

# The package `nicematrix`\*

F. Pantigny  
 fpantigny@wanadoo.fr

February 5, 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.

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

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.

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).

## Important

Since the version 5.0 of `nicematrix`, one must use the letters `l`, `c` and `r` in the preambles of the environments and no longer the letters `L`, `C` and `R`.

For sake of compatibility with the previous versions, there exists an option `define-L-C-R` which must be used when loading `nicematrix`.

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

---

\*This document corresponds to the version 5.10 of `nicematrix`, at the date of 2021/02/05.

# 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>1</sup>

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

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

---

<sup>1</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.

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

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

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>2</sup>

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

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

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

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

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

**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>2</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>3</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>3</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>4</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>5</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>4</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>5</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`).<sup>6</sup>

```
\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

<sup>6</sup>This is the behaviour since the version 5.1 of `nicematrix`. Prior to that version, the behaviour was the standard behaviour of `array`.

## 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>

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

$\begin{smallmatrix} y \\ \diagdown \\ x \end{smallmatrix}$	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

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}
```

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

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)
```

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

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`.

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 row and  $j$  the number of 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).

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

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

- 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>9</sup>. The *s* emphasizes the fact that there is *two* colors. This command colors alternately the rows of the tabular with the two colors (provided in second and third argument), beginning with the row whose number is given in first (mandatory) argument.

**New 5.8** 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`.

- **New 5.8** 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*.
- **New 5.8** With the key `restart`, each interval of rows (specified by the first mandatory argument) begins with the same color.<sup>10</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`).

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

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

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

```
$\begin{pNiceMatrix}[r,margin, code-before=\chessboardcolors{red!15}{blue!15}]
1 & -1 & 1 \\\
-1 & 1 & -1 \\\
1 & -1 & 1
\end{pNiceMatrix}$
```

1	-1	1
-1	1	-1
1	-1	1

<sup>9</sup>The command `\rowcolors` of `xcolor` is available when `xcolor` is loaded with the option `table`.

<sup>10</sup>Otherwise, the color of a given row relies only upon the parity of its number.

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).

```
\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{rl}{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>11</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).

```
\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}
```

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

<sup>11</sup>As of now, this key is not available in `\NiceMatrixOptions`.

## 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>12</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>13</sup>. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 4).

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

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

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

<sup>12</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>13</sup>At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

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}
 $\end{pNiceMatrix}$ 

```

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

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

```

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

### 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).

## 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>14</sup>

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>15</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

<sup>14</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`.

<sup>15</sup>The precise definition of a “non-empty cell” is given below (cf. p. 35).



to change the color of these lines with the option `color`.<sup>16</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}
```

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

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}
```

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

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}
```

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

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

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

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>17</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}
```

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

<sup>16</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>17</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 when the package `colortbl` is loaded (but you might have problem if you use `\rowcolor` on the same row as `\Hdotsfor`).

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
& \Hdotsfor{1} & \Vdots & \Ddots & \Vdots \\
C[a_n^{(p)},a_1] & \Cdots & C[a_n^{(p)},a_n]
& & C[a_n^{(p)},a_1^{(p)}] & \Cdots & C[a_n^{(p)},a_n^{(p)}] \\
\end{bNiceMatrix}
```

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

### 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>18</sup>

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>14</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>18</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 & \cdots & 1 & \\
0 & \ddots & & & \vdots \\
\vdots & \ddots & \ddots & \vdots & \\
0 & \cdots & 0 & & 1
\end{pmatrix}
\end{pmatrix}
```

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

## 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. 21) 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 \\
& \Ddots^{n \text{ times}} & & \\
0 & & & 1
\end{bNiceMatrix}
```

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

## 9.5 Customization 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. 21) 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` 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>19</sup>

<sup>19</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`:

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

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

## 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>20</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`.

Two special commands are available in the `\CodeAfter`: `line` and `\SubMatrix`.

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

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

<sup>20</sup>There is also a key `code-before` described p. 11.

```

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

```

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

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

**New 5.10** The command `\SubMatrix` provides a way to put delimiters on an 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, \lgroup, \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.

```

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

```

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

```

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

```

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.



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.

