not for the other).

```

4208 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4209 - \l_@@_left_margin_dim
4210 }
4211 \@@_qpoint:n { col - \@@_succ:n \c@jCol }
4212 \dim_set:Nn \l_@@_x_final_dim
4213 {
4214 \pgf@x -
4215 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4216 + \l_@@_right_margin_dim
4217 }

```

For reasons purely aesthetic, we do an adjustment in the case of a rounded bracket. The correction by `0.5 \l_@@_inter_dots_dim` is *ad hoc* for a better result.

```

4218 \tl_if_eq:NnF \l_@@_left_delim_tl (
4219 { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
4220 \tl_if_eq:NnF \l_@@_right_delim_tl )
4221 { \dim_gsub:Nn \l_@@_x_final_dim { 0.5 \l_@@_inter_dots_dim } }

```

As for now, we have no option to control the style of the lines drawn by `\hdottedline` and the specifier “:” in the preamble. That’s why we impose the style `standard`.

```

4222 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4223 \@@_draw_line:
4224 }

```

## Vertical dotted lines

```

4225 \cs_new_protected:Npn \@@_vdottedline:n #1
4226 {
4227 \bool_set_true:N \l_@@_initial_open_bool
4228 \bool_set_true:N \l_@@_final_open_bool

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”.

```

4229   \bool_if:NTF \c_@@_tikz_loaded_bool
4230   {
4231     \tikzpicture
4232     \@@_vdottedline_i:n { #1 }
4233     \endtikzpicture
4234   }
4235   {
4236     \pgfpicture
4237     \@@_vdottedline_i:n { #1 }
4238     \endpgfpicture
4239   }
4240 }

4241 \cs_new_protected:Npn \@@_vdottedline_i:n #1
4242 {

```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded.

```

4243   \CT@arc@
4244   \pgfrememberpicturepositiononpagetrue
4245   \@@_qpoint:n { col - \int_eval:n { #1 + 1 } }

```

We do a translation `par -\l_@@_radius_dim` because we want the dotted line to have exactly the same position as a vertical rule drawn by “|” (considering the rule having a width equal to the diameter of the dots).

```

4246   \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
4247   \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
4248   \@@_qpoint:n { row - 1 }

```

We arbitrary decrease the height of the dotted line by a quantity equal to `\l_@@_inter_dots_dim` in order to improve the visual impact.

```

4249   \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
4250   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
4251   \dim_set:Nn \l_@@_y_final_dim { \pgf@y + 0.5 \l_@@_inter_dots_dim }

```

As for now, we have no option to control the style of the lines drawn by `\hdottedline` and the specifier “:” in the preamble. That’s why we impose the style `standard`.

```

4252   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4253   \@@_draw_line:
4254 }

```

## The environment `{NiceMatrixBlock}`

The following flag will be raised when all the columns of the environments of the block must have the same width in “auto” mode.

```

4255 \bool_new:N \l_@@_block_auto_columns_width_bool

```

As for now, there is only one option available for the environment `{NiceMatrixBlock}`.

```

4256 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
4257 {
4258   auto-columns-width .code:n =
4259   {
4260     \bool_set_true:N \l_@@_block_auto_columns_width_bool
4261     \dim_gzero_new:N \g_@@_max_cell_width_dim
4262     \bool_set_true:N \l_@@_auto_columns_width_bool
4263   }
4264 }

```



```

4265 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
4266 {
4267   \int_gincr:N \g_@@_NiceMatrixBlock_int
4268   \dim_zero:N \l_@@_columns_width_dim
4269   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
4270   \bool_if:NT \l_@@_block_auto_columns_width_bool
4271   {
4272     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4273     {
4274       \exp_args:Nnc \dim_set:Nn \l_@@_columns_width_dim
4275         { @@_max_cell_width _ \int_use:N \g_@@_NiceMatrixBlock_int }
4276     }
4277   }
4278 }

```

At the end of the environment {NiceMatrixBlock}, we write in the main .aux file instructions for the column width of all the environments of the block (that’s why we have stored the number of the first environment of the block in the counter \l\_@@\_first\_env\_block\_int).

```

4279 {
4280   \bool_if:NT \l_@@_block_auto_columns_width_bool
4281   {
4282     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
4283     \iow_shipout:Nx \@mainaux
4284     {
4285       \cs_gset:cpn
4286         { @@ _ max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }

```

For technical reasons, we have to include the width of a potential rule on the right side of the cells.

```

4287       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
4288     }
4289     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
4290   }
4291 }

```

## The extra nodes

First, two variants of the functions \dim\_min:nn and \dim\_max:nn.

```

4292 \cs_generate_variant:Nn \dim_min:nn { v n }
4293 \cs_generate_variant:Nn \dim_max:nn { v n }

```

The following command is called in \@@\_use\_arraybox\_with\_notes\_c: just before the construction of the blocks (if the creation of medium nodes is required, medium nodes are also created for the blocks dans that construction uses the standard medium nodes).

```

4294 \cs_new_protected:Npn \@@_create_extra_nodes:
4295 {
4296   \bool_if:nTF \l_@@_medium_nodes_bool
4297   {
4298     \bool_if:nTF \l_@@_large_nodes_bool
4299     \@@_create_medium_and_large_nodes:
4300     \@@_create_medium_nodes:
4301   }
4302   { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
4303 }

```

We have three macros of creation of nodes: \@@\_create\_medium\_nodes:, \@@\_create\_large\_nodes: and \@@\_create\_medium\_and\_large\_nodes:.

We have to compute the mathematical coordinates of the “medium nodes”. These mathematical coordinates are also used to compute the mathematical coordinates of the “large nodes”. That’s why we write a command \@@\_computations\_for\_medium\_nodes: to do these computations.

The command `\@@_computations_for_medium_nodes:` must be used in a `{pgfpicture}`.

For each row  $i$ , we compute two dimensions `l_@@_row_i_min_dim` and `l_@@_row_i_max_dim`. The dimension `l_@@_row_i_min_dim` is the minimal  $y$ -value of all the cells of the row  $i$ . The dimension `l_@@_row_i_max_dim` is the maximal  $y$ -value of all the cells of the row  $i$ .

Similarly, for each column  $j$ , we compute two dimensions `l_@@_column_j_min_dim` and `l_@@_column_j_max_dim`. The dimension `l_@@_column_j_min_dim` is the minimal  $x$ -value of all the cells of the column  $j$ . The dimension `l_@@_column_j_max_dim` is the maximal  $x$ -value of all the cells of the column  $j$ .

Since these dimensions will be computed as maximum or minimum, we initialize them to `\c_max_dim` or `-\c_max_dim`.

```

4304 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
4305 {
4306   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4307   {
4308     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
4309     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
4310     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
4311     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
4312   }
4313   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4314   {
4315     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
4316     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
4317     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
4318     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
4319   }

```

We begin the two nested loops over the rows and the columns of the array.

```

4320   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4321   {
4322     \int_step_variable:nnNn
4323     \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

If the cell  $(i-j)$  is empty or an implicit cell (that is to say a cell after implicit ampersands `&`) we don't update the dimensions we want to compute.

```

4324     {
4325       \cs_if_exist:cT
4326       { pgf @ sh @ ns @ \@@_env: - \@@_i: - \@@_j: }

```

We retrieve the coordinates of the anchor south west of the (normal) node of the cell  $(i-j)$ . They will be stored in `\pgf@x` and `\pgf@y`.

```

4327     {
4328       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
4329       \dim_set:cn { l_@@_row_\@@_i: _min_dim }
4330       { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
4331       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4332       {
4333         \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
4334         { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
4335       }

```

We retrieve the coordinates of the anchor north east of the (normal) node of the cell  $(i-j)$ . They will be stored in `\pgf@x` and `\pgf@y`.

```

4336       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
4337       \dim_set:cn { l_@@_row _ \@@_i: _max_dim }
4338       { \dim_max:vn { l_@@_row _ \@@_i: _max_dim } \pgf@y }
4339       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4340       {
4341         \dim_set:cn { l_@@_column _ \@@_j: _max_dim }
4342         { \dim_max:vn { l_@@_column _ \@@_j: _max_dim } \pgf@x }
4343       }
4344     }
4345   }
4346 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

4347 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4348 {
4349   \dim_compare:nNnT
4350     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
4351     {
4352       \@@_qpoint:n { row - \@@_i: - base }
4353       \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
4354       \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
4355     }
4356 }
4357 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4358 {
4359   \dim_compare:nNnT
4360     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
4361     {
4362       \@@_qpoint:n { col - \@@_j: }
4363       \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
4364       \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
4365     }
4366 }
4367 }

```

Here is the command `\@@_create_medium_nodes:`. When this command is used, the “medium nodes” are created.

```

4368 \cs_new_protected:Npn \@@_create_medium_nodes:
4369 {
4370   \pgfpicture
4371   \pgfrememberpicturepositiononpagetrue
4372   \pgf@relevantforpicturesizefalse
4373   \@@_computations_for_medium_nodes:

```

Now, we can create the “medium nodes”. We use a command `\@@_create_nodes:` because this command will also be used for the creation of the “large nodes”.

```

4374   \tl_set:Nn \l_@@_suffix_tl { -medium }
4375   \@@_create_nodes:
4376   \endpgfpicture
4377 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones<sup>53</sup>. However, the computation of the mathematical coordinates of the “large nodes” needs the computation of the mathematical coordinates of the “medium nodes”. Hence, we use first `\@@_computations_for_medium_nodes:` and then the command `\@@_computations_for_large_nodes:`.

```

4378 \cs_new_protected:Npn \@@_create_large_nodes:
4379 {
4380   \pgfpicture
4381   \pgfrememberpicturepositiononpagetrue
4382   \pgf@relevantforpicturesizefalse
4383   \@@_computations_for_medium_nodes:
4384   \@@_computations_for_large_nodes:
4385   \tl_set:Nn \l_@@_suffix_tl { -large }
4386   \@@_create_nodes:
4387   \endpgfpicture
4388 }
4389 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
4390 {
4391   \pgfpicture
4392   \pgfrememberpicturepositiononpagetrue

```

<sup>53</sup>If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

```

4393 \pgf@relevantforpicturesizefalse
4394 \@@_computations_for_medium_nodes:

```

Now, we can create the “medium nodes”. We use a command `\@@_create_nodes:` because this command will also be used for the creation of the “large nodes”.

```

4395 \tl_set:Nn \l_@@_suffix_tl { - medium }
4396 \@@_create_nodes:
4397 \@@_computations_for_large_nodes:
4398 \tl_set:Nn \l_@@_suffix_tl { - large }
4399 \@@_create_nodes:
4400 \endpgfpicture
4401 }

```

For “large nodes”, the exterior rows and columns don’t interfere. That’s why the loop over the columns will start at 1 and stop at `\c@jCol` (and not `\g_@@_col_total_int`). Idem for the rows.

```

4402 \cs_new_protected:Npn \@@_computations_for_large_nodes:
4403 {
4404 \int_set:Nn \l_@@_first_row_int 1
4405 \int_set:Nn \l_@@_first_col_int 1

```

We have to change the values of all the dimensions `l_@@_row_i_min_dim`, `l_@@_row_i_max_dim`, `l_@@_column_j_min_dim` and `l_@@_column_j_max_dim`.

```

4406 \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
4407 {
4408 \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
4409 {
4410 (
4411 \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
4412 \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4413 )
4414 / 2
4415 }
4416 \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4417 { l_@@_row _ \@@_i: _ min _ dim }
4418 }
4419 \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
4420 {
4421 \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
4422 {
4423 (
4424 \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
4425 \dim_use:c
4426 { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4427 )
4428 / 2
4429 }
4430 \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4431 { l_@@_column _ \@@_j: _ max _ dim }
4432 }

```

Here, we have to use `\dim_sub:cn` because of the number 1 in the name.

```

4433 \dim_sub:cn
4434 { l_@@_column _ 1 _ min _ dim }
4435 \l_@@_left_margin_dim
4436 \dim_add:cn
4437 { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
4438 \l_@@_right_margin_dim
4439 }

```

The command `\@@_create_nodes:` is used twice: for the construction of the “medium nodes” and for the construction of the “large nodes”. The nodes are constructed with the value of all the dimensions

`l_@@_row_i_min_dim`, `l_@@_row_i_max_dim`, `l_@@_column_j_min_dim` and `l_@@_column_j_max_dim`. Between the construction of the “medium nodes” and the “large nodes”, the values of these dimensions are changed.

The function also uses `\l_@@_suffix_tl` (-medium or -large).

```

4440 \cs_new_protected:Npn \@@_create_nodes:
4441 {
4442   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4443   {
4444     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4445     {

```

We draw the rectangular node for the cell (`\@@_i-\@@_j`).

```

4446       \@@_pgf_rect_node:nnnnn
4447       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4448       { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
4449       { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
4450       { \dim_use:c { l_@@_column_ \@@_j: _max_dim } }
4451       { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
4452       \str_if_empty:NF \l_@@_name_str
4453       {
4454         \pgfnodealias
4455         { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4456         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4457       }
4458     }
4459   }

```

Now, we create the nodes for the cells of the `\multicolumn`. We recall that we have stored in `\g_@@_multicolumn_cells_seq` the list of the cells where a `\multicolumn{n}{...}{...}` with  $n > 1$  was issued and in `\g_@@_multicolumn_sizes_seq` the correspondent values of  $n$ .

```

4460   \seq_mapthread_function:NNN
4461   \g_@@_multicolumn_cells_seq
4462   \g_@@_multicolumn_sizes_seq
4463   \@@_node_for_multicolumn:nn
4464 }

```

```

4465 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
4466 {
4467   \cs_set_nopar:Npn \@@_i: { #1 }
4468   \cs_set_nopar:Npn \@@_j: { #2 }
4469 }

```

The command `\@@_node_for_multicolumn:nn` takes two arguments. The first is the position of the cell where the command `\multicolumn{n}{...}{...}` was issued in the format  $i-j$  and the second is the value of  $n$  (the length of the “multi-cell”).

```

4470 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
4471 {
4472   \@@_extract_coords_values: #1 \q_stop
4473   \@@_pgf_rect_node:nnnnn
4474   { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4475   { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
4476   { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
4477   { \dim_use:c { l_@@_column_ \int_eval:n { \@@_j: +#2-1 } _max_dim } }
4478   { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
4479   \str_if_empty:NF \l_@@_name_str
4480   {
4481     \pgfnodealias
4482     { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4483     { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
4484   }
4485 }

```

## The blocks

The code deals with the command `\Block`. This command has no direct link with the environment `{NiceMatrixBlock}`.

The options of the command `\Block` will be analyzed first in the cell of the array (and once again when the block will be put in the array). Here is the set of keys for the first pass.

```

4486 \keys_define:nn { NiceMatrix / Block / FirstPass }
4487 {
4488   l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4489   l .value_forbidden:n = true ,
4490   r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4491   r .value_forbidden:n = true ,
4492   c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4493   c .value_forbidden:n = true ,

```

The two following lines will be uncommented when we will give to the key `color` its new definition.

```

4494   % color .tl_set:N = \l_@@_color_tl ,
4495   % color .value_required:n = true ,
4496 }

```

The following command `\@@_Block:` will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewExpandableDocumentCommand` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label and before the beginning of the small array of the block). It's mandatory to use an expandable command.

```

4497 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
4498 {

```

If the first mandatory argument of the command (which is the size of the block with the syntax  $i-j$ ) has not been provided by the user, you use `1-1` (that is to say a block of only one cell).

```

4499   \tl_if_blank:nTF { #2 } { \@@_Block_i 1-1 \q_stop } { \@@_Block_i #2 \q_stop }
4500   { #1 } { #3 } { #4 }
4501 }

```

With the following construction, we extract the values of  $i$  and  $j$  in the first mandatory argument of the command.

```

4502 \cs_new:Npn \@@_Block_i #1-#2 \q_stop { \@@_Block_ii:nnnnn { #1 } { #2 } }

```

Now, the arguments have been extracted: `#1` is  $i$  (the number of rows of the block), `#2` is  $j$  (the number of columns of the block), `#3` is the list of key-values, `#4` are the tokens to put before the math mode and the beginning of the small array of the block and `#5` is the label of the block.

```

4503 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
4504 {

```

We recall that `#1` and `#2` have been extracted from the first mandatory argument of `\Block` (which is of the syntax  $i-j$ ). However, the user is allowed to omit  $i$  or  $j$  (or both). We detect that situation by replacing a missing value by 100 (it's a convention: when the block will actually be drawn these values will be detected and interpreted as *maximal possible value*).

```

4505   \bool_lazy_or:nnTF
4506     { \tl_if_blank_p:n { #1 } }
4507     { \str_if_eq_p:nn { #1 } { * } }
4508     { \int_set:Nn \l_tmpa_int { 100 } }
4509     { \int_set:Nn \l_tmpa_int { #1 } }
4510   \bool_lazy_or:nnTF
4511     { \tl_if_blank_p:n { #2 } }
4512     { \str_if_eq_p:nn { #2 } { * } }
4513     { \int_set:Nn \l_tmpb_int { 100 } }
4514     { \int_set:Nn \l_tmpb_int { #2 } }

```

```

4515 \int_compare:nNnTF \l_tmpb_int = 1
4516 {
4517   \tl_if_empty:NTF \l_@@_cell_type_tl
4518     { \tl_set:Nn \l_@@_pos_of_block_tl c }
4519     { \tl_set_eq:NN \l_@@_pos_of_block_tl \l_@@_cell_type_tl }
4520 }
4521 { \tl_set:Nn \l_@@_pos_of_block_tl c }

4522 \keys_set:known:n { NiceMatrix / Block / FirstPass } { #3 }

4523 \tl_set:Nx \l_tmpa_tl
4524 {
4525   { \int_use:N \c@iRow }
4526   { \int_use:N \c@jCol }
4527   { \int_eval:n { \c@iRow + \l_tmpa_int - 1 } }
4528   { \int_eval:n { \c@jCol + \l_tmpb_int - 1 } }
4529 }

```

Now, `\l_tmpa_tl` contains an “object” corresponding to the position of the block with four components, each of them surrounded by curly brackets: `{imin}{jmin}{imax}{jmax}`.

If the block is mono-column or mono-row, we have a special treatment. That’s why we have two macros: `\@@_Block_iv:nnnnn` and `\@@_Block_v:nnnnn` (the five arguments of those macros are provided by curryfication).

```

4530 \bool_lazy_or:nnTF
4531 { \int_compare_p:nNn { \l_tmpa_int } = 1 }
4532 { \int_compare_p:nNn { \l_tmpb_int } = 1 }
4533 { \exp_args:Nxx \@@_Block_iv:nnnnn }
4534 { \exp_args:Nxx \@@_Block_v:nnnnn }
4535 { \l_tmpa_int } { \l_tmpb_int } { #3 } { #4 } { #5 }
4536 }

```

The following macro is for the case of a `\Block` which is mono-row or mono-column (or both). In that case, the content of the block is composed right now in a box (because we have to take into account the dimensions of that box for the width of the current column or the height and the depth of the current row). However, that box will be put in the array *after the construction of the array* (by using PGF).

```

4537 \cs_new_protected:Npn \@@_Block_iv:nnnnn #1 #2 #3 #4 #5
4538 {
4539   \int_gincr:N \g_@@_block_box_int
4540   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4541   {
4542     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4543     {
4544       \@@_actually_diagbox:nnnnnn
4545       { \int_use:N \c@iRow }
4546       { \int_use:N \c@jCol }
4547       { \int_eval:n { \c@iRow + #1 - 1 } }
4548       { \int_eval:n { \c@jCol + #2 - 1 } }
4549       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4550     }
4551   }
4552   \box_gclear_new:c
4553   { g_@@_block_box_int \int_use:N \g_@@_block_box_int _ box }
4554   \hbox_gset:cn
4555   { g_@@_block_box_int \int_use:N \g_@@_block_box_int _ box }
4556   {

```

For a mono-column block, if the user has specified a color for the column in the preamble of the array, we want to fix that color in the box we construct. We do that with `\set@color` and not `\color_ensure_current:` because that command seems to be bugged: it doesn’t work in XeLaTeX when `fontspec` is loaded.

```

4557 \tl_if_empty:NTF \l_@@_color_tl
4558 { \int_compare:nNnT { #2 } = 1 \set@color }
4559 { \color { \l_@@_color_tl } }
4560 \group_begin:
4561 \cs_set:Npn \arraystretch { 1 }
4562 \dim_set_eq:NN \extrarowheight \c_zero_dim
4563 #4

```

If the box is rotated (the key `\rotate` may be in the previous #4), the tabular used for the content of the cell will be constructed with a format `c`. In the other cases, the tabular will be constructed with a format equal to the key of position of the box. In other words: the alignment internal to the tabular is the same as the external alignment of the tabular (that is to say the position of the block in its zone of merged cells).

```

4564 \bool_if:NT \g_@@_rotate_bool { \tl_set:Nn \l_@@_pos_of_block_tl c }
4565 \bool_if:NTF \l_@@_NiceTabular_bool
4566 {
4567   \exp_args:Nnx \begin { tabular }
4568     { @ { } \l_@@_pos_of_block_tl @ { } }
4569     #5
4570   \end { tabular }
4571 }
4572 {
4573   \c_math_toggle_token
4574   \exp_args:Nnx \begin { array }
4575     { @ { } \l_@@_pos_of_block_tl @ { } }
4576     #5
4577   \end { array }
4578   \c_math_toggle_token
4579 }
4580 \group_end:
4581 }
4582 \bool_if:NT \g_@@_rotate_bool
4583 {
4584   \box_grotate:cn
4585     { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4586     { 90 }
4587   \bool_gset_false:N \g_@@_rotate_bool
4588 }

```

If we are in a mono-column block, we take into account the width of that block for the width of the column.

```

4589 \int_compare:nNnT { #2 } = 1
4590 {
4591   \dim_gset:Nn \g_@@_blocks_wd_dim
4592   {
4593     \dim_max:nn
4594       \g_@@_blocks_wd_dim
4595       {
4596         \box_wd:c
4597           { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4598       }
4599   }
4600 }

```

If we are in a mono-row block, we take into account the height and the depth of that block for the height and the depth of the row.

```

4601 \int_compare:nNnT { #1 } = 1
4602 {
4603   \dim_gset:Nn \g_@@_blocks_ht_dim
4604   {
4605     \dim_max:nn
4606       \g_@@_blocks_ht_dim
4607       {
4608         \box_ht:c

```



```

4609         { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4610     }
4611 }
4612 \dim_gset:Nn \g_@@_blocks_dp_dim
4613 {
4614     \dim_max:nn
4615     \g_@@_blocks_dp_dim
4616     {
4617         \box_dp:c
4618         { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4619     }
4620 }
4621 }
4622 \seq_gput_right:Nx \g_@@_blocks_seq
4623 {
4624     \l_tmpa_tl

```

In the list of options #3, maybe there is a key for the horizontal alignment (l, r or c). In that case, that key has been read and stored in \l\_@@\_pos\_of\_block\_tl. However, maybe there were no key of the horizontal alignment and that's why we put a key corresponding to the value of \l\_@@\_pos\_of\_block\_tl, which is fixed by the type of current column.

```

4625     { \exp_not:n { #3 } , \l_@@_pos_of_block_tl }
4626     {
4627         \box_use_drop:c
4628         { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4629     }
4630 }
4631 }

```

The following macro is for the standard case, where the block is not mono-row and not mono-column. In that case, the content of the block is *not* composed right now in a box. The composition in a box will be done further, just after the construction of the array.

```

4632 \cs_new_protected:Npn \@@_Block_v:nnnnn #1 #2 #3 #4 #5
4633 {
4634     \seq_gput_right:Nx \g_@@_blocks_seq
4635     {
4636         \l_tmpa_tl
4637         { \exp_not:n { #3 } }
4638         \exp_not:n
4639         {
4640             {
4641                 \bool_if:NTF \l_@@_NiceTabular_bool
4642                 {
4643                     \group_begin:
4644                     \cs_set:Npn \arraystretch { 1 }
4645                     \dim_set_eq:NN \extrarowheight \c_zero_dim
4646                     #4

```

If the box is rotated (the key \rotate may be in the previous #4), the tabular used for the content of the cell will be constructed with a format c. In the other cases, the tabular will be constructed with a format equal to the key of position of the box. In other words: the alignment internal to the tabular is the same as the external alignment of the tabular (that is to say the position of the block in its zone of merged cells).

```

4647         \bool_if:NT \g_@@_rotate_bool
4648         { \tl_set:Nn \l_@@_pos_of_block_tl c }
4649         \exp_args:Nnx \begin { tabular }
4650         { @ { } \l_@@_pos_of_block_tl @ { } }
4651         #5
4652         \end { tabular }
4653         \group_end:
4654     }
4655     {
4656         \group_begin:

```

```

4657         \cs_set:Npn \arraystretch { 1 }
4658         \dim_set_eq:NN \extrarowheight \c_zero_dim
4659         #4
4660         \bool_if:NT \g_@@_rotate_bool
4661         { \tl_set:Nn \l_@@_pos_of_block_tl c }
4662         \c_math_toggle_token
4663         \exp_args:Nnx \begin { array }
4664         { @ { } \l_@@_pos_of_block_tl @ { } } #5 \end { array }
4665         \c_math_toggle_token
4666         \group_end:
4667     }
4668 }
4669 }
4670 }
4671 }

```

We recall that the options of the command `\Block` are analyzed twice: first in the cell of the array and once again when the block will be put in the array *after the construction of the array* (by using PGF).

```

4672 \keys_define:nn { NiceMatrix / Block / SecondPass }
4673 {
4674     fill .tl_set:N = \l_@@_fill_tl ,
4675     fill .value_required:n = true ,
4676     draw .tl_set:N = \l_@@_draw_tl ,
4677     draw .default:n = default ,

```

The following definition for the key `color` will be deleted when we will give to the key `color` of the command `\Block` its new definition. It will be replaced by the line which is commented.

```

4678     color .code:n =
4679         \@@_fatal:n { Key-color-for-Block }
4680         \tl_set:Nn \l_@@_fill_tl { #1 } ,
4681     % color .code:n = \color { #1 } \tl_set:Nn \l_@@_draw_tl { #1 } ,
4682     color .value_required:n = true ,
4683     line-width .dim_set:N = \l_@@_line_width_dim ,
4684     line-width .value_required:n = true ,
4685     l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4686     l .value_forbidden:n = true ,
4687     r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4688     r .value_forbidden:n = true ,
4689     c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4690     c .value_forbidden:n = true ,
4691     unknown .code:n = \@@_error:n { Unknown-key-for-Block }
4692 }

```

The command `\@@_draw_blocks:` will draw all the blocks. This command is used after the construction of the array. We have to revert to a clean version of `\ialign` because there may be tabulars in the `\Block` instructions that will be composed now.

```

4693 \cs_new_protected:Npn \@@_draw_blocks:
4694 {
4695     \cs_set_eq:NN \ialign \@@_old_ialign:
4696     \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iv:nnnnnn ##1 }
4697 }
4698 \cs_new_protected:Npn \@@_Block_iv:nnnnnn #1 #2 #3 #4 #5 #6
4699 {

```

The integer `\l_@@_last_row_int` will be the last row of the block and `\l_@@_last_col_int` its last column.

```

4700     \int_zero_new:N \l_@@_last_row_int
4701     \int_zero_new:N \l_@@_last_col_int

```

We remind that the first mandatory argument of the command `\Block` is the size of the block with the special format  $i-j$ . However, the user is allowed to omit  $i$  or  $j$  (or both). This will be interpreted as: the last row (resp. column) of the block will be the last row (resp. column) of the block (without the potential exterior row—resp. column—of the array). By convention, this is stored in `\g_@@_blocks_seq` as a number of rows (resp. columns) for the block equal to 100. That's what we detect now.

```

4702 \int_compare:nNnTF { #3 } > { 99 }
4703   { \int_set_eq:NN \l_@@_last_row_int \c@iRow }
4704   { \int_set:Nn \l_@@_last_row_int { #3 } }
4705 \int_compare:nNnTF { #4 } > { 99 }
4706   { \int_set_eq:NN \l_@@_last_col_int \c@jCol }
4707   { \int_set:Nn \l_@@_last_col_int { #4 } }
4708 \int_compare:nNnTF \l_@@_last_col_int > \g_@@_col_total_int
4709   {
4710     \int_compare:nTF
4711       { \l_@@_last_col_int <= \g_@@_static_num_of_col_int }
4712       {
4713         \msg_error:nnnn { nicematrix } { Block-too~large~2 } { #1 } { #2 }
4714         @@_msg_redirect_name:nn { Block-too~large~2 } { none }
4715         \group_begin:
4716         \globaldefs = 1
4717         @@_msg_redirect_name:nn { columns~not~used } { none }
4718         \group_end:
4719       }
4720       { \msg_error:nnnn { nicematrix } { Block-too~large~1 } { #1 } { #2 } }
4721   }
4722   {
4723     \int_compare:nNnTF \l_@@_last_row_int > \g_@@_row_total_int
4724       { \msg_error:nnnn { nicematrix } { Block-too~large~1 } { #1 } { #2 } }
4725       { @@_Block_v:nnnnnn { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } }
4726   }
4727 }
4728 \cs_new_protected:Npn @@_Block_v:nnnnnn #1 #2 #3 #4 #5 #6
4729   {

```

The sequence of the positions of the blocks will be used when drawing the rules (in fact, there is also the `\multicolumn` and the `\diagbox` in that sequence).

```

4730 \seq_gput_left:Nn \g_@@_pos_of_blocks_seq { { #1 } { #2 } { #3 } { #4 } }

```

The group is for the keys.

```

4731 \group_begin:
4732 \keys_set:nn { NiceMatrix / Block / SecondPass } { #5 }
4733 \bool_lazy_or:nnT
4734   { ! \tl_if_empty_p:N \l_@@_draw_tl }
4735   { \dim_compare_p:nNn \l_@@_line_width_dim > \c_zero_dim }
4736   {
4737     \tl_gput_right:Nx \g_nicematrix_code_after_tl
4738     {
4739       @@_stroke_block:nnn
4740       { \exp_not:n { #5 } }
4741       { #1 - #2 }
4742       { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
4743     }
4744     \seq_gput_right:Nn \g_@@_pos_of_stroken_blocks_seq
4745       { { #1 } { #2 } { #3 } { #4 } }
4746   }
4747 \tl_if_empty:NF \l_@@_fill_tl
4748   {
4749     \tl_gput_right:Nx \g_nicematrix_code_before_tl
4750     {
4751       \exp_not:N \rectanglecolor
4752       { \exp_not:V \l_@@_fill_tl }

```

```

4753         { #1 - #2 }
4754         { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
4755     }
4756 }

4757 \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4758 {
4759     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4760     {
4761         \@@_actually_diagbox:nnnnnn
4762         { #1 }
4763         { #2 }
4764         { \int_use:N \l_@@_last_row_int }
4765         { \int_use:N \l_@@_last_col_int }
4766         { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4767     }
4768 }

4769 \hbox_set:Nn \l_@@_cell_box { #6 }
4770 \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:

```

Let's consider the following `{NiceTabular}`. Because of the instruction `!\hspace{1cm}` in the preamble which increases the space between the columns (by adding, in fact, that space to the previous column, that is to say the second column of the tabular), we will create *two* nodes relative to the block: the node `1-1-block` and the node `1-1-block-short`. The latter will be used by `nicematrix` to put the label of the node. The first one won't be used explicitly.

```

\begin{NiceTabular}{cc!\hspace{1cm}}c}
\Block{2-2}{our block} & & one & \\
& & two & \\
three & four & five & \\
six & seven & eight & \\
\end{NiceTabular}

```

We highlight the node `1-1-block`

our block		one
		two
three	four	five
six	seven	eight

We highlight the node `1-1-block-short`

our block		one
		two
three	four	five
six	seven	eight

The construction of the node corresponding to the merged cells.

```

4771 \pgfpicture
4772 \pgfrememberpicturepositiononpagetrue
4773 \pgf@relevantforpicturesizefalse
4774 \@@_qpoint:n { row - #1 }
4775 \dim_set_eq:NN \l_tmpa_dim \pgf@y
4776 \@@_qpoint:n { col - #2 }
4777 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4778 \@@_qpoint:n { row - \@@_succ:n { \l_@@_last_row_int } }
4779 \dim_set_eq:NN \l_tmpc_dim \pgf@y
4780 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
4781 \dim_set_eq:NN \l_tmpd_dim \pgf@x

```

We construct the node for the block with the name `(#1-#2-block)`.

The function `\@@_pgf_rect_node:nnnnn` takes in as arguments the name of the node and the four coordinates of two opposite corner points of the rectangle.

```

4782 \begin { pgfscope }
4783 \@@_pgf_rect_node:nnnnn
4784 { \@@_env: - #1 - #2 - block }
4785 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
4786 \end { pgfscope }

```

We construct the `short` node.

```

4787 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
4788 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4789 {

```

We recall that, when a cell is empty, no (normal) node is created in that cell. That’s why we test the existence of the node before using it.

```

4790 \cs_if_exist:cT
4791 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
4792 {
4793 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4794 {
4795 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
4796 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
4797 }
4798 }
4799 }

```

If all the cells of the column were empty, `\l_tmpb_dim` has still the same value `\c_max_dim`. In that case, you use for `\l_tmpb_dim` the value of the position of the vertical rule.

```

4800 \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
4801 {
4802 \@@_qpoint:n { col - #2 }
4803 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4804 }
4805 \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
4806 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4807 {
4808 \cs_if_exist:cT
4809 { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
4810 {
4811 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4812 {
4813 \pgfpointanchor
4814 { \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
4815 { east }
4816 \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
4817 }
4818 }
4819 }
4820 \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
4821 {
4822 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
4823 \dim_set_eq:NN \l_tmpd_dim \pgf@x
4824 }
4825 \@@_pgf_rect_node:nnnnn
4826 { \@@_env: - #1 - #2 - block - short }
4827 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpe_dim

```

If the creation of the “medium nodes” is required, we create a “medium node” for the block. The function `\@@_pgf_rect_node:nnnnn` takes in as arguments the name of the node and two PGF points.

```

4828 \bool_if:NT \l_@@_medium_nodes_bool
4829 {
4830 \@@_pgf_rect_node:nnn
4831 { \@@_env: - #1 - #2 - block - medium }
4832 { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
4833 {
4834 \pgfpointanchor
4835 { \@@_env:
4836 - \int_use:N \l_@@_last_row_int
4837 - \int_use:N \l_@@_last_col_int - medium
4838 }
4839 { south-east }
4840 }

```

```
4841 }
```

Now, we will put the label of the block beginning with the case of a `\Block` of one row.

```
4842 \int_compare:nNnTF { #1 } = { #3 }
4843 {
```

We take into account the case of a block of one row in the “first row” or the “last row”.

```
4844 \int_compare:nNnTF { #1 } = 0
4845 { \l_@@_code_for_first_row_tl }
4846 {
4847 \int_compare:nNnT { #1 } = \l_@@_last_row_int
4848 \l_@@_code_for_last_row_tl
4849 }
```

If the block has only one row, we want the label of the block perfectly aligned on the baseline of the row. That’s why we have constructed a `\pgfcoordinate` on the baseline of the row, in the first column of the array. Now, we retrieve the  $y$ -value of that node and we store it in `\l_tmpa_dim`.

```
4850 \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }
```

We retrieve (in `\pgf@x`) the  $x$ -value of the center of the block.

```
4851 \pgfpointanchor
4852 { \@@_env: - #1 - #2 - block - short }
4853 {
4854 \str_case:Vn \l_@@_pos_of_block_tl
4855 {
4856 c { center }
4857 l { west }
4858 r { east }
4859 }
4860 }
```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```
4861 \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
4862 \pgfset { inner-sep = \c_zero_dim }
4863 \pgfnode
4864 { rectangle }
4865 {
4866 \str_case:Vn \l_@@_pos_of_block_tl
4867 {
4868 c { base }
4869 l { base-west }
4870 r { base-east }
4871 }
4872 }
4873 { \box_use_drop:N \l_@@_cell_box } { } { }
4874 }
```

If the number of rows is different of 1, we will put the label of the block by using the short node (the label of the block has been composed in `\l_@@_cell_box`).

```
4875 {
```

If we are in the first column, we must put the block as if it was with the key `r`.

```
4876 \int_compare:nNnT { #2 } = 0
4877 { \tl_set:Nn \l_@@_pos_of_block_tl r }
4878 \bool_if:nT \g_@@_last_col_found_bool
4879 {
4880 \int_compare:nNnT { #2 } = \g_@@_col_total_int
4881 { \tl_set:Nn \l_@@_pos_of_block_tl l }
4882 }
4883 \pgftransformshift
4884 {
4885 \pgfpointanchor
4886 { \@@_env: - #1 - #2 - block - short }
4887 {
4888 \str_case:Vn \l_@@_pos_of_block_tl
```

```

4889         {
4890             c { center }
4891             l { west }
4892             r { east }
4893         }
4894     }
4895 }
4896 \pgfset { inner-sep = \c_zero_dim }
4897 \pgfnode
4898 { rectangle }
4899 {
4900     \str_case:Vn \l_@@_pos_of_block_tl
4901     {
4902         c { center }
4903         l { west }
4904         r { east }
4905     }
4906 }
4907 { \box_use_drop:N \l_@@_cell_box } { } { }
4908 }
4909 \endpgfpicture
4910 \group_end:
4911 }

```

The first argument of `\@@_stroke_block:nnn` is a list of options for the rectangle that you will stroke. The second argument is the upper-left cell of the block (with, as usual, the syntax *i-j*) and the third is the last cell of the block (with the same syntax).

```

4912 \cs_new_protected:Npn \@@_stroke_block:nnn #1 #2 #3
4913 {
4914     \group_begin:
4915     \tl_clear:N \l_@@_draw_tl
4916     \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
4917     \keys_set_known:nn { NiceMatrix / BlockStroke } { #1 }
4918     \pgfpicture
4919     \pgfrememberpicturepositiononpagetrue
4920     \pgf@relevantforpicturesizefalse
4921     \tl_if_empty:NF \l_@@_draw_tl
4922     {

```

If the user has used the key `color` of the command `\Block` without value, the color fixed by `\arrayrulecolor` is used.

```

4923         \str_if_eq:VnTF \l_@@_draw_tl { default }
4924         { \CT@arc@ }
4925         { \pgfsetstrokecolor { \l_@@_draw_tl } }
4926     }
4927     \@@_cut_on_hyphen:w #2 \q_stop
4928     \bool_lazy_and:nnT
4929     { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
4930     { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
4931     {
4932         \@@_qpoint:n { row - \l_tmpa_tl }
4933         \dim_set:Nn \l_tmpb_dim { \pgf@y }
4934         \@@_qpoint:n { col - \l_tmpb_tl }
4935         \dim_set:Nn \l_tmpc_dim { \pgf@x }
4936         \@@_cut_on_hyphen:w #3 \q_stop
4937         \int_compare:nNnT \l_tmpa_tl > \c@iRow
4938         { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
4939         \int_compare:nNnT \l_tmpb_tl > \c@jCol
4940         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
4941         \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
4942         \dim_set:Nn \l_tmpa_dim { \pgf@y }
4943         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
4944         \dim_set:Nn \l_tmpd_dim { \pgf@x }

```

```

4945     \pgfpathrectanglecorners
4946     { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4947     { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
4948     \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }
4949     \pgfusepathqstroke
4950   }
4951   \endpgfpicture
4952   \group_end:
4953 }

```

Here is the set of keys for the command `\@@_stroke_block:nnn`.

```

4954 \keys_define:nn { NiceMatrix / BlockStroke }
4955 {

```

We will uncomment the following line when we will give to the key `color` of the command `\Block` its new definition.

```

4956   % color .tl_set:N = \l_@@_draw_tl ,
4957   draw .tl_set:N = \l_@@_draw_tl ,
4958   draw .default:n = default ,
4959   line-width .dim_set:N = \l_@@_line_width_dim ,
4960 }

```

## How to draw the dotted lines transparently

```

4961 \cs_set_protected:Npn \@@_renew_matrix:
4962 {
4963   \RenewDocumentEnvironment { pmatrix } { } {
4964     { \pNiceMatrix }
4965     { \endpNiceMatrix }
4966   }
4967   \RenewDocumentEnvironment { vmatrix } { } {
4968     { \vNiceMatrix }
4969     { \endvNiceMatrix }
4970   }
4971   \RenewDocumentEnvironment { Vmatrix } { } {
4972     { \VNiceMatrix }
4973     { \endVNiceMatrix }
4974   }
4975   \RenewDocumentEnvironment { bmatrix } { } {
4976     { \bNiceMatrix }
4977     { \endbNiceMatrix }
4978   }
4979   \RenewDocumentEnvironment { Bmatrix } { } {
4980     { \BNiceMatrix }
4981     { \endBNiceMatrix }
4982   }

```

## Automatic arrays

```

4979 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
4980 {
4981   \int_set:Nn \l_@@_nb_rows_int { #1 }
4982   \int_set:Nn \l_@@_nb_cols_int { #2 }
4983 }
4984 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
4985 {
4986   \int_zero_new:N \l_@@_nb_rows_int
4987   \int_zero_new:N \l_@@_nb_cols_int
4988   \@@_set_size:n #4 \q_stop
4989   \begin { NiceArrayWithDelims } { #1 } { #2 }
4990     { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
4991   \int_compare:nNnT \l_@@_first_row_int = 0
4992   {
4993     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4994     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }

```



```

4995     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4996   }
4997   \prg_replicate:nn \l_@@_nb_rows_int
4998   {
4999     \int_compare:nNnT \l_@@_first_col_int = 0 { & }

```

You put { } before #6 to avoid a hasty expansion of a potential `\arabic{iRow}` at the beginning of the row which would result in an incorrect value of that `iRow` (since `iRow` is incremented in the first cell of the row of the `\halign`).

```

5000     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
5001     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
5002   }
5003   \int_compare:nNnT \l_@@_last_row_int > { -2 }
5004   {
5005     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
5006     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
5007     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
5008   }
5009   \end { NiceArrayWithDelims }
5010 }
5011 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
5012 {
5013   \cs_set_protected:cpn { #1 AutoNiceMatrix }
5014   {
5015     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
5016     \AutoNiceMatrixWithDelims { #2 } { #3 }
5017   }
5018 }
5019 \@@_define_com:nnn p ( )
5020 \@@_define_com:nnn b [ ]
5021 \@@_define_com:nnn v | |
5022 \@@_define_com:nnn V \ | \ |
5023 \@@_define_com:nnn B \{ \}

```

We define also an command `\AutoNiceMatrix` similar to the environment `{NiceMatrix}`.

```

5024 \NewDocumentCommand \AutoNiceMatrix { 0 { } m 0 { } m ! 0 { } }
5025 {
5026   \group_begin:
5027   \bool_set_true:N \l_@@_NiceArray_bool
5028   \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
5029   \group_end:
5030 }

```

## The redefinition of the command `\dotfill`

```

5031 \cs_set_eq:NN \@@_old_dotfill \dotfill
5032 \cs_new_protected:Npn \@@_dotfill:
5033 {

```

First, we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in case of use of `\dotfill` “internally” in the cell (e.g. `\hbox to 1cm {\dotfill}`).

```

5034   \@@_old_dotfill
5035   \bool_if:NT \l_@@_NiceTabular_bool
5036   { \group_insert_after:N \@@_dotfill_ii: }
5037   { \group_insert_after:N \@@_dotfill_i: }
5038 }
5039 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
5040 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }

```

Now, if the box is not empty (unfortunately, we can’t actually test whether the box is empty and that’s why we only consider it’s width), we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in the cell of the array, and it will extend, since it is no longer in `\l_@@_cell_box`.

```

5041 \cs_new_protected:Npn \@@_dotfill_iii:

```

```
5042 { \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } = \c_zero_dim \@@_old_dotfill }
```

## The command `\diagbox`

The command `\diagbox` will be linked to `\diagbox:nn` in the environments of `nicematrix`.

```
5043 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
5044 {
5045   \tl_gput_right:Nx \g_@@_internal_code_after_tl
5046   {
5047     \@@_actually_diagbox:nnnnnn
5048     { \int_use:N \c@iRow }
5049     { \int_use:N \c@jCol }
5050     { \int_use:N \c@iRow }
5051     { \int_use:N \c@jCol }
5052     { \exp_not:n { #1 } }
5053     { \exp_not:n { #2 } }
5054   }
```

We put the cell with `\diagbox` in the sequence `\g_@@_pos_of_blocks_seq` because a cell with `\diagbox` must be considered as non empty by the key `except-corners`.

```
5055   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
5056   {
5057     { \int_use:N \c@iRow }
5058     { \int_use:N \c@jCol }
5059     { \int_use:N \c@iRow }
5060     { \int_use:N \c@jCol }
5061   }
5062 }
```

The command `\diagbox` is also redefined locally when we draw a block.