```

\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}

```

### 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` est une 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 customization 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{bNiceMatrix}[r]
\cos x & - \sin x \\
\sin x & \cos x
\end{bNiceMatrix}
```

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

## 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}
```

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

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 & \text{image of } e_2 & \text{image of } 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{bNiceArray}{cccc|c}[small,
last-col,
code-for-last-col = \scriptscriptstyle,
columns-width = 3mm ]
1 & -2 & 3 & 4 & 5 \\
0 & 3 & 2 & 1 & 2 & L_2 \text{ gets } 2L_1 - L_2 \\
0 & 1 & 1 & 2 & 3 & L_3 \text{ gets } L_1 + L_3
\end{bNiceArray}
```

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

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>21</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).

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

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

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-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>21</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>22</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`.

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

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

## 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>23</sup>

<sup>22</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>23</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. 41).

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

## 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>24</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>25</sup>

<sup>24</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>25</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>26</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 (with-out use of `margin` nor `extra-margin`).

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

### 13.3 The “row-nodes” and the “col-nodes”

The package `nicematrix` creates a PGF/Tikz node indicating the potential position of each horizontal rule (with the names `row-i`) and each vertical rule (with the names `col-j`), as described in the following figure. These nodes are available in the `code-before` and the `\CodeAfter`.

row-1	rose	tulipe	lys
row-2	arum	iris	violette
row-3	muguet	dahlia	souci
row-4	col-1	col-2	col-3

<sup>26</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.



If we use Tikz (we remind that `nicematrix` does not load Tikz by default), we can access (in the `code-before` and the `\CodeAfter`) to the intersection of the horizontal rule  $i$  and the vertical rule  $j$  with the syntax `(row- $i$ -|col- $j$ )`.

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

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`

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}\}`).

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

## 14 API for the developpers

The package `nicematrix` provides two variables which are internal but public<sup>27</sup>:

- `\g_nicematrix_code_before_tl` ;

<sup>27</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.

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

\NewDocumentCommand \hatchcell { ! O { black } }
{
  \tl_gput_right:Nx \g_nicematrix_code_before_tl
  {
    \__pantigny_hatchcell:nnn
    { \int_use:c { c@iRow } }
    { \int_use:c { c@jCol } }
    { #1 }
  }
}
\ExplSyntaxOff
```

Here is an example of use:

```
\begin{NiceTabular}{ccc}[hvlines]
Tokyo & Paris & London \\
Roma & \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 an 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>28</sup>:

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

<sup>28</sup>The command `\vrule` is a TeX (and not LaTeX) command.

The heavy vertical rule won't extend in the exterior rows.<sup>29</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}
```

$$\begin{pmatrix} C_1 & C_2 & C_3 & C_4 \\ a & b & c & d \\ e & f & g & h \\ C_1 & C_2 & C_3 & C_4 \end{pmatrix}$$

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>30</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}
```

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

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

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

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

The same example without parallelization:

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

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>29</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>30</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>31</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>32</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 of 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>31</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>32</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>33</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>34</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>33</sup>Of course, it's realistic only when there is very few notes in the tabular.

<sup>34</sup>In fact: the value of its argument.

## 16.2 Dotted lines

A permutation matrix (as an example, we have raised the value of `xdots/shorten`).

```

 $\begin{pNiceMatrix}[xdots/shorten=0.6em]$ 
0 & 1 & 0 & & \Cdots & 0 & \\
\Vdots & & & \Ddots & & \Vdots & \\
& & & \Ddots & & & \\
& & & \Ddots & & 0 & \\
0 & 0 & & & & 1 & \\
1 & 0 & & \Cdots & & 0 & \\
 $\end{pNiceMatrix}$ 

```

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

An example with `\Iddots` (we have raised again the value of `xdots/shorten`).

```

 $\begin{pNiceMatrix}[xdots/shorten=0.9em]$ 
1 & \Cdots & & 1 & \\
\Vdots & & & 0 & \\
& \Iddots & \Iddots & \Vdots & \\
1 & 0 & & \Cdots & 0 \\
 $\end{pNiceMatrix}$ 

```

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

An example with `\multicolumn`:

```

 $\begin{BNiceMatrix}[nullify-dots]$ 
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\Cdots & & \multicolumn{6}{C}{10 \text{ other rows}} & \Cdots \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
 $\end{BNiceMatrix}$ 