The first four arguments of `\@@_actually_diagbox:nnnnnn` correspond to the rectangle (=block) to slash (we recall that it's possible to use `\diagbox` in a `\Block`). The two other are the elements to draw below and above the diagonal line.

```
5063 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
5064 {
5065   \pgfpicture
5066   \pgf@relevantforpicturesizefalse
5067   \pgfrememberpicturepositiononpagetrue
5068   \@@_qpoint:n { row - #1 }
5069   \dim_set_eq:NN \l_tmpa_dim \pgf@y
5070   \@@_qpoint:n { col - #2 }
5071   \dim_set_eq:NN \l_tmpb_dim \pgf@x
5072   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
5073   \@@_qpoint:n { row - \@@_succ:n { #3 } }
5074   \dim_set_eq:NN \l_tmpc_dim \pgf@y
5075   \@@_qpoint:n { col - \@@_succ:n { #4 } }
5076   \dim_set_eq:NN \l_tmpd_dim \pgf@x
5077   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
5078   {
```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded.

```
5079   \CT@arc@
5080   \pgfsetroundcap
5081   \pgfusepathqstroke
5082 }
5083 \pgfset { inner-sep = 1 pt }
5084 \pgfscope
5085 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
5086 \pgfnode { rectangle } { south-west }
5087 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
5088 \endpgfscope
```

```

5089 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
5090 \pgfnode { rectangle } { north-east }
5091 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
5092 \endpgfpicture
5093 }

```

## The keyword `\CodeAfter`

The `\CodeAfter` (inserted with the key `code-after` or after the keyword `\CodeAfter`) may always begin with a list of pairs *key-value* between square brackets. Here is the corresponding set of keys.

```

5094 \keys_define:nn { NiceMatrix }
5095 { CodeAfter / rules .inherit:n = NiceMatrix / rules }
5096 \keys_define:nn { NiceMatrix / CodeAfter }
5097 {
5098   sub-matrix .code:n = \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
5099   sub-matrix .value_required:n = true ,
5100   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
5101   delimiters / color .value_required:n = true ,
5102   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
5103   rules .value_required:n = true ,
5104   unknown .code:n = \@@_error:n { Unknown-key-for-CodeAfter }
5105 }

```

In fact, in this subsection, we define the user command `\CodeAfter` for the case of the “normal syntax”. For the case of “light-syntax”, see the definition of the environment `{@@-light-syntax}` on p. 92.

In the environments of `nicematrix`, `\CodeAfter` will be linked to `\@@_CodeAfter:`. That macro must *not* be protected since it begins with `\omit`.

```

5106 \cs_new:Npn \@@_CodeAfter: { \omit \@@_CodeAfter_i:n }

```

However, in each cell of the environment, the command `\CodeAfter` will be linked to the following command `\@@_CodeAfter_i:n` which do *not* begin with `\omit` (and thus, the user will be able to use `\CodeAfter` without error and without the need to prefix by `\omit`).

We have to catch everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```

5107 \cs_new_protected:Npn \@@_CodeAfter_i:n #1 \end
5108 {
5109   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
5110   \@@_CodeAfter_ii:n
5111 }

```

We catch the argument of the command `\end` (in `#1`).

```

5112 \cs_new_protected:Npn \@@_CodeAfter_ii:n #1
5113 {

```

If this is really the end of the current environment (of `nicematrix`), we put back the command `\end` and its argument in the TeX flow.

```

5114   \str_if_eq:eeTF \@@_currenvir { #1 } { \end { #1 } }

```

If this is not the `\end` we are looking for, we put those tokens in `\g_nicematrix_code_after_tl` and we go on searching for the next command `\end` with a recursive call to the command `\@@_CodeAfter:n`.

```

5115   {
5116     \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
5117     \@@_CodeAfter_i:n
5118   }
5119 }

```

## The delimiters in the preamble

The command `\@@_delimiter:nnn` will be used to draw delimiters inside the matrix when delimiters are specified in the preamble of the array. It does *not* concern the exterior delimiters added by `{NiceArrayWithDelims}` (and `{pNiceArray}`, `{pNiceMatrix}`, etc.).

A delimiter in the preamble of the array will write an instruction `\@@_delimiter:nnn` in the `\g_@@_internal_code_after_tl` (and also potentially add instructions in the preamble provided to `\array` in order to add spaces between columns).

The first argument is the type of delimiter (`(`, `[`, `\{`, `)`, `]` ou `\}`). The second argument is the number of column. The third argument is a boolean equal to `\c_true_bool` (resp. `\c_false_true`) when the delimiter must be put on the left (resp. right) side.

```

5120 \cs_new_protected:Npn \@@_delimiter:nnn #1 #2 #3
5121 {
5122   \pgfpicture
5123   \pgfrememberpicturepositiononpagetrue
5124   \pgf@relevantforpicturesizefalse

```

`\l_@@_y_initial_dim` and `\l_@@_y_final_dim` will be the  $y$ -values of the extremities of the delimiter we will have to construct.

```

5125   \@@_qpoint:n { row - 1 }
5126   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
5127   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
5128   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y

```

We will compute in `\l_tmpa_dim` the  $x$ -value where we will have to put our delimiter (on the left side or on the right side).

```

5129   \bool_if:nTF { #3 }
5130   { \dim_set_eq:NN \l_tmpa_dim \c_max_dim }
5131   { \dim_set:Nn \l_tmpa_dim { - \c_max_dim } }
5132   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5133   {
5134     \cs_if_exist:cT
5135     { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
5136     {
5137       \pgfpointanchor
5138       { \@@_env: - ##1 - #2 }
5139       { \bool_if:nTF { #3 } { west } { east } }
5140       \dim_set:Nn \l_tmpa_dim
5141       { \bool_if:nTF { #3 } \dim_min:nn \dim_max:nn \l_tmpa_dim \pgf@x }
5142     }
5143   }

```

Now we can put the delimiter with a node of PGF.

```

5144   \pgfset { inner~sep = \c_zero_dim }
5145   \dim_zero:N \nulldelimiterspace
5146   \pgftransformshift
5147   {
5148     \pgfpoint
5149     { \l_tmpa_dim }
5150     { ( \l_@@_y_initial_dim + \l_@@_y_final_dim + \arrayrulewidth ) / 2 }
5151   }
5152   \pgfnode
5153   { rectangle }
5154   { \bool_if:nTF { #3 } { east } { west } }
5155   {

```

Here is the content of the PGF node, that is to say the delimiter, constructed with its right size.

```

5156   \nullfont
5157   \c_math_toggle_token
5158   \tl_if_empty:NF \l_@@_delimiters_color_tl
5159   { \color { \l_@@_delimiters_color_tl } }
5160   \bool_if:nTF { #3 } { \left #1 } { \left . }

```

```

5161     \vcenter
5162     {
5163         \nullfont
5164         \hrule \@height
5165             \dim_eval:n { \l_@@_y_initial_dim - \l_@@_y_final_dim }
5166             \@depth \c_zero_dim
5167             \@width \c_zero_dim
5168     }
5169     \bool_if:nTF { #3 } { \right . } { \right #1 }
5170     \c_math_toggle_token
5171 }
5172 { }
5173 { }
5174 \endpgfpicture
5175 }

```

## The command `\SubMatrix`

```

5176 \keys_define:nn { NiceMatrix / sub-matrix }
5177 {
5178     extra-height .dim_set:N = \l_@@_submatrix_extra_height_dim ,
5179     extra-height .value_required:n = true ,
5180     left-xshift .dim_set:N = \l_@@_submatrix_left_xshift_dim ,
5181     left-xshift .value_required:n = true ,
5182     right-xshift .dim_set:N = \l_@@_submatrix_right_xshift_dim ,
5183     right-xshift .value_required:n = true ,
5184     xshift .meta:n = { left-xshift = #1, right-xshift = #1 } ,
5185     xshift .value_required:n = true ,
5186     delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
5187     delimiters / color .value_required:n = true ,
5188     slim .bool_set:N = \l_@@_submatrix_slim_bool ,
5189     slim .default:n = true ,
5190 }
5191 \keys_define:nn { NiceMatrix }
5192 {
5193     SubMatrix .inherit:n = NiceMatrix / sub-matrix ,
5194     CodeAfter / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5195 }

```

The following keys set is for the command `\SubMatrix` itself (not the tuning of `\SubMatrix` that can be done elsewhere).

```

5196 \keys_define:nn { NiceMatrix / SubMatrix }
5197 {
5198     hlines .clist_set:N = \l_@@_submatrix_hlines_clist ,
5199     hlines .default:n = all ,
5200     vlines .clist_set:N = \l_@@_submatrix_vlines_clist ,
5201     vlines .default:n = all ,
5202     name .code:n =
5203         \tl_if_empty:nTF { #1 }
5204         { \@@_error:n { Invalid-name-format } }
5205         {
5206             \regex_match:nnTF { \A[A-Za-z][A-Za-z0-9]*\Z } { #1 }
5207             {
5208                 \seq_if_in:NnTF \g_@@_submatrix_names_seq { #1 }
5209                 { \@@_error:nn { Duplicate-name-for-SubMatrix } { #1 } }
5210                 {
5211                     \str_set:Nn \l_@@_submatrix_name_str { #1 }
5212                     \seq_gput_right:Nn \g_@@_submatrix_names_seq { #1 }
5213                 }
5214             }
5215             { \@@_error:n { Invalid-name-format } }
5216         } ,
5217     rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,

```

```

5218     rules .value_required:n = true ,
5219     name .value_required:n = true ,
5220     unknown .code:n = \@@_error:n { Unknown-key-for-SubMatrix }
5221 }

```

In the code-after, the following command `\@@_SubMatrix` will be linked to `\SubMatrix`.

- #1 is the left delimiter;
- #2 is the upper-left cell of the matrix with the format  $i-j$ ;
- #3 is the lower-right cell of the matrix with the format  $i-j$ ;
- #4 is the right delimiter;
- #5 is the list of options of the command.

```

5222 \NewDocumentCommand \@@_SubMatrix { m m m m ! O { } }
5223 {
5224     \group_begin:

```

The command `\@@_cut_on_hyphen:w` cuts on the hyphen an argument of the form  $i-j$ . The value of  $i$  is stored in `\l_tmpa_tl` and the value of  $j$  is stored in `\l_tmpb_tl`.

```

5225     \@@_cut_on_hyphen:w #3 \q_stop
5226     \tl_clear_new:N \l_tmpc_tl
5227     \tl_clear_new:N \l_tmpd_tl
5228     \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
5229     \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
5230     \@@_cut_on_hyphen:w #2 \q_stop
5231     \bool_lazy_or:nnTF
5232     { \int_compare_p:nNn \l_tmpc_tl > \g_@@_row_total_int }
5233     { \int_compare_p:nNn \l_tmpd_tl > \g_@@_col_total_int }
5234     { \@@_error:n { SubMatrix-too-large } }
5235     {
5236         \str_clear_new:N \l_@@_submatrix_name_str
5237         \keys_set:nn { NiceMatrix / SubMatrix } { #5 }
5238         \pgfpicture
5239         \pgfrememberpicturepositiononpagetrue
5240         \pgf@relevantforpicturesizefalse
5241         \pgfset { inner-sep = \c_zero_dim }
5242         \dim_set_eq:NN \l_@@_x_initial_dim \c_max_dim
5243         \dim_set:Nn \l_@@_x_final_dim { - \c_max_dim }

```

The last value of `\int_step_inline:nnn` is provided by currying.

```

5244     \bool_if:NTF \l_@@_submatrix_slim_bool
5245     { \int_step_inline:nnn \l_tmpa_tl \l_tmpc_tl }
5246     { \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int }
5247     {
5248         \cs_if_exist:cT
5249         { pgf @ sh @ ns @ \@@_env: - ##1 - \l_tmpb_tl }
5250         {
5251             \pgfpointanchor { \@@_env: - ##1 - \l_tmpb_tl } { west }
5252             \dim_set:Nn \l_@@_x_initial_dim
5253             { \dim_min:nn \l_@@_x_initial_dim \pgf@x }
5254         }
5255         \cs_if_exist:cT
5256         { pgf @ sh @ ns @ \@@_env: - ##1 - \l_tmpd_tl }
5257         {
5258             \pgfpointanchor { \@@_env: - ##1 - \l_tmpd_tl } { east }
5259             \dim_set:Nn \l_@@_x_final_dim
5260             { \dim_max:nn \l_@@_x_final_dim \pgf@x }
5261         }
5262     }
5263     \@@_qpoint:n { row - \l_tmpa_tl - base }
5264     \dim_set:Nn \l_@@_y_initial_dim
5265     { \pgf@y + ( \box_ht:N \strutbox + \extrarowheight ) * \arraystretch }

```

```

5266 \@@_qpoint:n { row - \l_tmpc_tl - base }
5267 \dim_set:Nn \l_@@_y_final_dim
5268 { \pgf@y - ( \box_dp:N \strutbox ) * \arraystretch }
5269 \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
5270 {
5271   \cs_if_exist:cT
5272   { \pgf @ sh @ ns @ \@@_env: - \l_tmpa_tl - ##1 }
5273   {
5274     \pgfpointanchor { \@@_env: - \l_tmpa_tl - ##1 } { north }
5275     \dim_set:Nn \l_@@_y_initial_dim
5276     { \dim_max:nn \l_@@_y_initial_dim \pgf@y }
5277   }
5278   \cs_if_exist:cT
5279   { \pgf @ sh @ ns @ \@@_env: - \l_tmpc_tl - ##1 }
5280   {
5281     \pgfpointanchor { \@@_env: - \l_tmpc_tl - ##1 } { south }
5282     \dim_set:Nn \l_@@_y_final_dim
5283     { \dim_min:nn \l_@@_y_final_dim \pgf@y }
5284   }
5285 }
5286 \dim_set:Nn \l_tmpa_dim
5287 {
5288   \l_@@_y_initial_dim - \l_@@_y_final_dim +
5289   \l_@@_submatrix_extra_height_dim - \arrayrulewidth
5290 }
5291 \dim_set_eq:NN \nulldelimiterspace \c_zero_dim

```

We will draw the rules in the `\SubMatrix`.

```

5292 \group_begin:
5293 \pgfsetlinewidth { 1.1 \arrayrulewidth }
5294 \tl_if_empty:NF \l_@@_rules_color_tl
5295 { \exp_after:wN \@@_set_CT@arc@: \l_@@_rules_color_tl \q_stop }
5296 \CT@arc@

```

Now, we draw the potential vertical rules specified in the preamble of the environments with the letter fixed with the key `vlines-in-sub-matrix`. The list of the columns where there is such rule to draw is in `\g_@@_cols_vlism_seq`.

```

5297 \seq_map_inline:Nn \g_@@_cols_vlism_seq
5298 {
5299   \int_compare:nNnT \l_tmpb_tl < { ##1 }
5300   {
5301     \int_compare:nNnT { ##1 } < { \int_eval:n { \l_tmpd_tl + 1 } }
5302     {

```

First, we extract the value of the abscissa of the rule we have to draw.

```

5303 \@@_qpoint:n { col - ##1 }
5304 \pgfpathmoveto { \pgfpoint \pgf@x \l_@@_y_initial_dim }
5305 \pgfpathlineto { \pgfpoint \pgf@x \l_@@_y_final_dim }
5306 \pgfusepathqstroke
5307 }
5308 }
5309 }

```

Now, we draw the vertical rules specified in the key `vlines` of `\SubMatrix`. The last argument of `\int_step_inline:nn` or `\clist_map_inline:Nn` is given by curryfication.

```

5310 \tl_if_eq:NnTF \l_@@_submatrix_vlines_clist { all }
5311 { \int_step_inline:nn { \l_tmpd_tl - \l_tmpb_tl } }
5312 { \clist_map_inline:Nn \l_@@_submatrix_vlines_clist }
5313 {
5314   \bool_lazy_and:nnTF
5315   { \int_compare_p:nNn { ##1 } > 0 }
5316   { \int_compare_p:nNn { ##1 } < { \l_tmpd_tl - \l_tmpb_tl + 1 } }
5317   {

```

```

5318         \@@_qpoint:n { col - \int_eval:n { ##1 + \l_tmpb_tl } }
5319         \pgfpathmoveto { \pgfpoint \pgf@x \l_@@_y_initial_dim }
5320         \pgfpathlineto { \pgfpoint \pgf@x \l_@@_y_final_dim }
5321         \pgfusepathqstroke
5322     }
5323     { \@@_error:nnn { Wrong-line-in-SubMatrix } { vertical } { ##1 } }
5324 }

```

Now, we draw the horizontal rules specified in the key `hlines` of `\SubMatrix`. The last argument of `\int_step_inline:nn` or `\clist_map_inline:Nn` is given by curryfication.

```

5325 \tl_if_eq:NnTF \l_@@_submatrix_hlines_clist { all }
5326 { \int_step_inline:nn { \l_tmpc_tl - \l_tmpa_tl } }
5327 { \clist_map_inline:Nn \l_@@_submatrix_hlines_clist }
5328 {
5329     \bool_lazy_and:nnTF
5330     { \int_compare_p:nNn { ##1 } > 0 }
5331     { \int_compare_p:nNn { ##1 } < { \l_tmpc_tl - \l_tmpa_tl + 1 } }
5332     {
5333         \@@_qpoint:n { row - \int_eval:n { ##1 + \l_tmpa_tl } }

```

We use a group to protect `\l_tmpa_dim` and `\l_tmpb_dim`.

```

5334     \group_begin:

```

We compute in `\l_tmpa_dim` the  $x$ -value of the left end of the rule.

```

5335         \dim_set:Nn \l_tmpa_dim
5336         { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
5337         \str_case:nn { #1 }
5338         {
5339             ( { \dim_sub:Nn \l_tmpa_dim { 0.9 mm } }
5340             [ { \dim_sub:Nn \l_tmpa_dim { 0.2 mm } }
5341             \{ { \dim_sub:Nn \l_tmpa_dim { 0.9 mm } }
5342         }
5343         \pgfpathmoveto { \pgfpoint \l_tmpa_dim \pgf@y }

```

We compute in `\l_tmpb_dim` the  $x$ -value of the right end of the rule.

```

5344         \dim_set:Nn \l_tmpb_dim
5345         { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
5346         \str_case:nn { #4 }
5347         {
5348             ) { \dim_add:Nn \l_tmpb_dim { 0.9 mm } }
5349             ] { \dim_add:Nn \l_tmpb_dim { 0.2 mm } }
5350             \} { \dim_add:Nn \l_tmpb_dim { 0.9 mm } }
5351         }
5352         \pgfpathlineto { \pgfpoint \l_tmpb_dim \pgf@y }
5353         \pgfusepathqstroke
5354         \group_end:
5355     }
5356     { \@@_error:nnn { Wrong-line-in-SubMatrix } { horizontal } { ##1 } }
5357 }

```

If the key `name` has been used for the command `\SubMatrix`, we create a PGF node with that name for the submatrix (this node does not take into account the delimiters that we will put after).

```

5358     \str_if_empty:NF \l_@@_submatrix_name_str
5359     {
5360         \@@_pgf_rect_node:nnnnn \l_@@_submatrix_name_str
5361         \l_@@_x_initial_dim \l_@@_y_initial_dim
5362         \l_@@_x_final_dim \l_@@_y_final_dim
5363     }
5364     \group_end:

```

The group was for `\CT@arc@` (the color of the rules).

Now, we deal with the left delimiter. Of course, the environment `{pgfscope}` is for the `\pgftransformshift`.

```

5365     \begin { pgfscope }

```



```

5366 \pgftransformshift
5367 {
5368   \pgfpoint
5369   { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
5370   { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
5371 }
5372 \str_if_empty:NTF \l_@@_submatrix_name_str
5373 { \@@_node_left:nn #1 { } }
5374 { \@@_node_left:nn #1 { \@@_env: - \l_@@_submatrix_name_str - left } }
5375 \end { pgfscope }

```

Now, we deal with the right delimiter.

```

5376 \pgftransformshift
5377 {
5378   \pgfpoint
5379   { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
5380   { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
5381 }
5382 \str_if_empty:NTF \l_@@_submatrix_name_str
5383 { \@@_node_right:nn #4 { } }
5384 {
5385   \@@_node_right:nn #4 { \@@_env: - \l_@@_submatrix_name_str - right }
5386 }
5387 \endpgfpicture
5388 }
5389 \group_end:
5390 }

```

The command `\@@_node_left:nn` puts the left delimiter with the correct size. The argument #1 is the delimiter to put. The argument #2 is the name we will give to this PGF node (if the key `name` has been used in `\SubMatrix`).

```

5391 \cs_new_protected:Npn \@@_node_left:nn #1 #2
5392 {
5393   \pgfnode
5394   { rectangle }
5395   { east }
5396   {
5397     \nullfont
5398     \c_math_toggle_token
5399     \tl_if_empty:NF \l_@@_delimiters_color_tl
5400     { \color { \l_@@_delimiters_color_tl } }
5401     \left #1
5402     \vcenter
5403     {
5404       \nullfont
5405       \hrule \@height \l_tmpa_dim
5406       \@depth \c_zero_dim
5407       \@width \c_zero_dim
5408     }
5409     \right .
5410     \c_math_toggle_token
5411   }
5412   { #2 }
5413   { }
5414 }

```

The command `\@@_node_right:nn` puts the right delimiter with the correct size. The argument #1 is the delimiter to put. The argument #2 is the name we will give to this PGF node (if the key `name` has been used in `\SubMatrix`).

```

5415 \cs_new_protected:Npn \@@_node_right:nn #1 #2
5416 {
5417   \pgfnode
5418   { rectangle }

```

```

5419 { west }
5420 {
5421   \nullfont
5422   \c_math_toggle_token
5423   \tl_if_empty:NF \l_@@_delimiters_color_tl
5424   { \color { \l_@@_delimiters_color_tl } }
5425   \left .
5426   \vcenter
5427   {
5428     \nullfont
5429     \hrule \@height \l_tmpa_dim
5430             \@depth \c_zero_dim
5431             \@width \c_zero_dim
5432   }
5433   \right #1
5434   \c_math_toggle_token
5435 }
5436 { #2 }
5437 { }
5438 }

```

## We process the options at package loading

We process the options when the package is loaded (with `\usepackage`) but we recommend to use `\NiceMatrixOptions` instead.

We must process these options after the definition of the environment `{NiceMatrix}` because the option `renew-matrix` executes the code `\cs_set_eq:NN \env@matrix \NiceMatrix`.

Of course, the command `\NiceMatrix` must be defined before such an instruction is executed.

The boolean `\g_@@_footnotehyper_bool` will indicate if the option `footnotehyper` is used.

```

5439 \bool_new:N \c_@@_footnotehyper_bool

```

The boolean `\c_@@_footnote_bool` will indicate if the option `footnote` is used, but quickly, it will also be set to true if the option `footnotehyper` is used.

```

5440 \bool_new:N \c_@@_footnote_bool
5441 \@@_msg_new:nnn { Unknown~option~for~package }
5442 {
5443   The~key~'\l_keys_key_str'~is~unknown. \\
5444   If~you~go~on,~it~will~be~ignored. \\
5445   For~a~list~of~the~available~keys,~type~H~<return>.
5446 }
5447 {
5448   The~available~keys~are~(in~alphabetic~order):~
5449   define-L-C-R,~
5450   footnote,~
5451   footnotehyper,~
5452   renew-dots,~and
5453   renew-matrix.
5454 }
5455 \@@_msg_new:nn { Key~transparent }
5456 {
5457   The~key~'transparent'~is~now~obsolete~(because~it's~name~
5458   is~not~clear).~You~should~use~the~conjunction~of~'renew-dots'~
5459   and~'renew-matrix'.~However,~you~can~go~on.
5460 }
5461 \keys_define:nn { NiceMatrix / Package }
5462 {
5463   define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
5464   define-L-C-R .default:n = true ,
5465   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,

```

```

5466   renew-dots .value_forbidden:n = true ,
5467   renew-matrix .code:n = \@@_renew_matrix: ,
5468   renew-matrix .value_forbidden:n = true ,
5469   transparent .code:n =
5470   {
5471     \@@_renew_matrix:
5472     \bool_set_true:N \l_@@_renew_dots_bool
5473     \@@_error:n { Key~transparent }
5474   } ,
5475   transparent .value_forbidden:n = true,
5476   footnote .bool_set:N = \c_@@_footnote_bool ,
5477   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
5478   unknown .code:n = \@@_error:n { Unknown~option~for~package }
5479 }
5480 \ProcessKeysOptions { NiceMatrix / Package }

5481 \@@_msg_new:nn { footnote~with~footnotehyper~package }
5482 {
5483   You~can't~use~the~option~'footnote'~because~the~package~
5484   footnotehyper~has~already~been~loaded.~
5485   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
5486   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
5487   of~the~package~footnotehyper.\\
5488   If~you~go~on,~the~package~footnote~won't~be~loaded.
5489 }

5490 \@@_msg_new:nn { footnotehyper~with~footnote~package }
5491 {
5492   You~can't~use~the~option~'footnotehyper'~because~the~package~
5493   footnote~has~already~been~loaded.~
5494   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
5495   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
5496   of~the~package~footnote.\\
5497   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
5498 }

5499 \bool_if:NT \c_@@_footnote_bool
5500 {

```

The class beamer has its own system to extract footnotes and that's why we have nothing to do if beamer is used.

```

5501   \@ifclassloaded { beamer }
5502   { \bool_set_false:N \c_@@_footnote_bool }
5503   {
5504     \@ifpackageloaded { footnotehyper }
5505     { \@@_error:n { footnote~with~footnotehyper~package } }
5506     { \usepackage { footnote } }
5507   }
5508 }

5509 \bool_if:NT \c_@@_footnotehyper_bool
5510 {

```

The class beamer has its own system to extract footnotes and that's why we have nothing to do if beamer is used.

```

5511   \@ifclassloaded { beamer }
5512   { \bool_set_false:N \c_@@_footnote_bool }
5513   {
5514     \@ifpackageloaded { footnote }
5515     { \@@_error:n { footnotehyper~with~footnote~package } }
5516     { \usepackage { footnotehyper } }
5517   }
5518   \bool_set_true:N \c_@@_footnote_bool
5519 }

```

The flag `\c_@@_footnote_bool` is raised and so, we will only have to test `\c_@@_footnote_bool` in order to know if we have to insert an environment `{savenotes}`.

## Error messages of the package

```

5520 \seq_new:N \c_@@_types_of_matrix_seq
5521 \seq_set_from_clist:Nn \c_@@_types_of_matrix_seq
5522 {
5523   NiceMatrix ,
5524   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
5525 }
5526 \seq_set_map_x:NNn \c_@@_types_of_matrix_seq \c_@@_types_of_matrix_seq
5527 { \tl_to_str:n { #1 } }

```

If the user uses too much columns, the command `\@@_error_too_much_cols:` is executed. This command raises an error but try to give the best information to the user in the error message. The command `\seq_if_in:NVTF` is not expandable and that's why we can't put it in the error message itself. We have to do the test before the `\@@_fatal:n`.

```

5528 \cs_new_protected:Npn \@@_error_too_much_cols:
5529 {
5530   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
5531   {
5532     \int_compare:nNnTF \l_@@_last_col_int = { -2 }
5533     { \@@_fatal:n { too-much-cols-for-matrix } }
5534     {
5535       \bool_if:NF \l_@@_last_col_without_value_bool
5536       { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
5537     }
5538   }
5539   { \@@_fatal:n { too-much-cols-for-array } }
5540 }

```

The following command must *not* be protected since it's used in an error message.

```

5541 \cs_new:Npn \@@_message_hdotsfor:
5542 {
5543   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
5544   { ~Maybe-your-use-of~\token_to_str:N \Hdotsfor\ is-incorrect.}
5545 }
5546 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
5547 {
5548   You-try-to-use-more-columns-than-allowed-by-your~
5549   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of~
5550   columns-is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus-the~
5551   exterior-columns).~This-error-is-fatal.
5552 }
5553 \@@_msg_new:nn { too-much-cols-for-matrix }
5554 {
5555   You-try-to-use-more-columns-than-allowed-by-your~
5556   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall-that-the-maximal~
5557   number-of-columns-for-a-matrix-is-fixed-by-the-LaTeX-counter~
5558   'MaxMatrixCols'.~Its-actual-value-is~\int_use:N \c@MaxMatrixCols.~
5559   This-error-is-fatal.
5560 }

```

For the following message, remind that the test is not done after the construction of the array but in each row. That's why we have to put `\c@jCol-1` and not `\c@jCol`.

```

5561 \@@_msg_new:nn { too-much-cols-for-array }
5562 {
5563   You-try-to-use-more-columns-than-allowed-by-your~
5564   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of-columns-is~
5565   \int_use:N \g_@@_static_num_of_col_int\

```

```

5566 ~-(plus~the~potential~exterior~ones).~
5567 This~error~is~fatal.
5568 }
5569 \@@_msg_new:nn { last~col~not~used }
5570 {
5571   The~key~'last~col'~is~in~force~but~you~have~not~used~that~last~column~
5572   in~your~\@@_full_name_env:~.~However,~you~can~go~on.
5573 }
5574 \@@_msg_new:nn { columns~not~used }
5575 {
5576   The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
5577   \g_@@_static_num_of_col_int\ columns~but~you~use~only~\int_use:N \c@jCol.\\
5578   You~can~go~on~but~the~columns~you~did~not~used~won't~be~created.
5579 }
5580 \@@_msg_new:nn { in~first~col }
5581 {
5582   You~can't~use~the~command~#1 in~the~first~column~(number~0)~of~the~array.\\
5583   If~you~go~on,~this~command~will~be~ignored.
5584 }
5585 \@@_msg_new:nn { in~last~col }
5586 {
5587   You~can't~use~the~command~#1 in~the~last~column~(exterior)~of~the~array.\\
5588   If~you~go~on,~this~command~will~be~ignored.
5589 }
5590 \@@_msg_new:nn { in~first~row }
5591 {
5592   You~can't~use~the~command~#1 in~the~first~row~(number~0)~of~the~array.\\
5593   If~you~go~on,~this~command~will~be~ignored.
5594 }
5595 \@@_msg_new:nn { in~last~row }
5596 {
5597   You~can't~use~the~command~#1 in~the~last~row~(exterior)~of~the~array.\\
5598   If~you~go~on,~this~command~will~be~ignored.
5599 }
5600 \@@_msg_new:nn { bad~option~for~line~style }
5601 {
5602   Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line~style'~
5603   is~'standard'.~If~you~go~on,~this~key~will~be~ignored.
5604 }
5605 \@@_msg_new:nn { Unknown~key~for~xdots }
5606 {
5607   As~for~now,~there~is~only~three~key~available~here:~'color',~'line~style'~
5608   and~'shorten'~(and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
5609   this~key~will~be~ignored.
5610 }
5611 \@@_msg_new:nn { Unknown~key~for~rowcolors }
5612 {
5613   As~for~now,~there~is~only~two~keys~available~here:~'cols'~and~'respect~blocks'~
5614   (and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
5615   this~key~will~be~ignored.
5616 }
5617 \@@_msg_new:nn { Key~color~for~Block }
5618 {
5619   The~key~'color'~for~the~command~\token_to_str:N \Block\
5620   is~deprecated:~you~should~use~'fill'~instead.\\
5621   This~error~is~fatal.
5622 }
5623 }
5624 \@@_msg_new:nn { ampersand~in~light~syntax }

```

```

5625 {
5626     You~can't~use~an~ampersand~(\token_to_str:N &)~to~separate~columns~because~
5627     ~you~have~used~the~key~'light-syntax'.~This~error~is~fatal.
5628 }

5629 \@@_msg_new:nn { SubMatrix~too~large }
5630 {
5631     Your~command~\token_to_str:N \SubMatrix\
5632     can't~be~drawn~because~your~matrix~is~too~small.\\
5633     If~you~go~on,~this~command~will~be~ignored.
5634 }

5635 \@@_msg_new:nn { double-backslash-in-light-syntax }
5636 {
5637     You~can't~use~\token_to_str:N \\~to~separate~rows~because~you~have~used~
5638     the~key~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
5639     (set~by~the~key~'end-of-row').~This~error~is~fatal.
5640 }

5641 \@@_msg_new:nn { standard-cline~in~document }
5642 {
5643     The~key~'standard-cline'~is~available~only~in~the~preamble.\\
5644     If~you~go~on~this~command~will~be~ignored.
5645 }

5646 \@@_msg_new:nn { old~column~type }
5647 {
5648     The~column~type~'#1'~is~no~longer~defined~in~'nicematrix'.~
5649     Since~version~5.0,~you~have~to~use~'l',~'c'~and~'r'~instead~of~'L',~
5650     'C'~and~'R'.~You~can~also~use~the~key~'define-L-C-R'.\\
5651     This~error~is~fatal.
5652 }

5653 \@@_msg_new:nn { bad~value~for~baseline }
5654 {
5655     The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
5656     valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
5657     \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\\
5658     If~you~go~on,~a~value~of~1~will~be~used.
5659 }

5660 \@@_msg_new:nn { Invalid-name~format }
5661 {
5662     You~can't~give~the~name~'\l_keys_value_tl'~to~a~\token_to_str:N
5663     \SubMatrix.\\
5664     A~name~must~be~accepted~by~the~regular~expression~[A-Za-z][A-Za-z0-9]*.\\
5665     If~you~go~on,~this~key~will~be~ignored.
5666 }

5667 \@@_msg_new:nn { Wrong~line~in~SubMatrix }
5668 {
5669     You~try~to~draw~a~#1~line~of~number~'#2'~in~a~
5670     \token_to_str:N \SubMatrix\ of~your~\@@_full_name_env:\ but~that~
5671     number~is~not~valid.~If~you~go~on,~it~will~be~ignored.
5672 }

5673 \@@_msg_new:nn { empty~environment }
5674 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }

5675 \@@_msg_new:nn { Delimiter~with~small }
5676 {
5677     You~can't~put~a~delimiter~in~the~preamble~of~your~\@@_full_name_env:\
5678     because~the~key~'small'~is~in~force.\\
5679     This~error~is~fatal.
5680 }

5681 \@@_msg_new:nn { unknown~cell~for~line~in~CodeAfter }
5682 {
5683     Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~

```

```

5684     can't-be-executed-because-a-cell-doesn't-exist.\\
5685     If-you-go-on-this-command-will-be-ignored.
5686 }

5687 \@@_msg_new:nnn { Duplicate-name-for-SubMatrix }
5688 {
5689     The-name-#1'-is-already-used-for-a-token_to_str:N \SubMatrix\
5690     in-this-@@_full_name_env:.\
5691     If-you-go-on,~this-key-will-be-ignored.\\
5692     For-a-list-of-the-names-already-used,~type-H<return>.
5693 }
5694 {
5695     The-names-already-defined-in-this-@@_full_name_env:\ are:-
5696     \seq_use:Nnnn \g_@@_submatrix_names_seq { ~and~ } { ,~ } { ~and~ }.
5697 }

5698 \@@_msg_new:nn { Hdotsfor-in-col-0 }
5699 {
5700     You-can't-use-token_to_str:N \Hdotsfor\ in-an-exterior-column-of-
5701     the-array.~This-error-is-fatal.
5702 }

5703 \@@_msg_new:nn { bad-corner }
5704 {
5705     #1-is-an-incorrect-specification-for-a-corner-(in-the-keys-
5706     'except-corners'~and~'hvlines-except-corners').~The-available-
5707     values-are:~NW,~SW,~NE-and-SE.\\
5708     If-you-go-on,~this-specification-of-corner-will-be-ignored.
5709 }

5710 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
5711 {
5712     In-the-@@_full_name_env:,~you-must-use-the-key-
5713     'last-col'~without-value.\\
5714     However,~you-can-go-on-for-this-time-
5715     (the-value-\l_keys_value_tl'-will-be-ignored).
5716 }

5717 \@@_msg_new:nn { last-col-non-empty-for-NiceMatrixOptions }
5718 {
5719     In\NiceMatrixoptions,~you-must-use-the-key-
5720     'last-col'~without-value.\\
5721     However,~you-can-go-on-for-this-time-
5722     (the-value-\l_keys_value_tl'-will-be-ignored).
5723 }

5724 \@@_msg_new:nn { Block-too-large-1 }
5725 {
5726     You-try-to-draw-a-block-in-the-cell-#1-#2-of-your-matrix-but-the-matrix-is-
5727     too-small-for-that-block. \\
5728 }

5729 \@@_msg_new:nn { Block-too-large-2 }
5730 {
5731     The-preamble-of-your-@@_full_name_env:\ announces~\int_use:N
5732     \g_@@_static_num_of_col_int\
5733     columns~but-you-use-only~\int_use:N \c@jCol\ and-that's-why-a-block-
5734     specified-in-the-cell-#1-#2-can't-be-drawn.~You-should-add-some-ampersands~
5735     (&)~at-the-end-of-the-first-row-of-your-
5736     \@@_full_name_env:.\
5737     If-you-go-on,this-block-and-maybe-others-will-be-ignored.
5738 }

5739 \@@_msg_new:nn { unknown-column-type }
5740 {
5741     The-column-type-#1'-in-your-@@_full_name_env:\
5742     is-unknown. \\
5743     This-error-is-fatal.

```

```

5744 }
5745 \@@_msg_new:nn { tabularnote-forbidden }
5746 {
5747   You~can't~use~the~command~\token_to_str:N\tabularnote\
5748   ~in~a~\@@_full_name_env:~.~This~command~is~available~only~in~
5749   \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
5750   If~you~go~on,~this~command~will~be~ignored.
5751 }
5752 \@@_msg_new:nn { bottomrule-without-booktabs }
5753 {
5754   You~can't~use~the~key~'tabular/bottomrule'~because~you~haven't~
5755   loaded~'booktabs'. \\
5756   If~you~go~on,~this~key~will~be~ignored.
5757 }
5758 \@@_msg_new:nn { enumitem-not-loaded }
5759 {
5760   You~can't~use~the~command~\token_to_str:N\tabularnote\
5761   ~because~you~haven't~loaded~'enumitem'. \\
5762   If~you~go~on,~this~command~will~be~ignored.
5763 }
5764 \@@_msg_new:nn { Wrong-last-row }
5765 {
5766   You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
5767   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
5768   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
5769   last~row.~You~can~avoid~this~problem~by~using~'last-row'~
5770   without~value~(more~compilations~might~be~necessary).
5771 }
5772 \@@_msg_new:nn { Yet~in~env }
5773 { Environments~of~nicematrix~can't~be~nested. \\ This~error~is~fatal. }
5774 \@@_msg_new:nn { Outside-math-mode }
5775 {
5776   The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
5777   (and~not~in~\token_to_str:N \vcenter). \\
5778   This~error~is~fatal.
5779 }
5780 \@@_msg_new:nn { One-letter-allowed }
5781 {
5782   The~value~of~key~'\l_keys_key_str'~must~be~of~length~1. \\
5783   If~you~go~on,~it~will~be~ignored.
5784 }
5785 \@@_msg_new:nnn { Unknown-key-for-Block }
5786 {
5787   The~key~'\l_keys_key_str'~is~unknown~for~the~command~\token_to_str:N
5788   \Block. \\ If~you~go~on,~it~will~be~ignored. \\
5789   For~a~list~of~the~available~keys,~type~H<return>.
5790 }
5791 {
5792   The~available~keys~are~(in~alphabetic~order):~,~,~c,~draw,~fill,~l,~
5793   line-width~and~r.
5794 }
5795 \@@_msg_new:nnn { Unknown-key-for-CodeAfter }
5796 {
5797   The~key~'\l_keys_key_str'~is~unknown. \\
5798   If~you~go~on,~it~will~be~ignored. \\
5799   For~a~list~of~the~available~keys~in~\token_to_str:N
5800   \CodeAfter,~type~H<return>.
5801 }
5802 {
5803   The~available~keys~are~(in~alphabetic~order):~

```



```

5804     delimiters/color,~
5805     rules~(with~the~subkeys~'color'~and~'width'),~
5806     sub-matrix~(several~subkeys)~
5807     and~xdots~(several~subkeys).~
5808     The~latter~is~for~the~command~\token_to_str:N \line.
5809 }
5810 \@@_msg_new:nnn { Unknown~key~for~SubMatrix }
5811 {
5812     The~key~'\l_keys_key_str'~is~unknown.\\
5813     If~you~go~on,~this~key~will~be~ignored. \\
5814     For~a~list~of~the~available~keys~in~\token_to_str:N
5815     \SubMatrix,~type~H~<return>.
5816 }
5817 {
5818     The~available~keys~are~(in~alphabetic~order):~
5819     'delimiters/color',~
5820     'extra-height',~
5821     'hlines',~
5822     'left-xshift',~
5823     'name',~
5824     'right-xshift',~
5825     'rules'~(with~the~subkeys~'color'~and~'width'),~
5826     'slim',~
5827     'vlines'~and~'xshift'~(which~set~both~'left-xshift'~
5828     and~'right-xshift').\\
5829 }
5830 \@@_msg_new:nnn { Unknown~key~for~notes }
5831 {
5832     The~key~'\l_keys_key_str'~is~unknown.\\
5833     If~you~go~on,~it~will~be~ignored. \\
5834     For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
5835 }
5836 {
5837     The~available~keys~are~(in~alphabetic~order):~
5838     bottomrule,~
5839     code-after,~
5840     code-before,~
5841     enumitem-keys,~
5842     enumitem-keys-para,~
5843     para,~
5844     label-in-list,~
5845     label-in-tabular~and~
5846     style.
5847 }
5848 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
5849 {
5850     The~key~'\l_keys_key_str'~is~unknown~for~the~command~
5851     \token_to_str:N \NiceMatrixOptions. \\
5852     If~you~go~on,~it~will~be~ignored. \\
5853     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5854 }
5855 {
5856     The~available~keys~are~(in~alphabetic~order):~
5857     allow-duplicate-names,~
5858     cell-space-bottom-limit,~
5859     cell-space-limits,~
5860     cell-space-top-limit,~
5861     code-for-first-col,~
5862     code-for-first-row,~
5863     code-for-last-col,~
5864     code-for-last-row,~
5865     create-extra-nodes,~
5866     create-medium-nodes,~

```

```

5867     create-large-nodes,~
5868     delimiters/color,~
5869     end-of-row,~
5870     first-col,~
5871     first-row,~
5872     hlines,~
5873     hvlines,~
5874     hvlines-except-corners,~
5875     last-col,~
5876     last-row,~
5877     left-margin,~
5878     letter-for-dotted-lines,~
5879     light-syntax,~
5880     notes~(several~subkeys),~
5881     nullify-dots,~
5882     renew-dots,~
5883     renew-matrix,~
5884     right-margin,~
5885     rules~(with~the~subkeys~'color'~and~'width'),~
5886     small,~
5887     sub-matrix~(several~subkeys),
5888     vlines,~
5889     xdots~(several~subkeys).
5890 }

5891 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
5892 {
5893     The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
5894     \{NiceArray\}. \\
5895     If~you~go~on,~it~will~be~ignored. \\
5896     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5897 }
5898 {
5899     The~available~keys~are~(in~alphabetic~order):~
5900     b,~
5901     baseline,~
5902     c,~
5903     cell-space-bottom-limit,~
5904     cell-space-limits,~
5905     cell-space-top-limit,~
5906     code-after,~
5907     code-for-first-col,~
5908     code-for-first-row,~
5909     code-for-last-col,~
5910     code-for-last-row,~
5911     colortbl-like,~
5912     columns-width,~
5913     create-extra-nodes,~
5914     create-medium-nodes,~
5915     create-large-nodes,~
5916     delimiters/color,~
5917     extra-left-margin,~
5918     extra-right-margin,~
5919     first-col,~
5920     first-row,~
5921     hlines,~
5922     hvlines,~
5923     hvlines-except-corners,~
5924     last-col,~
5925     last-row,~
5926     left-margin,~
5927     light-syntax,~
5928     name,~
5929     notes/bottomrule,~

```

```

5930 notes/para,~
5931 nullify-dots,~
5932 renew-dots,~
5933 right-margin,~
5934 rules~(with~the~subkeys~'color'~and~'width'),~
5935 small,~
5936 t,~
5937 vlines,~
5938 xdots/color,~
5939 xdots/shorten~and~
5940 xdots/line-style.
5941 }

```

This error message is used for the set of keys NiceMatrix/NiceMatrix and NiceMatrix/pNiceArray (but not by NiceMatrix/NiceArray because, for this set of keys, there is also the keys t, c and b).