```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{pmatrix}$$

An example with `\Hdotsfor`:

```

 $\begin{pNiceMatrix}[nullify-dots]$ 
0 & 1 & 1 & 1 & 1 & 0 & \\
0 & 1 & 1 & 1 & 1 & 0 & \\
\Vdots & \Hdotsfor{4} & & \Vdots & \\
& \Hdotsfor{4} & & & \\
& \Hdotsfor{4} & & & \\
& \Hdotsfor{4} & & & \\
0 & 1 & 1 & 1 & 1 & 0 & \\
 $\end{pNiceMatrix}$ 

```

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

An example for the resultant of two polynoms:

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

$$\left| \begin{array}{ccccccc} a_0 & & & & & & \\ & \ddots & & & & & \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & \ddots & \\ & & & & & & a_p \end{array} \right| \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array} \left| \begin{array}{ccccccc} b_0 & & & & & & \\ & \ddots & & & & & \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & \ddots & \\ & & & & & & b_q \end{array} \right| \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array}$$

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}
```

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

### 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 no longer dotted.

```
\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]^{n \text{ columns}} \\
& 1 & 1 & 1 & \Ldots & 1 \\
& 1 & 1 & 1 & & 1
\end{pNiceMatrix}
```

```

\vdots[line-style={solid,<->}]_{{n} \text{ rows}} & 1 & 1 & 1 & & 1 \\
& 1 & 1 & 1 & & 1 \\
& 1 & 1 & 1 & \Ldots & 1 \\
\end{pNiceMatrix}$

```

$$\begin{array}{c}
\overbrace{\left( \begin{array}{ccccc} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & \dots & 1 \end{array} \right)}^{n \text{ columns}} \\
\updownarrow \\
n \text{ rows}
\end{array}$$

## 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}

```

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



## 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>35</sup>).

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

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

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>36</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). 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`.

That example and the following ones require Tikz (by default, `nicematrix` only loads PGF) 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}}
```

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

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

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

<sup>35</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

<sup>36</sup>For the command `\cline`, see the remark p. 7.

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.

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

In the following example, we use the “large nodes” to highlight a zone of the matrix.

```
\begin{pNiceArray}{>\strut}cccc[create-large-nodes,margin,extra-margin=2pt]
  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} \\
\CodeAfter
  \tikz \path [name suffix = -large,fill = red!15,blend mode = multiply]
    (1-1.north west)
    |- (2-2.north west)
    |- (3-3.north west)
    |- (4-4.north west)
    |- (4-4.south east)
    |- (1-1.north west) ;
\end{pNiceArray}
```

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

## 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}
& \\
& \end{array}
```

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}

```

$$L_i \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \quad \begin{bmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\ \vdots & & \vdots & & \vdots \\ b_{n1} & \cdots & b_{nj} & \cdots & b_{nn} \end{bmatrix}$$

## 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: 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.

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

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

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

## Technical definitions

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

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

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.

```

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

```

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. As of January 2021, the current version `revtex4-2` is 4.2e (compatible with `booktabs`).

```

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

```

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

```

56 \cs_if_exist:NT \rvtx@ifformat@geq { \bool_set_true:N \c_@@_revtex_bool }
57 \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.

```

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

```

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.

```

72 \AtBeginDocument
73 {
74   \@ifpackageloaded { booktabs }
75   { \iow_now:Nn \@mainaux \nicematrix@redefine@check@rerun }
76   { }
77 }

```

```

78 \cs_set_protected:Npn \nicematrix@redefine@check@rerun
79 {
80   \cs_set_eq:NN \@@_old_pgfulil@check@rerun \pgfulil@check@rerun

```

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

```

81   \cs_set_protected:Npn \pgfulil@check@rerun ##1 ##2
82   {
83     \str_if_eq:eeF { nm- } { \tl_range:nnn { ##1 } 1 3 }
84     { \@@_old_pgfulil@check@rerun { ##1 } { ##2 } }
85   }
86 }

```

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

```

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

```

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. Idem for

```

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

```

Idem for `\CT@drs@`.

```

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

```

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.

```

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

```

The following `\skip_horizontal:N \c_zero_dim` is to prevent a potential `\unskip` to delete the `\leaders`<sup>37</sup>

```
128 \skip_horizontal:N \c_zero_dim
129 }
```

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`.