```

5942 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
5943 {
5944   The~key~'\l_keys_key_str'~is~unknown~for~the~
5945   \@@_full_name_env:. \\\
5946   If~you~go~on,~it~will~be~ignored. \\\
5947   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5948 }
5949 {
5950   The~available~keys~are~(in~alphabetic~order):~
5951   b,~
5952   baseline,~
5953   c,~
5954   cell-space-bottom-limit,~
5955   cell-space-limits,~
5956   cell-space-top-limit,~
5957   code-after,~
5958   code-for-first-col,~
5959   code-for-first-row,~
5960   code-for-last-col,~
5961   code-for-last-row,~
5962   colortbl-like,~
5963   columns-width,~
5964   create-extra-nodes,~
5965   create-medium-nodes,~
5966   create-large-nodes,~
5967   delimiters/color,~
5968   extra-left-margin,~
5969   extra-right-margin,~
5970   first-col,~
5971   first-row,~
5972   hlines,~
5973   hvlines,~
5974   hvlines-except-corners,~
5975   l,~
5976   last-col,~
5977   last-row,~
5978   left-margin,~
5979   light-syntax,~
5980   name,~
5981   nullify-dots,~
5982   r,~
5983   renew-dots,~
5984   right-margin,~
5985   rules~(with~the~subkeys~'color'~and~'width'),~
5986   small,~
5987   t,~
5988   vlines,~
5989   xdots/color,~
5990   xdots/shorten~and~

```

```

5991     xdots/line-style.
5992 }
5993 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
5994 {
5995     The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
5996     \{NiceTabular\}. \\
5997     If~you~go~on,~it~will~be~ignored. \\
5998     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5999 }
6000 {
6001     The~available~keys~are~(in~alphabetic~order):~
6002     b,~
6003     baseline,~
6004     c,~
6005     cell-space-bottom-limit,~
6006     cell-space-limits,~
6007     cell-space-top-limit,~
6008     code-after,~
6009     code-for-first-col,~
6010     code-for-first-row,~
6011     code-for-last-col,~
6012     code-for-last-row,~
6013     colortbl-like,~
6014     columns-width,~
6015     create-extra-nodes,~
6016     create-medium-nodes,~
6017     create-large-nodes,~
6018     extra-left-margin,~
6019     extra-right-margin,~
6020     first-col,~
6021     first-row,~
6022     hlines,~
6023     hvlines,~
6024     hvlines-except-corners,~
6025     last-col,~
6026     last-row,~
6027     left-margin,~
6028     light-syntax,~
6029     name,~
6030     notes/bottomrule,~
6031     notes/para,~
6032     nullify-dots,~
6033     renew-dots,~
6034     right-margin,~
6035     rules~(with~the~subkeys~'color'~and~'width'),~
6036     t,~
6037     vlins,~
6038     xdots/color,~
6039     xdots/shorten~and~
6040     xdots/line-style.
6041 }
6042 \@@_msg_new:nnn { Duplicate~name }
6043 {
6044     The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
6045     the~same~environment~name~twice.~You~can~go~on,~but,~
6046     maybe,~you~will~have~incorrect~results~especially~
6047     if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
6048     message~again,~use~the~key~'allow-duplicate-names'~in~
6049     '\token_to_str:N \NiceMatrixOptions'.\\
6050     For~a~list~of~the~names~already~used,~type~H~<return>. \\
6051 }
6052 {
6053     The~names~already~defined~in~this~document~are:~

```

```

6054     \seq_use:Nnnn \g_@@_names_seq { ~and~ } { ,~ } { ~and~ }.
6055   }
6056 \@@_msg_new:nn { Option-auto-for-columns-width }
6057   {
6058     You-can't-give-the-value-'auto'~to-the-key-'columns-width'~here.~
6059     If-you-go-on,~the-key~will~be~ignored.
6060   }

```

## 18 History

The successive versions of the file `nicematrix.sty` provided by TeXLive are available on the SVN server of TeXLive:

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

### Changes between versions 1.0 and 1.1

The dotted lines are no longer drawn with Tikz nodes but with Tikz circles (for efficiency).  
Modification of the code which is now twice faster.

### Changes between versions 1.1 and 1.2

New environment `{NiceArray}` with column types L, C and R.

### Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.  
Correction of a bug in the definition of `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` (in fact, it was a typo).  
Options are now available locally in `{pNiceMatrix}` and its variants.  
The names of the options are changed. The old names were names in “camel style”.

### Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `{NiceArray}`, `{pNiceArrayC}` and its variants with the same meaning as in the package `array`.  
New option `columns-width` to fix the same width for all the columns of the array.

### Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of `nicematrix` were focused on the continuous dotted lines whereas the version 2.0 of `nicematrix` provides different features to improve the typesetting of mathematical matrices.

### Changes between version 2.0 and 2.1

New implementation of the environment `{pNiceArrayRC}`. With this new implementation, there is no restriction on the width of the columns.  
The package `nicematrix` no longer loads `mathtools` but only `amsmath`.  
Creation of “medium nodes” and “large nodes”.

## Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.

Following a discussion on TeX StackExchange<sup>54</sup>, Tikz externalization is now deactivated in the environments of the package `nicematrix`.<sup>55</sup>

## Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\left( \begin{array}{ccc} & C_j & \\ 0 & \vdots & 0 \\ & a & \cdots \\ 0 & & 0 \end{array} \right) L_i$$

## Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See [www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)

## Changes between version 2.1.4 and 2.1.5

Compatibility with the classes `revtex4-1` and `revtex4-2`.

Option `allow-duplicate-names`.

## Changes between version 2.1.5 and 2.2

Possibility to draw horizontal dotted lines to separate rows with the command `\hdottedline` (similar to the classical command `\hline` and the command `\hdashline` of `arydshln`).

Possibility to draw vertical dotted lines to separate columns with the specifier “:” in the preamble (similar to the classical specifier “|” and the specifier “:” of `arydshln`).

## Changes between version 2.2 and 2.2.1

Improvement of the vertical dotted lines drawn by the specifier “:” in the preamble.

Modification of the position of the dotted lines drawn by `\hdottedline`.

## Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.

Option `hlines`.

## Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of “|”) as `\hdotsfor` does.

Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

---

<sup>54</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](http://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>55</sup>Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

## Changes between version 3.0 and 3.1

Command `\Block` to draw block matrices.

Error message when the user gives an incorrect value for `last-row`.

A dotted line can no longer cross another dotted line (excepted the dotted lines drawn by `\cdottedline`, the symbol “:” (in the preamble of the array) and `\line` in `code-after`).

The starred versions of `\Cdots`, `\Ldots`, etc. are now deprecated because, with the new implementation, they become pointless. These starred versions are no longer documented.

The vertical rules in the matrices (drawn by “|”) are now compatible with the color fixed by `colortbl`.

Correction of a bug: it was not possible to use the colon “:” in the preamble of an array when `pdflatex` was used with `french-babel` (because `french-babel` activates the colon in the beginning of the document).

## Changes between version 3.1 and 3.2 (and 3.2a)

Option `small`.

## Changes between version 3.2 and 3.3

The options `first-row`, `last-row`, `first-col` and `last-col` are now available in the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.

The option `columns-width=auto` doesn't need any more a second compilation.

The options `renew-dots`, `renew-matrix` and `transparent` are now available as package options (as said in the documentation).

The previous version of `nicematrix` was incompatible with a recent version of `expl3` (released 2019/09/30). This version is compatible.

## Changes between version 3.3 and 3.4

Following a discussion on TeX StackExchange<sup>56</sup>, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

## Changes between version 3.4 and 3.5

Correction on a bug on the two previous versions where the `code-after` was not executed.

## Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate-name` in the environments of `amsmath`.

## Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

---

<sup>56</sup>cf. [tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize](https://tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize)

## Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`. An error is raised when an obsolete environment is used.

## Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.  
New options `create-medium-nodes` and `create-large-nodes`.

## Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).  
New option `dotted-lines-margin` for fine tuning of the dotted lines.

## Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

## Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.  
Options `vlines`, `hlines` and `hvlines`.  
Option `baseline` pour `{NiceArray}` (not for the other environments).  
The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell  $i-j$ , the name is  $i-j$ -block and, if the creation of the “medium nodes” is required, a node  $i-j$ -block-medium is created.  
If the user tries to use more columns than allowed by its environment, an error is raised by `nicematrix` (instead of a low-level error).  
The package must be loaded with the option `obsolete-environments` if we want to use the deprecated environments.

## Changes between versions 3.12 and 3.13

The behaviour of the command `\rotate` is improved when used in the “last row”.  
The option `dotted-lines-margin` has been renamed in `xdots/shorten` and the options `xdots/color` and `xdots/line-style` have been added for a complete customisation of the dotted lines.  
In the environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, etc.), it’s possible to use the options `l` (=L) or `r` (=R) to specify the type of the columns.  
The starred versions of the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots` and `\Iddots` are deprecated since the version 3.1 of `nicematrix`. Now, one should load `nicematrix` with the option `starred-commands` to avoid an error at the compilation.  
The code of `nicematrix` no longer uses Tikz but only PGF. By default, Tikz is *not* loaded by `nicematrix`.

## Changes between versions 3.13 and 3.14

Correction of a bug (question 60761504 on [stackoverflow](https://stackoverflow.com)).  
Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.



## Changes between versions 3.14 and 3.15

It's possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the `code-after` with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New keyword `\CodeAfter` (in the environments of `nicematrix`).

## Changes between versions 3.15 and 4.0

New environment `{NiceTabular}`

Commands to color cells, rows and columns with a perfect result in the PDF.

## Changes between versions 4.0 and 4.1

New keys `cell-space-top-limit` and `cell-space-bottom-limit`

New command `\diagbox`

The key `hylvline` don't draw rules in the blocks (commands `\Block`) and in the virtual blocks corresponding to the dotted lines.

## Changes between versions 4.1 and 4.2

It's now possible to write `\begin{pNiceMatrix}a&b\c&d\end{pNiceMatrix}`<sup>2</sup> with the expected result.

## Changes between versions 4.2 and 4.3

The horizontal centering of the content of a `\Block` is correct even when an instruction such as `!\qqquad` is used in the preamble of the array.

It's now possible to use the command `\Block` in the “last row”.

## Changes between versions 4.3 and 4.4

New key `hvlines-except-corners`.

## Changes between versions 4.4 and 5.0

Use of the standard column types `l`, `c` and `r` instead of `L`, `C` and `R`.

It's now possible to use the command `\diagbox` in a `\Block`.

Command `\tabularnote`

## Changes between versions 5.0 and 5.1

The vertical rules specified by `|` in the preamble are not broken by `\hline\hline` (and other).

Environment `{NiceTabular*}`

Command `\Vdotsfor` similar to `\Hdotsfor`

The variable `\g_nicematrix_code_after_tl` is now public.

## Changes between versions 5.1 and 5.2

The vertical rules specified by `|` or `||` in the preamble respect the blocks.

Key `respect-blocks` for `\rowcolors` (with a *s*) in the `code-before`.

The variable `\g_nicematrix_code_before_tl` is now public.

The key `baseline` may take in as value an expression of the form *line-i* to align the `\hline` in the row *i*.

The key `hvlines-except-corners` may take in as value a list of corners (eg: NW,SE).

## Changes between versions 5.2 and 5.3

Keys `c`, `r` and `l` for the command `\Block`.

It's possible to use the key `draw-first` with `\Ddots` and `\Iddots` to specify which dotted line will be drawn first (the other lines will be drawn parallel to that one if parallelization is activated).

## Changes between versions 5.3 and 5.4

Key `tabularnote`.

Different behaviour for the mono-column blocks.

## Changes between versions 5.4 and 5.5

The user must never put `\omit` before `\CodeAfter`.

Correction of a bug: the tabular notes `\tabularnotes` were not composed when present in a block (except a mono-column block).

## Changes between versions 5.5 and 5.6

Different behaviour for the mono-row blocks.

New command `\NotEmpty`.

## Changes between versions 5.6 and 5.7

New key `delimiters-color`

Keys `fill`, `draw` and `line-width` for the command `\Block`.

## Changes between versions 5.7 and 5.8

Keys `cols` and `restart` of the command `\rowcolors` in the `code-before`.

Modification of the behaviour of `\\` in the columns of type `p`, `m` or `b` (for a behaviour similar to the environments of `array`).

Better error messages for the command `\Block`.

## Changes between versions 5.8 and 5.9

Correction of a bug: in the previous versions, it was not possible to use the key `line-style` for the continuous dotted lines when the Tikz library `babel` was loaded.

New key `cell-space-limits`.

## Changes between versions 5.9 and 5.10

New command `\SubMatrix` available in the `\CodeAfter`.

It's possible to provide options (between brackets) to the command `\CodeAfter`.

A (non fatal) error is raised when the key `transparent`, which is deprecated, is used.

## Changes between versions 5.10 and 5.11

It's now possible, in the `code-before` and in the `\CodeAfter`, to use the syntax `|(i-|j)` for the Tikz node at the intersection of the (potential) horizontal rule number  $i$  and the (potential) vertical rule number  $j$ .

## Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<b>@@ commands:</b>	
<code>\@@_Block:</code> .....	1122, 4497
<code>\@@_Block_i</code> .....	4499, 4502
<code>\@@_Block_ii:nnnnn</code> .....	4502, 4503
<code>\@@_Block_iv:nnnnn</code> .....	4533, 4537
<code>\@@_Block_iv:nnnnnn</code> .....	4696, 4698
<code>\@@_Block_v:nnnnn</code> .....	4534, 4632
<code>\@@_Block_v:nnnnnn</code> .....	4725, 4728
<code>\@@_Cdots</code> .....	1047, 1112, 3113
<code>\g_@@_Cdots_lines_tl</code> .....	1139, 2430
<code>\@@_Cell:</code> .....	200, 809, 1541, 1588, 1610, 2203, 3213, 3214, 3215, 3216, 3217, 3218, 3219, 3220, 3221, 3222, 3223, 3224
<code>\@@_CodeAfter:</code> .....	1126, 5106
<code>\@@_CodeAfter_i:n</code> .....	811, 2081, 2126, 5106, 5107, 5117
<code>\@@_CodeAfter_ii:n</code> .....	5110, 5112
<code>\@@_CodeAfter_keys:</code> .....	2369, 2391
<code>\@@_Ddots</code> .....	1049, 1114, 3145
<code>\g_@@_Ddots_lines_tl</code> .....	1142, 2428
<code>\g_@@_HVdotsfor_lines_tl</code> .....	1144, 2426, 3272, 3348, 5543
<code>\@@_Hdotsfor:</code> .....	1052, 1119, 3248
<code>\@@_Hdotsfor:nnnn</code> .....	3274, 3286
<code>\@@_Hdotsfor_i</code> .....	3257, 3263, 3270
<code>\@@_Hline:</code> .....	1117, 4009
<code>\@@_Hline_i:n</code> .....	4009, 4010, 4016
<code>\@@_Hline_ii:nn</code> .....	4013, 4016
<code>\@@_Hline_iii:n</code> .....	4014, 4017
<code>\@@_Hspace:</code> .....	1118, 3199
<code>\@@_Iddots</code> .....	1050, 1115, 3169
<code>\g_@@_Iddots_lines_tl</code> .....	1143, 2429
<code>\@@_Ldots</code> .....	1046, 1051, 1111, 3097
<code>\g_@@_Ldots_lines_tl</code> .....	1140, 2431
<code>\l_@@_Matrix_bool</code> ....	246, 1411, 1427, 2194
<code>\l_@@_NiceArray_bool</code> .....	243, 352, 1187, 1301, 1356, 1458, 1470, 2170, 3886, 3887, 4005, 4006, 4208, 4215, 5027
<code>\g_@@_NiceMatrixBlock_int</code> .....	234, 4267, 4272, 4275, 4286
<code>\l_@@_NiceTabular_bool</code> .....	156, 244, 817, 982, 1162, 1243, 1247, 1389, 1393, 1459, 1471, 2223, 2232, 4565, 4641, 5035
<code>\@@_NotEmpty:</code> .....	1128, 2217
<code>\@@_OnlyMainNiceMatrix:n</code> .....	1124, 3747
<code>\@@_OnlyMainNiceMatrix_i:n</code> .....	3750, 3757, 3760
<code>\@@_SubMatrix</code> .....	2366, 5222
<code>\@@_Vdots</code> .....	1048, 1113, 3129
<code>\g_@@_Vdots_lines_tl</code> .....	1141, 2427
<code>\@@_Vdotsfor:</code> .....	1120, 3346
<code>\@@_Vdotsfor:nnnn</code> .....	3350, 3361
<code>\@@_W:</code> .....	1430, 1499
<code>\@@_actually_color:</code> .....	1246, 3486
<code>\@@_actually_diagbox:nnnnnn</code> .....	4544, 4761, 5047, 5063
<code>\@@_actually_draw_Cdots:</code> .....	2690, 2694
<code>\@@_actually_draw_Ddots:</code> .....	2816, 2820
<code>\@@_actually_draw_Iddots:</code> .....	2867, 2871
<code>\@@_actually_draw_Ldots:</code> ..	2648, 2652, 3337
<code>\@@_actually_draw_Vdots:</code> ..	2743, 2747, 3412
<code>\@@_adapt_S_column:</code> .....	170, 185, 1161
<code>\@@_add_to_colors_seq:nn</code> .....	3473, 3485, 3507, 3522, 3537
<code>\@@_adjust_pos_of_blocks_seq:</code> ..	2338, 2393
<code>\@@_adjust_pos_of_blocks_seq_i:nnnn</code> ..	2396, 2398
<code>\@@_adjust_size_box:</code> .....	888, 914, 1619, 2105, 2150
<code>\@@_after_array:</code> .....	1420, 2236
<code>\g_@@_after_col_zero_bool</code> .....	271, 1023, 2082, 3254
<code>\@@_analyze_end:Nn</code> .....	1879, 1924
<code>\l_@@_argspec_tl</code> .....	3095, 3096, 3097, 3113, 3129, 3145, 3169, 3268, 3269, 3270, 3344, 3345, 3346, 3422, 3423, 3424
<code>\@@_array:</code> .....	977, 1880, 1907
<code>\c_@@_arydshln_loaded_bool</code> ...	24, 31, 1527
<code>\l_@@_auto_columns_width_bool</code> .....	457, 591, 1991, 1995, 4262
<code>\l_@@_baseline_tl</code> .....	446, 447, 584, 585, 586, 587, 990, 1358, 1692, 1704, 1709, 1711, 1716, 1721, 1803, 1804, 1808, 1813, 1815, 1820
<code>\@@_begin_of_NiceMatrix:nn</code> ....	2192, 2213
<code>\@@_begin_of_row:</code> .....	814, 838, 2083
<code>\l_@@_block_auto_columns_width_bool</code> ..	1175, 1996, 4255, 4260, 4270, 4280
<code>\g_@@_block_box_int</code> .....	298, 1155, 4539, 4553, 4555, 4585, 4597, 4609, 4618, 4628
<code>\g_@@_blocks_dp_dim</code> .....	240, 896, 899, 900, 4612, 4615
<code>\g_@@_blocks_ht_dim</code> .....	239, 902, 905, 906, 4603, 4606
<code>\g_@@_blocks_seq</code> .....	285, 1177, 1742, 4622, 4634, 4696
<code>\g_@@_blocks_wd_dim</code> .....	238, 890, 893, 894, 4591, 4594
<code>\c_@@_booktabs_loaded_bool</code> ..	25, 34, 1062, 1774
<code>\@@_cartesian_path:</code> .....	3516, 3531, 3552, 3637, 3649, 3684

<code>\l_@@_cell_box</code>	816, 862, 864, 870, 876, 879, 883, 892, 893, 898, 899, 904, 905, 915, 916, 917, 918, 920, 923, 927, 929, 947, 1064, 1256, 1258, 1609, 1620, 2084, 2108, 2111, 2113, 2130, 2153, 2157, 4769, 4873, 4907, 5042	<code>\@@_cut_on_hyphen:w</code>	..... 3498, 3544, 3549, 3605, 3690, 3691, 3707, 3708, 4927, 4936, 5225, 5230
<code>\l_@@_cell_space_bottom_limit_dim</code>	..... 435, 505, 918	<code>\g_@@_ddots_int</code>	..... 2317, 2840, 2841
<code>\l_@@_cell_space_top_limit_dim</code>	434, 503, 916	<code>\@@_def_env:nnn</code>	..... 2176, 2187, 2188, 2189, 2190, 2191
<code>\l_@@_cell_type_tl</code>	..... 236, 237, 1541, 1611, 4517, 4519	<code>\@@_define_L_C_R:</code>	..... 222, 1313
<code>\@@_cellcolor</code>	..... 1237, 3554, 3566, 3567	<code>\c_@@_define_L_C_R_bool</code>	... 221, 1313, 5463
<code>\@@_cellcolor_tabular</code>	..... 1056, 3724	<code>\@@_define_com:nnn</code>	..... 5011, 5019, 5020, 5021, 5022, 5023
<code>\g_@@_cells_seq</code>	..... 1918, 1919, 1920, 1922	<code>\@@_delimiter:nnn</code>	..... 1639, 1647, 5120
<code>\@@_chessboardcolors</code>	..... 1242, 3559	<code>\l_@@_delimiters_color_tl</code>	..... 468, 704, 706, 760, 762, 779, 781, 1381, 1382, 5100, 5158, 5159, 5186, 5399, 5400, 5423, 5424
<code>\@@_cline</code>	..... 135, 1110	<code>\g_@@_delta_x_one_dim</code>	.... 2319, 2843, 2853
<code>\@@_cline_i:nn</code>	..... 136, 137, 149, 152	<code>\g_@@_delta_x_two_dim</code>	.... 2321, 2894, 2904
<code>\@@_cline_i:w</code>	..... 137, 138	<code>\g_@@_delta_y_one_dim</code>	.... 2320, 2845, 2853
<code>\l_@@_code_before_bool</code>	..... 275, 581, 608, 997, 1183, 1193, 1938, 1955, 1973, 2004, 2030, 2057, 2282, 2385	<code>\g_@@_delta_y_two_dim</code>	.... 2322, 2896, 2904
<code>\l_@@_code_before_tl</code>	.. 274, 580, 1184, 1245	<code>\@@_diagbox:nn</code>	..... 1127, 5043
<code>\l_@@_code_for_first_col_tl</code>	..... 523, 2095	<code>\@@_dotfill:</code>	..... 5032
<code>\l_@@_code_for_first_row_tl</code>	.. 527, 826, 4845	<code>\@@_dotfill_i:</code>	..... 5037, 5039
<code>\l_@@_code_for_last_col_tl</code>	..... 525, 2139	<code>\@@_dotfill_ii:</code>	..... 5036, 5039, 5040
<code>\l_@@_code_for_last_row_tl</code>	.. 529, 833, 4848	<code>\@@_dotfill_iii:</code>	..... 5040, 5041
<code>\g_@@_col_total_int</code>	..... 815, 1135, 1342, 2023, 2024, 2060, 2064, 2069, 2070, 2129, 2240, 2243, 2248, 2255, 2299, 2780, 3244, 3245, 3407, 4313, 4323, 4357, 4444, 4708, 4880, 5233, 5269	<code>\@@_double_int_eval:n</code>	.... 3418, 3432, 3433
<code>\l_@@_color_tl</code>	..... 293, 4494, 4557, 4559	<code>\g_@@_dp_ante_last_row_dim</code>	.... 841, 1095
<code>\l_@@_colors_seq</code>	1244, 3476, 3480, 3481, 3490	<code>\g_@@_dp_last_row_dim</code>	.... 841, 842, 1098, 1099, 1257, 1258, 1375
<code>\@@_colortbl_like:</code>	..... 1054, 1129	<code>\g_@@_dp_row_zero_dim</code>	.... 861, 862, 1089, 1090, 1368, 1797, 1836
<code>\l_@@_colortbl_like_bool</code>	432, 607, 1129, 1446	<code>\@@_draw_Cdots:nnn</code>	..... 2676
<code>\c_@@_colortbl_loaded_bool</code>	... 88, 92, 1079	<code>\@@_draw_Ddots:nnn</code>	..... 2808
<code>\l_@@_cols_tl</code>	..... 3515, 3530, 3551, 3576, 3584, 3585, 3686	<code>\@@_draw_Iddots:nnn</code>	..... 2859
<code>\g_@@_cols_vlism_seq</code>	.. 255, 1441, 1518, 5297	<code>\@@_draw_Ldots:nnn</code>	..... 2634
<code>\@@_columncolor</code>	..... 1241, 3518	<code>\@@_draw_Vdots:nnn</code>	..... 2728
<code>\@@_columncolor:n</code>	..... 3524, 3527	<code>\@@_draw_blocks:</code>	..... 1742, 4693
<code>\@@_columncolor_preamble</code>	..... 1058, 3738	<code>\@@_draw_dotted_lines:</code>	..... 2336, 2415
<code>\c_@@_columncolor_regex</code>	..... 209, 1449	<code>\@@_draw_dotted_lines_i:</code>	..... 2418, 2422
<code>\l_@@_columns_width_dim</code>	..... 235, 592, 726, 1992, 1998, 4268, 4274	<code>\l_@@_draw_first_bool</code>	.. 297, 3160, 3184, 3195
<code>\g_@@_com_or_env_str</code>	..... 259, 262	<code>\@@_draw_hlines:</code>	..... 2349, 4002
<code>\@@_computations_for_large_nodes:</code>	..... 4384, 4397, 4402	<code>\@@_draw_line:</code>	..... 2674, 2726, 2806, 2857, 2908, 2910, 3471, 4223, 4253
<code>\@@_computations_for_medium_nodes:</code>	..... 4304, 4373, 4383, 4394	<code>\@@_draw_line_ii:nn</code>	..... 3451, 3455
<code>\@@_compute_a_corner:nnnnnn</code>	..... 4080, 4082, 4084, 4086, 4091	<code>\@@_draw_line_iii:nn</code>	..... 3458, 3462
<code>\@@_compute_corners:</code>	..... 2337, 4072	<code>\@@_draw_non_standard_dotted_line:</code>	.. 2916, 2918
<code>\@@_construct_preamble:</code>	..... 1314, 1424	<code>\@@_draw_non_standard_dotted_line:n</code>	.. 2921, 2924
<code>\@@_create_col_nodes:</code>	.... 1883, 1911, 1930	<code>\@@_draw_non_standard_dotted_line:nnn</code>	..... 2926, 2931, 2945
<code>\@@_create_diag_nodes:</code>	... 1227, 2307, 2438	<code>\@@_draw_standard_dotted_line:</code>	.. 2915, 2946
<code>\@@_create_extra_nodes:</code>	..... 1741, 4294	<code>\@@_draw_standard_dotted_line_i:</code>	3009, 3013
<code>\@@_create_large_nodes:</code>	..... 4302, 4378	<code>\l_@@_draw_tl</code>	..... 292, 4676, 4681, 4734, 4915, 4921, 4923, 4925, 4956, 4957
<code>\@@_create_medium_and_large_nodes:</code>	.. 4299, 4389	<code>\@@_draw_vlines:</code>	..... 2350, 3883
<code>\@@_create_medium_nodes:</code>	..... 4300, 4368	<code>\g_@@_empty_cell_bool</code>	.... 282, 922, 931, 2118, 2165, 3111, 3127, 3143, 3167, 3190, 3201
<code>\@@_create_nodes:</code>	4375, 4386, 4396, 4399, 4440	<code>\l_@@_empty_corner_cells_seq</code>	.... 2343, 3821, 3827, 3834, 3942, 3948, 3955, 4074, 4147
<code>\@@_create_row_node:</code>	..... 993, 1026, 1063	<code>\@@_end_Cell:</code>	..... 202, 909, 1543, 1598, 1615, 2203, 3213, 3214, 3215, 3216, 3217, 3218, 3219, 3220, 3221, 3222, 3223, 3224
		<code>\l_@@_end_of_row_tl</code>	..... 465, 466, 517, 1903, 1904, 5638

\c_@@_endpgfortikzpicture_tl .....	1933, 1953, 2293, 3252, 3698, 3749, 4313,
..... 43, 47, 2419, 3459, 4194	4323, 4357, 4405, 4444, 4993, 4999, 5005, 5269
\c_@@_enumitem_loaded_bool .....	\l_@@_first_row_int .....
..... 26, 37, 325, 635, 640, 651, 656	... 304, 305, 520, 791, 1133, 1366, 1723,
\@@_env: .....	1794, 1822, 1833, 2291, 4306, 4320, 4347,
847, 853, 948, 954, 1002, 1008, 1014, 1207,	4404, 4442, 4788, 4806, 4991, 5132, 5246, 5656
1208, 1214, 1215, 1222, 1223, 1234, 1939,	\c_@@_footnote_bool .....
1942, 1944, 1960, 1966, 1969, 1978, 1984,	1150, 1422, 5440, 5476, 5499, 5502, 5512, 5518
1987, 2009, 2015, 2018, 2035, 2041, 2047,	\c_@@_footnotehyper_bool . 5439, 5477, 5509
2060, 2064, 2070, 2312, 2361, 2447, 2509,	\@@_full_name_env: 260, 5549, 5556, 5564,
2577, 2616, 2627, 3300, 3318, 3375, 3393,	5572, 5576, 5670, 5674, 5677, 5690, 5695,
3444, 3446, 3465, 3468, 4102, 4121, 4139,	5712, 5731, 5736, 5741, 5748, 5767, 5776, 5945
4326, 4328, 4336, 4447, 4456, 4474, 4784,	\@@_hdottedline: .....
4791, 4795, 4809, 4814, 4826, 4831, 4832,	1116, 4176
4835, 4852, 4886, 5135, 5138, 5249, 5251,	\@@_hdottedline:n .....
5256, 5258, 5272, 5274, 5279, 5281, 5374, 5385	4184, 4188
\g_@@_env_int .. 228, 229, 231, 1174, 1181,	\@@_hdottedline_i: .....
1185, 1195, 1199, 1202, 1211, 1212, 1219,	4179, 4181
1220, 1266, 1269, 1284, 1287, 2247, 2268,	\@@_hdottedline_i:n .....
2286, 2289, 2302, 2381, 3592, 3595, 3620, 4483	4193, 4197
\@@_error:n .....	\@@_hline:nn .....
12, 328, 353, 478,	3890, 4007, 4025
488, 537, 661, 709, 719, 725, 734, 742, 764,	\@@_hline_i:nn .....
771, 783, 789, 794, 805, 807, 1337, 1347,	2347, 3893, 3896
1414, 1726, 1779, 1825, 3579, 4691, 5104,	\@@_hline_i_complete:nn .....
5204, 5215, 5220, 5234, 5473, 5478, 5505, 5515	2347, 4000
\@@_error:nn .....	\@@_hline_ii:nnnn .. 3918, 3929, 3962, 4001
..... 13, 599, 1507, 1508, 1509, 3100,	\l_@@_hlines_bool .....
3103, 3116, 3119, 3132, 3135, 3149, 3150,	..... 452, 531, 543, 573, 1027, 2349
3155, 3156, 3173, 3174, 3179, 3180, 4088, 5209	\g_@@_ht_last_row_dim .....
\@@_error:nnn .....	..... 843, 1096, 1097, 1255, 1256, 1374
14, 3449, 5323, 5356	\g_@@_ht_row_one_dim .. 869, 870, 1093, 1094
\@@_error_too_much_cols: .....	\g_@@_ht_row_zero_dim .....
1480, 5528	..... 863, 864, 1091, 1092, 1369, 1796, 1835
\@@_everycr: .....	\@@_i: .....
1019, 1084, 1087	4306, 4308,
\@@_everycr_i: .....	4309, 4310, 4311, 4320, 4326, 4328, 4329,
1019, 1020	4330, 4331, 4336, 4337, 4338, 4339, 4347,
\l_@@_except_corners_clist .....	4350, 4352, 4353, 4354, 4406, 4408, 4411,
..... 453, 567, 571, 3787, 3909, 4075	4412, 4416, 4417, 4442, 4447, 4449, 4451,
\l_@@_exterior_arraycolsep_bool .....	4455, 4456, 4467, 4474, 4476, 4478, 4482, 4483
..... 448, 722, 1461, 1473	\g_@@_iddots_int .....
\l_@@_extra_left_margin_dim .....	2318, 2891, 2892
..... 463, 559, 1317, 2116	\l_@@_in_env_bool .... 242, 352, 1165, 1166
\l_@@_extra_right_margin_dim .....	\c_@@_in_preamble_bool . 21, 22, 23, 631, 647
..... 464, 560, 1329, 2161, 2783	\l_@@_initial_i_int .....
\@@_extract_coords_values: .... 4465, 4472	2324,
\@@_fatal:n 15, 251, 1165, 1635, 1645, 1888,	2454, 2529, 2532, 2557, 2565, 2569, 2578,
1892, 1894, 1927, 3259, 4679, 5533, 5536, 5539	2586, 2596, 2617, 2659, 2714, 2716, 2759,
\@@_fatal:nn .....	2824, 2875, 3290, 3291, 3301, 3369, 3379, 3381
16, 1532	\l_@@_initial_j_int .....
\l_@@_fill_tl ... 291, 4674, 4680, 4747, 4752	..... 2325, 2455, 2530, 2537,
\l_@@_final_i_int .....	2542, 2548, 2558, 2566, 2570, 2579, 2587,
..... 2326, 2456, 2461, 2464, 2489,	2597, 2618, 2656, 2698, 2773, 2775, 2780,
2497, 2501, 2510, 2518, 2598, 2628, 2668,	2826, 2877, 3294, 3304, 3306, 3365, 3366, 3377
2765, 2832, 2883, 3291, 3319, 3387, 3397, 3399	\l_@@_initial_open_bool .....
\l_@@_final_j_int .....	..... 2328, 2531, 2535, 2538, 2545, 2551,
2327, 2457, 2462, 2469, 2474, 2480, 2490,	2555, 2571, 2654, 2696, 2711, 2721, 2750,
2498, 2502, 2511, 2519, 2599, 2629, 2665,	2757, 2769, 2822, 2873, 3015, 3062, 3288,
2705, 2834, 2885, 3312, 3322, 3324, 3366, 3395	3295, 3307, 3363, 3370, 3382, 3440, 4190, 4227
\l_@@_final_open_bool .... 2329, 2463,	\@@_insert_tabularnotes: .....
2467, 2470, 2477, 2483, 2487, 2503, 2663,	..... 1746, 1749
2703, 2712, 2723, 2750, 2763, 2771, 2792,	\@@_instruction_of_type:nnn .....
2830, 2881, 3017, 3032, 3063, 3064, 3289,	..... 958, 3105, 3121, 3137, 3160, 3184
3313, 3325, 3364, 3388, 3400, 3441, 4191, 4228	\l_@@_inter_dots_dim .....
\@@_find_extremities_of_line:nnnn ...	. 436, 437, 2333, 3020, 3027, 3038, 3046,
..... 2451, 2638, 2680, 2732, 2812, 2863	3053, 3058, 3070, 3078, 4219, 4221, 4249, 4251
\l_@@_first_col_int .....	\g_@@_internal_code_after_tl .....
123,	..... 267, 1577, 1638, 1646,
136, 306, 307, 519, 787, 814, 1351, 1453,	1666, 2351, 2352, 4024, 4183, 4542, 4759, 5045
	\@@_intersect_our_row:nnnn .....
	3673
	\@@_intersect_our_row_p:nnnn .....
	3624
	\@@_j: .....
	4313, 4315,
	4316, 4317, 4318, 4323, 4326, 4328, 4331,

4333, 4334, 4336, 4339, 4341, 4342, 4357,	5646, 5653, 5660, 5667, 5673, 5675, 5681,
4360, 4362, 4363, 4364, 4419, 4421, 4424,	5698, 5703, 5710, 5717, 5724, 5729, 5739,
4426, 4430, 4431, 4444, 4447, 4448, 4450,	5745, 5752, 5758, 5764, 5772, 5774, 5780, 6056
4455, 4456, 4468, 4474, 4475, 4477, 4482, 4483	\@@_msg_new:nnn ... 18, 5441, 5687, 5785,
\l_@@_l_dim ..... 2993, 2994, 3007, 3008, 3020, 3026,	5795, 5810, 5830, 5848, 5891, 5942, 5993, 6042
3037, 3045, 3053, 3058, 3070, 3071, 3078, 3079	\@@_msg_redirect_name:nn ..... 19, 728, 1418, 4714, 4717
\l_@@_large_nodes_bool 460, 550, 4298, 4302	\@@_multicolumn:nnn ..... 1121, 3205
\g_@@_last_col_found_bool .. 314, 1138,	\g_@@_multicolumn_cells_seq ..... 1131, 3232, 4331, 4339, 4461, 4793, 4811
1343, 1406, 2022, 2051, 2127, 2239, 2296, 4878	\g_@@_multicolumn_sizes_seq 1132, 3234, 4462
\l_@@_last_col_int ..... 312, 313, 710, 753, 755, 772, 790,	\g_@@_name_env_str ..... 258,
806, 1205, 1280, 1286, 1293, 1346, 1465,	263, 264, 1159, 1160, 1926, 2171, 2172,
2199, 2201, 2240, 2243, 2295, 2737, 2778,	2180, 2181, 2210, 2221, 2229, 2387, 5015, 5530
3102, 3118, 3156, 3180, 4701, 4706, 4707,	\l_@@_name_str ..... 458, 601, 849, 852, 950, 953, 1010,
4708, 4711, 4742, 4754, 4765, 4780, 4809,	1013, 1264, 1273, 1276, 1282, 1291, 1294,
4814, 4822, 4837, 4995, 5001, 5007, 5532, 5550	1943, 1944, 1968, 1969, 1986, 1987, 2017,
\l_@@_last_col_without_value_bool ... 311, 752, 2241, 5535	2018, 2043, 2046, 2066, 2069, 2250, 2254,
\l_@@_last_empty_column_int ..... 4112, 4113, 4126, 4132, 4145	2271, 2275, 2308, 2312, 4452, 4455, 4479, 4482
\l_@@_last_empty_row_int ..... 4094, 4095, 4108, 4129	\g_@@_names_seq ..... 241, 598, 600, 6054
\l_@@_last_row_int ..... 308, 309, 521, 831, 877, 1031,	\l_@@_nb_cols_int ..... 4982, 4987, 4990, 4994, 5000, 5006
1203, 1251, 1261, 1268, 1275, 1331, 1335,	\l_@@_nb_rows_int ..... 4981, 4986, 4997
1338, 1350, 1372, 1905, 1906, 2091, 2092,	\@@_newcolumnntype ..... 1037, 1429, 1430
2136, 2137, 2262, 2643, 2685, 3134, 3150,	\@@_node_for_multicolumn:nn .... 4463, 4470
3174, 3755, 3763, 4700, 4703, 4704, 4723,	\@@_node_for_the_cell: 928, 934, 2112, 2162
4742, 4754, 4764, 4778, 4836, 4847, 5003, 5766	\@@_node_left:nn ..... 5373, 5374, 5391
\l_@@_last_row_without_value_bool ... 310, 1263, 1333, 2260	\@@_node_position: .. 1214, 1216, 1222, 1224
\l_@@_left_delim_dim ..... 1299, 1303, 1308, 1871, 2114	\@@_node_right:nn ..... 5383, 5385, 5415
\l_@@_left_delim_tl .. 1152, 1307, 1383, 4218	\g_@@_not_empty_cell_bool 273, 926, 932, 2218
\l_@@_left_margin_dim ..... 461, 553, 1316, 2115, 4209, 4435	\@@_not_in_exterior:nnnn ..... 3665
\l_@@_letter_for_dotted_lines_str ... 733, 744, 745, 1513	\@@_not_in_exterior_p:nnnn ..... 3597
\l_@@_letter_vlism_tl ..... 254, 536, 1516	\l_@@_notes_above_space_dim ..... 454, 455
\l_@@_light_syntax_bool ..... 445, 515, 1319, 1324, 2263	\l_@@_notes_bottomrule_bool ..... 619, 775, 800, 1772
\@@_light_syntax_i ..... 1896, 1899	\l_@@_notes_code_after_tl ..... 617, 1781
\@@_line ..... 2365, 3424	\l_@@_notes_code_before_tl ..... 615, 1753
\@@_line_i:nn ..... 3431, 3438	\@@_notes_label_in_list:n 321, 340, 348, 627
\l_@@_line_width_dim ..... 294, 4683, 4735, 4916, 4948, 4959	\@@_notes_label_in_tabular:n . 320, 361, 624
\@@_line_with_light_syntax:n ... 1910, 1914	\l_@@_notes_para_bool .. 613, 773, 798, 1757
\@@_line_with_light_syntax_i:n ..... 1909, 1915, 1916	\@@_notes_style:n ..... 319, 322, 340, 348, 364, 369, 621
\@@_math_toggle_token: ..... 155, 911, 2085, 2102, 2131, 2147, 5087, 5091	\l_@@_nullify_dots_bool ..... 456, 548, 3109, 3125, 3141, 3165, 3188
\g_@@_max_cell_width_dim ..... 919, 920, 1176, 1997, 4261, 4287	\l_@@_number_of_notes_int 318, 355, 365, 375
\l_@@_max_delimiter_width_bool ..... 469, 514, 1402	\@@_old_CT@arc@ ..... 1167, 2389
\c_@@_max_l_dim ..... 3007, 3012	\@@_old_cdots ..... 1104, 3126
\l_@@_medium_nodes_bool 459, 549, 4296, 4828	\@@_old_ddots ..... 1106, 3166
\@@_message_hdotsfor: 5541, 5549, 5556, 5564	\@@_old_dotfill ..... 5031, 5034, 5042
\@@_msg_new:nn ..... 17, 5455, 5481, 5490, 5546, 5553, 5561,	\@@_old_dotfill: ..... 1125
5569, 5574, 5580, 5585, 5590, 5595, 5600,	\l_@@_old_iRow_int ..... 268, 1066, 2435
5605, 5611, 5617, 5624, 5629, 5635, 5641,	\@@_old_ialign: ..... 992, 1100, 4695
	\@@_old_iddots ..... 1107, 3189
	\l_@@_old_jCol_int ..... 269, 1069, 2436
	\@@_old_ldots ..... 1103, 3110
	\@@_old_multicolumn ..... 3204, 3208
	\@@_old_pgful@check@rerun ..... 81, 85
	\@@_old_vdots ..... 1105, 3142
	\l_@@_parallelize_diags_bool ..... 449, 450, 545, 2315, 2838, 2889
	\@@_patch_preamble:n ..... 1443, 1484,
	1522, 1530, 1551, 1580, 1640, 1660, 1668, 1680



\@@_patch_preamble_i:n	1488, 1489, 1490, 1537	\@@_rectanglecolor:nn	3539, 3542
\@@_patch_preamble_ii:nn	1491, 1492, 1493, 1548	\@@_renew_NC@rewrite@S:	191, 193, 1137
\@@_patch_preamble_iii:n	1494, 1553, 1561	\@@_renew_dots:	1044, 1130
\@@_patch_preamble_iii_i:n	1556, 1558	\l_@@_renew_dots_bool	546, 718, 1130, 5465, 5472
\@@_patch_preamble_iv:nnn	1495, 1496, 1497, 1583	\@@_renew_matrix:	713, 717, 4961, 5467, 5471
\@@_patch_preamble_ix:n	1673, 1683	\l_@@_respect_blocks_bool	3574, 3589, 3617
\@@_patch_preamble_v:nnnn	1498, 1499, 1604	\@@_restore_iRow_jCol:	2388, 2433
\@@_patch_preamble_vi:n	1500, 1626	\@@_revtex_array:	969, 980
\@@_patch_preamble_vii:n	1501, 1502, 1503, 1632	\c_@@_revtex_bool	50, 52, 55, 57, 979
\@@_patch_preamble_viii:n	1504, 1505, 1506, 1642	\l_@@_right_delim_dim	1300, 1304, 1310, 1874, 2159
\@@_patch_preamble_viii_i:n	1648, 1650	\l_@@_right_delim_tl	1153, 1309, 1399, 4220
\@@_patch_preamble_x:n	1546, 1602, 1624, 1630, 1670, 1686	\l_@@_right_margin_dim	462, 555, 1328, 2160, 2782, 4216, 4438
\@@_patch_preamble_xi:n	1514, 1662	\@@_rotate:	1123, 3417
\@@_pgf_rect_node:nnn	408, 4830	\g_@@_rotate_bool	247, 886, 913, 1618, 2104, 2149, 3417, 4564, 4582, 4587, 4647, 4660, 4770
\@@_pgf_rect_node:nnnnn	383, 4446, 4473, 4783, 4825, 5360	\@@_rotate_cell_box:	874, 913, 1618, 2104, 2149, 4770
\c_@@_pgfortikzpicture_tl	42, 46, 2417, 3457, 4192	\g_@@_row_of_col_done_bool	272, 1024, 1158, 1952
\@@_picture_position:	1208, 1216, 1224	\g_@@_row_total_int	1134, 1349, 1724, 1823, 2262, 2269, 2276, 2292, 3332, 4306, 4320, 4347, 4442, 4723, 4788, 4806, 5132, 5232, 5246, 5657
\l_@@_pos_of_block_tl	295, 296, 4488, 4490, 4492, 4518, 4519, 4521, 4564, 4568, 4575, 4625, 4648, 4650, 4661, 4664, 4685, 4687, 4689, 4854, 4866, 4877, 4881, 4888, 4900	\@@_rowcolor	1239, 3503
\g_@@_pos_of_blocks_seq	286, 1178, 2303, 2341, 2395, 3235, 3781, 3903, 4159, 4730, 5055	\@@_rowcolor:n	3509, 3512
\g_@@_pos_of_stroken_blocks_seq	288, 1179, 3785, 3907, 4744	\@@_rowcolor_tabular	1057, 3729
\g_@@_pos_of_xdots_seq	287, 1180, 2342, 2594, 3783, 3905	\@@_rowcolors	1240, 3581
\@@_pre_array:	1060, 1298	\@@_rowcolors_i:nnnn	3625, 3660
\c_@@_preamble_first_col_tl	1454, 2077	\l_@@_rowcolors_restart_bool	3577, 3608
\c_@@_preamble_last_col_tl	1466, 2122	\g_@@_rows_seq	1902, 1904, 1906, 1908, 1910
\g_@@_preamble_tl	1154, 1431, 1437, 1440, 1451, 1454, 1463, 1466, 1475, 1479, 1520, 1529, 1539, 1550, 1563, 1585, 1606, 1628, 1637, 1659, 1664, 1677, 1685, 1880, 1907	\l_@@_rows_tl	3514, 3529, 3550, 3627, 3703
\@@_pred:n	124, 154, 2201, 3822, 3835, 3943, 3956	\l_@@_rules_color_tl	270, 492, 1191, 1192, 5294, 5295
\@@_provide_pgfsyspdfmark:	210, 219, 1149	\@@_set_CT@arc@:	157, 1192, 5295
\@@_put_box_in_flow:	1404, 1688, 1873	\@@_set_CT@arc@_i:	158, 159
\@@_put_box_in_flow_bis:nn	1403, 1840	\@@_set_CT@arc@_ii:	158, 161
\@@_put_box_in_flow_i:	1694, 1696	\@@_set_final_coords:	2607, 2632
\@@_qpoint:n	232, 1699, 1701, 1713, 1729, 1788, 1790, 1806, 1817, 1828, 2444, 2446, 2656, 2659, 2665, 2668, 2698, 2705, 2714, 2716, 2759, 2765, 2773, 2775, 2824, 2826, 2832, 2834, 2875, 2877, 2883, 2885, 3465, 3468, 3697, 3701, 3714, 3716, 3845, 3847, 3849, 3966, 3968, 3970, 4200, 4204, 4211, 4245, 4248, 4250, 4352, 4362, 4774, 4776, 4778, 4780, 4802, 4822, 4850, 4932, 4934, 4941, 4943, 5068, 5070, 5073, 5075, 5125, 5127, 5263, 5266, 5303, 5318, 5333	\@@_set_final_coords_from_anchor:n	2623, 2671, 2709, 2753, 2768, 2837, 2888
\l_@@_radius_dim	440, 441, 1665, 2332, 2672, 2673, 3087, 4178, 4202, 4246, 4247	\@@_set_initial_coords:	2602, 2621
\l_@@_real_left_delim_dim	1842, 1857, 1872	\@@_set_initial_coords_from_anchor:n	2612, 2662, 2702, 2752, 2762, 2829, 2880
\l_@@_real_right_delim_dim	1843, 1869, 1875	\@@_set_size:n	4979, 4988
\@@_rectanglecolor	1238, 3533, 3557	\c_@@_siunitx_loaded_bool	163, 167, 172, 190
		\l_@@_small_bool	711, 758, 768, 792, 820, 1072, 1634, 1644, 2086, 2132, 2330
		\@@_standard_cline	120, 1109
		\@@_standard_cline:w	120, 121
		\l_@@_standard_cline_bool	433, 501, 1108
		\c_@@_standard_tl	443, 444, 2914, 4222, 4252
		\g_@@_static_num_of_col_int	290, 1413, 1444, 4711, 5565, 5577, 5732
		\l_@@_stop_loop_bool	2458, 2459, 2491, 2504, 2513, 2526, 2527, 2559, 2572, 2581
		\@@_stroke_block:nnn	4739, 4912
		\l_@@_submatrix_extra_height_dim	299, 5178, 5289
		\l_@@_submatrix_hlines_clist	302, 5198, 5325, 5327

<code>\l_@@_submatrix_left_xshift_dim</code> .....	300, 5180, 5336, 5369
<code>\l_@@_submatrix_name_str</code> .....	5211, 5236, 5358, 5360, 5372, 5374, 5382, 5385
<code>\g_@@_submatrix_names_seq</code> .....	289, 2368, 5208, 5212, 5696
<code>\l_@@_submatrix_right_xshift_dim</code> .....	301, 5182, 5345, 5379
<code>\l_@@_submatrix_slim_bool</code> .....	5188, 5244
<code>\l_@@_submatrix_vlines_clist</code> .....	303, 5200, 5310, 5312
<code>\@@_succ:n</code> .....	149, 153, 1002, 1008, 1578, 1639, 1701, 2035, 2041, 2046, 2047, 2060, 2064, 2069, 2070, 2297, 2665, 2705, 2716, 2765, 2775, 2832, 2834, 2877, 2883, 3669, 3701, 3714, 3818, 3849, 3887, 3939, 3970, 4006, 4025, 4164, 4166, 4168, 4170, 4211, 4250, 4412, 4416, 4426, 4430, 4778, 4780, 4822, 4941, 4943, 5073, 5075, 5127
<code>\l_@@_suffix_tl</code> .....	4374, 4385, 4395, 4398, 4447, 4455, 4456, 4474, 4482, 4483
<code>\c_@@_table_collect_begin_tl</code> ..	180, 182, 200
<code>\c_@@_table_print_tl</code> .....	183, 184, 202
<code>\l_@@_tabular_width_dim</code> .....	245, 985, 987, 1477, 2230
<code>\l_@@_tabularnote_tl</code> .....	317, 777, 802, 1745, 1754
<code>\g_@@_tabularnotes_seq</code> .....	316, 356, 1760, 1766, 1782
<code>\@@_test_hline_in_block:nnnn</code> .....	3904, 3906, 4028
<code>\@@_test_hline_in_stroken_block:nnnn</code> ..	3908, 4050
<code>\@@_test_if_cell_in_a_block:nn</code> .....	4098, 4116, 4134, 4154
<code>\@@_test_if_cell_in_block:nnnnnnn</code> ..	4160, 4162
<code>\@@_test_if_math_mode:</code> .....	248, 1164, 2182
<code>\@@_test_in_corner_h:</code> .....	3909, 3937
<code>\@@_test_in_corner_v:</code> .....	3788, 3816
<code>\@@_test_vline_in_block:nnnn</code> .....	3782, 3784, 4039
<code>\@@_test_vline_in_stroken_block:nnnn</code> ..	3786, 4061
<code>\l_@@_the_array_box</code> ..	1312, 1315, 1739, 1740
<code>\c_@@_tikz_loaded_bool</code> .....	27, 41, 1229, 2353, 4229
<code>\@@_true_c:</code> .....	201, 1500
<code>\l_@@_type_of_col_tl</code> ..	756, 757, 2211, 2213
<code>\c_@@_types_of_matrix_seq</code> .....	5520, 5521, 5526, 5530
<code>\@@_update_for_first_and_last_row:</code> ..	857, 921, 1253, 2106, 2151
<code>\@@_use_arraybox_with_notes:</code> ...	1363, 1801
<code>\@@_use_arraybox_with_notes_b:</code> ..	1360, 1785
<code>\@@_use_arraybox_with_notes_c:</code> .....	1361, 1392, 1737, 1799, 1838
<code>\@@_vdottedline:n</code> .....	1667, 4225
<code>\@@_vdottedline_i:n</code> .....	4232, 4237, 4241
<code>\@@_vline:nn</code> .....	1578, 3765, 3888
<code>\@@_vline_i:nn</code> .....	2346, 3770, 3774
<code>\@@_vline_i_complete:nn</code> .....	2346, 3881
<code>\@@_vline_ii:nnnn</code> ...	3797, 3808, 3841, 3882
<code>\l_@@_vlines_bool</code> .....	451, 532, 542, 572, 1435, 1460, 1472, 1675, 2350
<code>\@@_w:</code> .....	1429, 1498
<code>\g_@@_width_first_col_dim</code> .....	284, 1157, 1354, 1947, 2107, 2108
<code>\g_@@_width_last_col_dim</code> .....	283, 1156, 1408, 2056, 2152, 2153
<code>\l_@@_x_final_dim</code> .....	278, 2609, 2666, 2667, 2706, 2707, 2755, 2777, 2785, 2789, 2793, 2795, 2800, 2802, 2835, 2844, 2852, 2886, 2895, 2903, 2942, 2957, 2966, 3000, 3052, 3068, 3469, 4212, 4221, 4247, 5243, 5259, 5260, 5345, 5362, 5379
<code>\l_@@_x_initial_dim</code> .....	276, 2604, 2657, 2658, 2699, 2700, 2755, 2776, 2777, 2784, 2789, 2793, 2795, 2797, 2800, 2802, 2827, 2844, 2852, 2878, 2895, 2903, 2939, 2956, 2966, 3000, 3052, 3066, 3068, 3086, 3088, 3466, 4205, 4219, 4246, 5242, 5252, 5253, 5336, 5361, 5369
<code>\l_@@_xdots_color_tl</code> .....	467, 481, 2647, 2689, 2741, 2742, 2815, 2866, 2922, 3336, 3411, 3428
<code>\l_@@_xdots_down_tl</code> ...	485, 2929, 2950, 2985
<code>\l_@@_xdots_line_style_tl</code> .....	442, 444, 477, 2914, 2922, 4222, 4252
<code>\l_@@_xdots_shorten_dim</code> .....	438, 439, 483, 2334, 2936, 2937, 3026, 3037, 3045
<code>\l_@@_xdots_up_tl</code> ...	486, 2928, 2949, 2975
<code>\l_@@_y_final_dim</code> .....	279, 2610, 2669, 2673, 2718, 2722, 2724, 2766, 2833, 2846, 2849, 2884, 2897, 2900, 2942, 2957, 2965, 3002, 3057, 3076, 3470, 4203, 4251, 5128, 5150, 5165, 5267, 5282, 5283, 5288, 5305, 5320, 5362, 5370, 5380
<code>\l_@@_y_initial_dim</code> .....	277, 2605, 2660, 2672, 2717, 2718, 2722, 2724, 2760, 2825, 2846, 2851, 2876, 2897, 2902, 2939, 2956, 2965, 3002, 3057, 3074, 3076, 3086, 3089, 3467, 4201, 4202, 4203, 4249, 5126, 5150, 5165, 5264, 5275, 5276, 5288, 5304, 5319, 5361, 5370, 5380
<code>\</code> .....	1893, 1915, 4995, 5001, 5007, 5443, 5444, 5487, 5496, 5577, 5582, 5587, 5592, 5597, 5620, 5632, 5637, 5643, 5650, 5657, 5663, 5664, 5678, 5684, 5690, 5691, 5707, 5713, 5720, 5727, 5736, 5742, 5749, 5755, 5761, 5773, 5777, 5782, 5788, 5797, 5798, 5812, 5813, 5828, 5832, 5833, 5851, 5852, 5894, 5895, 5945, 5946, 5996, 5997, 6049, 6050
<code>\{</code> .....	264, 1503, 1656, 2189, 5023, 5341, 5683, 5749, 5894, 5996
<code>\}</code> .....	264, 1506, 2189, 5023, 5350, 5683, 5749, 5894, 5996
<code>\ </code> .....	2191, 5022
<code>\_</code> .....	5544, 5549, 5556, 5564, 5565, 5576, 5577, 5619, 5631, 5656, 5657, 5670, 5674, 5677, 5689, 5695, 5700, 5731, 5732, 5733, 5741, 5747, 5760, 5767, 5768, 5776
<b>A</b>	
<code>\A</code> .....	5206
<code>\aboverulesep</code> .....	1776



<code>\addtocounter</code> .....	373
<code>\alph</code> .....	319
<code>\arraybackslash</code> .....	1591
<code>\arraycolsep</code> .....	554, 556, 558, 984, 1075, 1303, 1304, 1391, 1395, 4208, 4215
<code>\arrayrulecolor</code> .....	95
<code>\arrayrulewidth</code> .....	128, 133, 145, 494, 848, 1001, 1003, 1009, 1032, 1438, 1521, 1571, 1678, 1793, 1832, 1959, 1961, 1967, 1977, 1979, 1985, 2008, 2010, 2016, 2034, 2036, 2042, 3699, 3700, 3702, 3715, 3717, 3857, 3858, 3860, 3871, 3877, 3979, 3990, 3996, 4021, 4287, 4916, 5150, 5289, 5293
<code>\arraystretch</code> ..	1074, 4561, 4644, 4657, 5265, 5268
<code>\AtBeginDocument</code> .....	23, 28, 73, 89, 164, 188, 323, 437, 439, 441, 455, 633, 649, 2413, 3093, 3266, 3342, 3420, 3453, 4186
<code>\AutoNiceMatrix</code> .....	5024
<code>\AutoNiceMatrixWithDelims</code> ...	4984, 5016, 5028
<b>B</b>	
<code>\baselineskip</code> .....	98, 105
<code>\bgroup</code> .....	1151
<code>\Block</code> .....	1122, 5619, 5788
<code>\BNiceMatrix</code> .....	4976
<code>\bNiceMatrix</code> .....	4973
bool commands:	
<code>\bool_do_until:Nn</code> .....	2459, 2527
<code>\bool_gset_false:N</code> .....	886, 931, 932, 1023, 1138, 1158, 2118, 2165, 4037, 4048, 4059, 4070, 4587
<code>\bool_gset_true:N</code> .....	1952, 2082, 2127, 2218, 3111, 3127, 3143, 3167, 3190, 3201, 3417, 3780, 3902
<code>\bool_if:NTF</code> .....	156, 172, 635, 640, 651, 656, 817, 820, 913, 997, 1024, 1027, 1062, 1072, 1129, 1130, 1150, 1165, 1175, 1193, 1229, 1243, 1247, 1313, 1333, 1406, 1411, 1422, 1427, 1446, 1618, 1634, 1644, 1675, 1772, 1938, 1955, 1973, 1991, 2004, 2030, 2051, 2057, 2086, 2104, 2132, 2149, 2239, 2241, 2260, 2263, 2282, 2315, 2330, 2349, 2350, 2353, 2721, 2723, 2838, 2889, 3109, 3125, 3141, 3165, 3188, 4107, 4125, 4143, 4270, 4280, 4302, 4564, 4582, 4647, 4660, 4770, 4828, 5035, 5499, 5509, 5535
<code>\bool_if:nTF</code> .....	190, 325, 352, 960, 1343, 3442, 3675, 4296, 4878, 5129, 5139, 5141, 5154, 5160, 5169
<code>\bool_lazy_all:nTF</code> .....	1456, 1468, 2339, 4030, 4041, 4052, 4063
<code>\bool_lazy_and:nnTF</code> .....	1525, 1994, 2087, 2294, 2710, 2948, 3250, 3588, 3616, 3851, 3972, 4928, 5314, 5329
<code>\bool_lazy_any:nTF</code> .....	1652
<code>\bool_lazy_or:nnTF</code> .....	474, 925, 1722, 1743, 1821, 2135, 2750, 3006, 3667, 4099, 4117, 4135, 4505, 4510, 4530, 4733, 5231
<code>\bool_lazy_or_p:nn</code> .....	2090
<code>\bool_not_p:n</code> .....	1459, 1460, 1461, 1471, 1472, 1473, 1996, 2296
<code>\bool_set:Nn</code> .....	2754, 3610
<code>\c_false_bool</code> .....	1647, 3105, 3121, 3137

<code>\g_tmpa_bool</code> .....	3780, 3789, 3823, 3831, 3836, 3902, 3910, 3944, 3952, 3957, 4037, 4048, 4059, 4070
<code>\l_tmpb_bool</code> ...	4104, 4118, 4136, 4158, 4171
<code>\c_true_bool</code> .....	1639
box commands:	
<code>\box_clear_new:N</code> .....	1064, 1312
<code>\box_dp:N</code> .....	842, 862, 899, 918, 1090, 1099, 1258, 1596, 1691, 1851, 1864, 4617, 5268
<code>\box_gclear_new:N</code> .....	4552
<code>\box_grotate:Nn</code> .....	4584
<code>\box_ht:N</code> .....	843, 864, 870, 882, 905, 916, 1092, 1094, 1097, 1256, 1592, 1690, 1851, 1864, 4608, 5265
<code>\box_move_up:nn</code> ..	64, 66, 68, 1734, 1799, 1838
<code>\box_rotate:Nn</code> .....	876
<code>\box_set_dp:Nn</code> .....	898, 917, 1691
<code>\box_set_ht:Nn</code> .....	904, 915, 1690
<code>\box_set_wd:Nn</code> .....	892
<code>\box_use:N</code> .....	376, 883
<code>\box_use_drop:N</code> ...	923, 929, 947, 1620, 1693, 1734, 1735, 1740, 2113, 4627, 4873, 4907
<code>\box_wd:N</code> .....	377, 893, 920, 927, 1308, 1310, 1739, 1858, 1870, 2108, 2111, 2153, 2157, 4596, 5042
<code>\l_tmpa_box</code> .....	359, 376, 377, 1307, 1308, 1309, 1310, 1378, 1690, 1691, 1693, 1734, 1735, 1851, 1864
<code>\l_tmpb_box</code> .....	1844, 1858, 1859, 1870

## C

<code>\c</code> .....	209, 1450
<code>\Cdots</code> .....	1112, 3116, 3119
<code>\cdots</code> .....	1047, 1104
<code>\cellcolor</code> .....	1056, 1237, 3727
<code>\chessboardcolors</code> .....	1242
<code>\cline</code> .....	148, 1109, 1110
clist commands:	
<code>\clist_if_empty:NTF</code> .....	3787, 3909
<code>\clist_map_inline:Nn</code> .....	3686, 3703, 4075, 5312, 5327
<code>\clist_map_inline:nn</code> .....	2206, 3556, 3601
<code>\clist_new:N</code> .....	302, 303, 453
<code>\clist_set:Nn</code> .....	571
<code>\CodeAfter</code> .....	811, 1126, 1896, 1899, 2081, 2126, 2367, 5800
<code>\color</code> .....	99, 106, 160, 162, 1382, 2641, 2644, 2647, 2683, 2686, 2689, 2735, 2738, 2742, 2815, 2866, 3330, 3333, 3336, 3405, 3408, 3411, 3428, 3492, 3634, 3635, 3646, 3647, 4559, 4681, 5159, 5400, 5424
<code>\colorlet</code> .....	256, 257, 827, 834, 2096, 2140
<code>\columncolor</code> .....	1058, 1241, 2376, 3743
<code>\cr</code> .....	132, 150, 2075
<code>\crrcr</code> .....	1932
cs commands:	
<code>\cs_generate_variant:Nn</code> .....	58, 152, 2945, 3485, 4292, 4293
<code>\cs_gset:Npn</code> .....	99, 106, 2247, 2254, 2268, 2275, 4285
<code>\cs_gset_eq:NN</code> ...	185, 219, 1083, 1167, 2389
<code>\cs_if_exist:NTF</code> .....	57, 1065, 1068, 1168, 1171, 1266, 1273, 1284, 1291, 2435, 2436, 2494, 2507,

2562, 2575, 3298, 3316, 3373, 3391, 4272, 4325, 4790, 4808, 5134, 5248, 5255, 5271, 5278	\endNiceArray ..... 2226, 2235
\cs_if_exist_p:N ..... 475, 3591, 3619, 4101, 4120, 4138	\endNiceArrayWithDelims ..... 2175, 2185
\cs_if_free:Ntf ..... 215, 2636, 2678, 2730, 2810, 2861	\endpgfscope ..... 2990, 5088
\cs_if_free_p:N ..... 3444, 3446	\endpNiceMatrix ..... 4965
\cs_new_protected:Npx ..... 2415, 3455, 4188	\endsavenotes ..... 1422
\cs_set:Nn ..... 621, 624, 627	\endtabularnotes ..... 1767
\cs_set:Npn ..... 95, 96, 102, 103, 108, 120, 121, 135, 137, 138, 160, 162, 322, 1039, 2453, 2515, 2583, 3340, 3415, 4009, 4010, 4016, 4017, 4561, 4644, 4657	\endVNiceMatrix ..... 4971
\cs_set_nopar:Npn ..... 974, 986, 1074, 1077, 4467, 4468	\endvNiceMatrix ..... 4968
\cs_set_nopar:Npx ..... 987	\enskip ..... 1637, 1659
\cs_set_protected:Npn ..... 5013	\ensuremath ..... 3110, 3126, 3142, 3166, 3189
\cs_set_protected_nopar:Npn ..... 4540, 4757	\everycr ..... 131, 150, 1087
<b>D</b>	exp commands:
\Ddots ..... 1114, 3149, 3150, 3155, 3156	\exp_after:wN 197, 1192, 1383, 1399, 1443, 5295
\ddots ..... 1049, 1106	\exp_args:Ne ..... 3211
\diagbox ..... 1127, 4540, 4757	\exp_args:Nnc ..... 4274
dim commands:	\exp_args:Nne ..... 3207
\dim_add:Nn ..... 4436	\exp_args:Nne ..... 2213
\dim_compare:nNnTF ..... 98, 105, 890, 896, 902, 1477, 1992, 2157, 2795, 4349, 4359, 4800, 4820, 5042	\exp_args:NNV ..... 1904, 3097, 3113, 3129, 3145, 3169, 3270, 3346, 3424
\dim_gzero:N ..... 894, 900, 906	\exp_args:NNv ..... 1184
\dim_max:nn ..... 4338, 4342	\exp_args:Nnx ..... 4567, 4574, 4649, 4663
\dim_min:nn ..... 4330, 4334	\exp_args:No ..... 2921
\dim_ratio:nn ..... 2853, 2904, 3020, 3025, 3036, 3044, 3053, 3058, 3069, 3077	\exp_args:NV ..... 1431, 1880, 1907, 1909
\dim_set:Nn ..... 4311, 4318, 4329, 4333, 4337, 4341, 4353, 4354, 4363, 4364, 4408, 4421	\exp_args:Nxx ..... 4533, 4534
\dim_set_eq:NN ..... 4309, 4316, 4416, 4430	\exp_last_unbraced:NV ..... 2369
\dim_sub:Nn ..... 4433	\exp_not:N ..... 42, 43, 46, 47, 1521, 1565, 3733, 3743, 4190, 4191, 4751
\dim_use:N ..... 4350, 4360, 4411, 4412, 4424, 4425, 4448, 4449, 4450, 4451, 4475, 4476, 4477, 4478	\exp_not:n ..... 966, 2382, 3280, 3356, 3735, 4549, 4625, 4637, 4638, 4740, 4752, 4766, 5052, 5053
\dim_zero_new:N ..... 4308, 4310, 4315, 4317	\ExplSyntaxOff . 217, 2258, 2279, 2305, 2384, 4289
\c_max_dim ..... 4309, 4311, 4316, 4318, 4350, 4360, 4787, 4800, 4805, 4820, 5130, 5131, 5242, 5243	\ExplSyntaxOn . 214, 2244, 2265, 2284, 2377, 4282
\l_tmpc_dim ..... 280, 3699, 3700, 3719, 3850, 3858, 3863, 3868, 3874, 3971, 3982, 3987, 3993, 4779, 4785, 4827, 4935, 4946, 5074, 5077, 5085	\extrarowheight ..... 4562, 4645, 4658, 5265
\l_tmpd_dim ..... 281, 3717, 3719, 3859, 3863, 3978, 3982, 4781, 4785, 4805, 4816, 4820, 4823, 4827, 4944, 4947, 5076, 5077, 5089	<b>F</b>
\dotfill ..... 1125, 5031	\fi ..... 110, 1433, 4009, 4026
\dots ..... 1051	fi commands:
\doublerulesep 1572, 3860, 3872, 3979, 3991, 4022	\fi: ..... 252
\doublerulesepcolor ..... 102	\fontdimen ..... 1730
\draw ..... 2933	fp commands:
<b>E</b>	\fp_eval:n ..... 2961
\egroup ..... 1421	\fp_to_dim:n ..... 2996
else commands:	\futurelet ..... 115
\else: ..... 250	<b>G</b>
\endarray ..... 1884, 1912	\globaldefs ..... 1417, 4716
\endBNiceMatrix ..... 4977	group commands:
\endbNiceMatrix ..... 4974	\group_insert_after:N 5036, 5037, 5039, 5040
	<b>H</b>
	\halign ..... 1101
	\hbox ..... 995, 1387, 1799, 1838, 1957, 1975, 2002, 2006, 2032
	hbox commands:
	\hbox:n ..... 64, 66, 69
	\hbox_gset:Nn ..... 4554
	\hbox_overlap_left:n ..... 1936, 2109
	\hbox_overlap_right:n ..... 376, 2053, 2155
	\hbox_set:Nn ..... 359, 1307, 1309, 1378, 1844, 1859, 4769
	\hbox_set:Nw ..... 816, 1315, 1609, 2084, 2130
	\hbox_set_end: . 912, 1330, 1617, 2103, 2148
	\hbox_to_wd:nn ..... 401, 426
	\Hdotsfor ..... 1119, 5544, 5700



O	
\omit	123, 1935, 1951, 2027, 5106
\OnlyMainNiceMatrix	1124, 3746
P	
\par	1754, 1762
peek commands:	
\peek_meaning:NTF	158, 357, 1040
\peek_meaning_ignore_spaces:NTF	1879, 4012
\peek_meaning_remove_ignore_spaces:NTF	148
\peek_remove_spaces:n	3228
\pgfextracty	4850
\pgfgetlastxy	419
\pgfpathcircle	3085
\pgfpathlineto	3868, 3874, 3987, 3993, 5077, 5305, 5320, 5352
\pgfpathmoveto	3867, 3873, 3986, 3992, 5072, 5304, 5319, 5343
\pgfpathrectanglecorners	3718, 3861, 3980, 4945
\pgfpointadd	417
\pgfpointanchor	233, 2614, 2625, 4328, 4336, 4795, 4813, 4832, 4834, 4851, 4885, 5137, 5251, 5258, 5274, 5281
\pgfpointdiff	418, 1216, 1224
\pgfpointlineatime	2955
\pgfpointorigin	1942, 2065
\pgfpointscale	417
\pgfpointshapeborder	3465, 3468
\pgfrememberpicturepositiononpagetrue	845, 938, 1007, 1941, 1965, 1983, 2014, 2040, 2063, 2311, 2424, 2443, 2912, 3464, 3843, 3964, 4199, 4244, 4371, 4381, 4392, 4772, 4919, 5067, 5123, 5239
\pgfscope	2952, 5084
\pgfset	386, 411, 939, 4862, 4896, 5083, 5144, 5241
\pgfsetbaseline	937
\pgfsetlinewidth	3877, 3996, 4948, 5293
\pgfsetrectcap	3878, 3997
\pgfsetroundcap	5080
\pgfsetstrokecolor	4925
\pgfsyspdfmark	215, 216
\pgftransformrotate	2959
\pgftransformshift	392, 417, 2953, 4861, 4883, 5085, 5089, 5146, 5366, 5376
\pgfusepath	2979, 2989
\pgfusepathqfill	3091, 3494, 3638, 3650, 3864, 3983
\pgfusepathqstroke	3879, 3998, 4949, 5081, 5306, 5321, 5353
\phantom	3110, 3126, 3142, 3166, 3189
\pNiceMatrix	4964
prg commands:	
\prg_do_nothing:	176, 185, 191, 219, 487, 1083, 2367
\prg_new_conditional:Nnn	3665, 3673
\prg_replicate:nn	365, 2023, 2024, 3283, 3869, 3988, 4994, 4997, 5000, 5006
\prg_return_false:	3670, 3682
\prg_return_true:	3671, 3681
\ProcessKeysOptions	5480
\ProvideDocumentCommand	59
\ProvidesExplPackage	4

Q	
\quad	346
quark commands:	
\q_stop	120, 121, 137, 138, 159, 161, 1192, 1443, 1510, 1657, 1896, 1899, 3418, 3432, 3433, 3498, 3544, 3549, 3605, 3690, 3691, 3707, 3708, 4465, 4472, 4499, 4502, 4927, 4936, 4979, 4988, 5225, 5230, 5295
R	
\rectanglecolor	1238, 2375, 3733, 4751
\refstepcounter	374
regex commands:	
\regex_const:Nn	209
\regex_match:nnTF	5206
\regex_replace_all:NnN	1448
\relax	153, 154
\renewcommand	195
\RenewDocumentEnvironment	4963, 4966, 4969, 4972, 4975
\RequirePackage	1, 3, 9, 10, 11
\right	1399, 1854, 1866, 5169, 5409, 5433
\rotate	1123
\rowcolor	1057, 1239
\rowcolors	1240
S	
\savenotes	1150
scan commands:	
\scan_stop:	2370
\scriptstyle	820, 2086, 2132, 2940, 2941, 2975, 2985
seq commands:	
\seq_clear:N	1441
\seq_clear_new:N	1244, 2286, 4074
\seq_count:N	1906, 3481
\seq_gclear:N	1177, 1178, 1179, 1180, 1782, 2368
\seq_gclear_new:N	1131, 1132, 1902, 1918
\seq_gpop_left:NN	1908, 1920
\seq_gput_left:Nn	600, 3232, 3234, 4730
\seq_gput_right:Nn	356, 1518, 2594, 3235, 4622, 4634, 4744, 5055, 5212
\seq_gset_from_clist:Nn	2289, 2301
\seq_gset_map_x:NNn	2395
\seq_gset_split:Nnn	1904, 1919
\seq_if_empty:NTF	1742
\seq_if_empty_p:N	2341, 2342, 2343
\seq_if_exist:NTF	1195
\seq_if_in:NnTF	598, 3820, 3826, 3833, 3941, 3947, 3954, 4331, 4339, 4793, 4811, 5208, 5530
\seq_item:Nn	1199, 1202, 1211, 1212, 1219, 1220
\seq_map_function:NN	1910
\seq_map_indexed_inline:Nn	3476, 3490
\seq_map_inline:Nn	1760, 1766, 1922, 3625, 3781, 3783, 3785, 3903, 3905, 3907, 4159, 4696, 5297
\seq_mapthread_function:NNN	4460
\seq_new:N	241, 255, 285, 286, 287, 288, 289, 316, 5520
\seq_put_right:Nn	3480, 4146
\seq_set_eq:NN	3594
\seq_set_filter:NNn	3596, 3623

<code>\seq_set_from_clist:Nn</code> .....	5521	<code>\@array@array</code> .....	975
<code>\seq_set_map_x:NNn</code> .....	5526	<code>\@arrayacol</code> .....	971, 972, 973
<code>\seq_use:Nnnn</code> .....	2303, 5696, 6054	<code>\@arstrutbox</code> .....	842, 843,
<code>\l_tmpa_seq</code> .....	3596, 3623	882, 1090, 1092, 1094, 1097, 1099, 1592, 1596	
<code>\l_tmpb_seq</code> .....	3594, 3596, 3623, 3625	<code>\@currenvir</code> .....	5114
<code>\setlist</code> .....	332, 343, 636, 641, 652, 657	<code>\@depth</code> .....	5166, 5406, 5430
skip commands:		<code>\@gobblethree</code> .....	216
<code>\skip_gadd:Nn</code> .....	1999	<code>\@halignto</code> .....	974, 986, 987
<code>\skip_gset:Nn</code> .....	1990	<code>\@height</code> ....	113, 128, 145, 5164, 5405, 5429
<code>\skip_gset_eq:NN</code> .....	1997, 1998	<code>\@ifclassloaded</code> .....	51, 54, 5501, 5511
<code>\skip_horizontal:N</code> .....	129,	<code>\@ifnextchar</code> .....	1136
146, 1316, 1317, 1328, 1329, 1353, 1354,		<code>\@ifpackageloaded</code> .....	
1390, 1391, 1394, 1395, 1408, 1409, 1438,		.....	30, 33, 36, 39, 75, 91, 166, 5504, 5514
1521, 1665, 1678, 1871, 1872, 1874, 1875,		<code>\@mainaux</code> ....	76, 212, 2244, 2245, 2252,
1946, 1947, 1959, 1961, 1977, 1979, 2001,		2258, 2265, 2266, 2273, 2279, 2284, 2285,	
2008, 2010, 2029, 2034, 2036, 2055, 2056,		2287, 2305, 2377, 2378, 2384, 4282, 4283, 4289	
2114, 2115, 2116, 2119, 2154, 2159, 2160, 2161		<code>\@tabarray</code> .....	988
<code>\skip_horizontal:n</code> .....	377, 1567	<code>\@tempswafalse</code> .....	1433
<code>\skip_vertical:N</code> .....		<code>\@tempswatrue</code> .....	1432
133, 1001, 1003, 1386, 1397, 1751, 1776, 4178		<code>\@temptokena</code> ..	175, 178, 197, 199, 1431, 1443
<code>\skip_vertical:n</code> .....	882, 4019	<code>\@whiles</code> .....	1433
<code>\g_tmpa_skip</code> 1990, 1997, 1998, 1999, 2001, 2029		<code>\@width</code> .....	113, 5167, 5407, 5431
<code>\c_zero_skip</code> .....	1088	<code>\@xhline</code> .....	116
<code>\space</code> .....	263, 264	<code>\bBigg@</code> .....	1307, 1309
<code>\stepcounter</code> .....	363, 368	<code>\c@MaxMatrixCols</code> .....	2200, 5558
str commands:		<code>\c@tabularnote</code> .....	1744, 1755, 1783
<code>\c_backslash_str</code> .....	263	<code>\col@sep</code> ....	983, 984, 1353, 1409, 1946,
<code>\c_colon_str</code> .....	745	1999, 2055, 2119, 2154, 2658, 2667, 2700, 2707	
<code>\str_case:nn</code> .....		<code>\CT@arc</code> .....	95, 96
... 3211, 4854, 4866, 4888, 4900, 5337, 5346		<code>\CT@arc@</code> .....	
<code>\str_case:nnTF</code> .....	1358, 1486, 1716, 4077	... 94, 99, 114, 127, 144, 160, 162, 1167,	
<code>\str_clear_new:N</code> .....	5236	1777, 2389, 3876, 3995, 4243, 4924, 5079, 5296	
<code>\str_foldcase:n</code> .....	3211	<code>\CT@drs</code> .....	102, 103
<code>\str_gclear:N</code> .....	2387	<code>\CT@drsc@</code> ..	101, 106, 3853, 3856, 3974, 3977
<code>\str_gset:Nn</code> .....		<code>\CT@everycr</code> .....	1081
... 1160, 2172, 2181, 2210, 2221, 2229, 5015		<code>\CT@row@color</code> .....	1083
<code>\str_if_empty:NTF</code> .....		<code>\if@temp</code> swa .....	1433
. 849, 950, 1010, 1159, 1264, 1282, 1943,		<code>\NC@</code> .....	1039
1968, 1986, 2017, 2043, 2066, 2171, 2180,		<code>\NC@find</code> .....	176, 204
2250, 2271, 2308, 4452, 4479, 5358, 5372, 5382		<code>\NC@list</code> .....	1433
<code>\str_if_eq:nnTF</code> ....	84, 262, 990, 1513,	<code>\NC@rewrite@S</code> .....	177, 195
1516, 1560, 1672, 1891, 1893, 1926, 4923, 5114		<code>\new@ifnextchar</code> .....	1136
<code>\str_if_eq_p:nn</code> .....		<code>\newcol@</code> .....	1041, 1042
476, 1526, 1654, 1655, 1656, 1657, 4507, 4512		<code>\nicematrix@redefine@check@rerun</code> ..	76, 79
<code>\str_if_in:NnTF</code> .....	1704, 1808	<code>\pgf@relevantforpicturesizefalse</code> ....	
<code>\str_new:N</code> .....	258, 458, 744	2425, 2913, 3082, 3489, 3600, 3844, 3965,	
<code>\str_range:Nnn</code> .....	1708, 1812	4372, 4382, 4393, 4773, 4920, 5066, 5124, 5240	
<code>\str_set:Nn</code> .....	597, 733, 5211	<code>\pgfsys@getposition</code> .....	1208, 1214, 1222
<code>\str_set_eq:NN</code> .....	601, 745	<code>\pgfsys@markposition</code> .....	
<code>\l_tmpa_str</code> .....	597, 598, 600, 601	1002, 1207, 1939, 1960, 1978, 2009, 2035, 2059	
<code>\strut</code> .....	1760, 1766	<code>\pgfutil@check@rerun</code> .....	81, 82
<code>\strutbox</code> .....	5265, 5268	<code>\reserved@a</code> .....	115
<code>\SubMatrix</code> ...	2366, 5631, 5663, 5670, 5689, 5815	<code>\rvtx@ifformat@geq</code> .....	57
		<code>\set@color</code> .....	4558
		<code>\tikz@library@external@loaded</code> .....	1168
		tex commands:	
		<code>\tex_mkern:D</code> .....	63, 65, 67, 70
		<code>\tex_the:D</code> .....	199
		<code>\textfont</code> .....	1730
		<code>\textit</code> .....	319
		<code>\textsuperscript</code> .....	320, 321
		<code>\the</code> .....	153, 154, 1433, 1443
		<code>\thetabularnote</code> .....	322

## T

<code>\tabcolsep</code> .....	983, 1390, 1394
<code>\tabskip</code> .....	1088
<code>\tabularnote</code> .....	327, 350, 357, 5747, 5760
<code>\tabularnotes</code> .....	1765
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
<code>\@BTnormal</code> .....	1063
<code>\@acol</code> .....	973
<code>\@acoll</code> .....	971
<code>\@acolr</code> .....	972





<b>4</b>	<b>The blocks</b>	<b>4</b>
4.1	General case . . . . .	4
4.2	The mono-column blocks . . . . .	5
4.3	The mono-row blocks . . . . .	5
4.4	The mono-cell blocks . . . . .	5
4.5	A small remark . . . . .	6
<b>5</b>	<b>The rules</b>	<b>6</b>
5.1	Some differences with the classical environments . . . . .	7
5.1.1	The vertical rules . . . . .	7
5.1.2	The command <code>\cline</code> . . . . .	7
5.2	The thickness and the color of the rules . . . . .	8
5.3	The keys <code>hlines</code> and <code>vlines</code> . . . . .	8
5.4	The key <code>hlines</code> . . . . .	8
5.5	The key <code>hlines-except-corners</code> . . . . .	9
5.6	The command <code>\diagbox</code> . . . . .	9
5.7	Dotted rules . . . . .	10
<b>6</b>	<b>The color of the rows and columns</b>	<b>10</b>
6.1	Use of <code>colortbl</code> . . . . .	10
6.2	The tools of <code>nicematrix</code> in the code-before . . . . .	11
6.3	Color tools with the syntax of <code>colortbl</code> . . . . .	13
<b>7</b>	<b>The width of the columns</b>	<b>14</b>
<b>8</b>	<b>The exterior rows and columns</b>	<b>15</b>
<b>9</b>	<b>The continuous dotted lines</b>	<b>16</b>
9.1	The option <code>nullify-dots</code> . . . . .	18
9.2	The commands <code>\Hdotsfor</code> and <code>\Vdotsfor</code> . . . . .	18
9.3	How to generate the continuous dotted lines transparently . . . . .	19
9.4	The labels of the dotted lines . . . . .	20
9.5	Customisation of the dotted lines . . . . .	20
9.6	The dotted lines and the rules . . . . .	21
<b>10</b>	<b>The <code>\CodeAfter</code></b>	<b>21</b>
10.1	The command <code>\line</code> in the <code>\CodeAfter</code> . . . . .	22
10.2	The command <code>\SubMatrix</code> in the <code>\CodeAfter</code> . . . . .	22
<b>11</b>	<b>The notes in the tabulars</b>	<b>23</b>
11.1	The footnotes . . . . .	23
11.2	The notes of tabular . . . . .	23
11.3	Customisation of the tabular notes . . . . .	25
11.4	Use of <code>{NiceTabular}</code> with <code>threeparttable</code> . . . . .	27
<b>12</b>	<b>Other features</b>	<b>27</b>
12.1	Use of the column type <code>S</code> of <code>siunitx</code> . . . . .	27
12.2	Alignment option in <code>{NiceMatrix}</code> . . . . .	28
12.3	The command <code>\rotate</code> . . . . .	28
12.4	The option <code>small</code> . . . . .	28
12.5	The counters <code>iRow</code> and <code>jCol</code> . . . . .	29
12.6	The option <code>light-syntax</code> . . . . .	30
12.7	Color of the delimiters . . . . .	30
12.8	The environment <code>{NiceArrayWithDelims}</code> . . . . .	30
<b>13</b>	<b>Use of <code>Tikz</code> with <code>nicematrix</code></b>	<b>30</b>
13.1	The nodes corresponding to the contents of the cells . . . . .	30
13.2	The “medium nodes” and the “large nodes” . . . . .	31
13.3	The nodes which indicate the position of the rules . . . . .	32
13.4	The nodes corresponding to the command <code>\SubMatrix</code> . . . . .	33

<b>14</b>	<b>API for the developpers</b>	<b>34</b>
<b>15</b>	<b>Technical remarks</b>	<b>34</b>
15.1	Definition of new column types . . . . .	34
15.2	Diagonal lines . . . . .	35
15.3	The “empty” cells . . . . .	35
15.4	The option exterior-arraycolsep . . . . .	36
15.5	Incompatibilities . . . . .	36
<b>16</b>	<b>Examples</b>	<b>36</b>
16.1	Notes in the tabulars . . . . .	36
16.2	Dotted lines . . . . .	38
16.3	Dotted lines which are no longer dotted . . . . .	38
16.4	Width of the columns . . . . .	39
16.5	How to highlight cells of a matrix . . . . .	40
16.6	Direct use of the Tikz nodes . . . . .	43
<b>17</b>	<b>Implementation</b>	<b>44</b>
<b>18</b>	<b>History</b>	<b>181</b>
	<b>Index</b>	<b>187</b>