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

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.

```
134 \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`.

```
135 { \@@_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*.

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

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).

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

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

```
147 \peek_meaning_remove_ignore_spaces:NTF \cline
148 { & \@@_cline_i:en { \@@_succ:n { #3 } } }
149 { \everycr { } \cr }
150 }
151 \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.

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

The following command is a small shortcut.

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

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

---

<sup>37</sup>See question 99041 on TeX StackExchange.



## 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`.

```

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

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:`.

```

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

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

```

175     \cs_set_eq:NN \NC@find \prg_do_nothing:
176     \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`).

```

177     \tl_gset:NV \g_tmpa_tl \@temptokena
178     \group_end:
```

```

179 \tl_new:N \c_@@_table_collect_begin_tl
180 \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
181 \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
182 \tl_new:N \c_@@_table_print_tl
183 \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).

```

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

```

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

```

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

```

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

```

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

```

208 \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 avoid that situation.

```

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

```

## 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 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.

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

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.

```
227 \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).

```
228 \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.

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

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

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

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

```
233 \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).

```
234 \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.

```
235 \tl_new:N \l_@@_cell_type_tl
236 \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`.

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

Idem pour the blocks mono-row.

```
238 \dim_new:N \g_@@_blocks_ht_dim
239 \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`.

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

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

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

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

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

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

```
243 \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.

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

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

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

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

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

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

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

```
253 \colorlet { nicematrix-last-col } { . }
254 \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`).

```
255 \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*.

```
256 \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).

```
257 \cs_new:Npn \@@_full_name_env:
258 {
259   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
260     { command \space \c_backslash_str \g_@@_name_env_str }
261     { environment \space \{ \g_@@_name_env_str \} }
262 }
```

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`).

```
263 \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.

```
264 \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.

```
265 \int_new:N \l_@@_old_iRow_int
```

```
266 \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.

```
267 \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.

```
268 \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).

```
269 \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).

```
270 \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`.

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

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

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

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

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

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

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

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

```
277 \dim_zero_new:N \l_tmpc_dim
```

```
278 \dim_zero_new:N \l_tmpd_dim
```

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

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

The following dimension will be used to save the current value of `\arraycolsep`.

```
280 \dim_new:N \g_@@_old_arraycolsep_dim
```

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

```
281 \dim_new:N \g_@@_width_last_col_dim
282 \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 braces:

`{imin}{jmin}{imax}{jmax}{options}{contents}`.

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

```
283 \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}`.

```
284 \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}`.

```
285 \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 `color=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.

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

The list of the names of the potential `\SubMatrix` in the `code-after` 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`).

```
287 \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).

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

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

```
289 \tl_new:N \l_@@_fill_tl
290 \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).

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

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

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

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

```
293 \tl_new:N \l_@@_pos_of_block_tl
294 \tl_set:Nn \l_@@_pos_of_block_tl { c }
```

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

```
295 \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.

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

## 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.

```
297 \int_new:N \l_@@_first_row_int
298 \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.

```
299 \int_new:N \l_@@_first_col_int
300 \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).

```
301 \int_new:N \l_@@_last_row_int
302 \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>38</sup>

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

Idem for `\l_@@_last_col_without_value_bool`

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

---

<sup>38</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.

- **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.

```
305 \int_new:N \l_@@_last_col_int
306 \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}
```

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.

```
307 \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.

```
308 \newcounter { tablarnote }
```

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

```
309 \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.

```
310 \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.  $a,b,c$ ).

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

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

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

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

```
313 \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`.

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



We define `\thetabularnote` 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`.

```
315 \cs_set:Npn \thetabularnote { { \@@_notes_style:n { tabularnote } } }
```

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 `tabularnotes` in the general case and a list `tabularnotes*` 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`).

```
316 \AtBeginDocument
317 {
318   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
319   {
320     \NewDocumentCommand \tabularnote { m }
321     { \@@_error:n { enumitem-not-loaded } }
322   }
323 }
```

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.

```
324   \newlist { tabularnotes } { enumerate } { 1 }
325   \setlist [ tabularnotes ]
326   {
327     topsep = 0pt ,
328     noitemsep ,
329     leftmargin = * ,
330     align = left ,
331     labelsep = 0pt ,
332     label =
333       \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
334   }
335   \newlist { tabularnotes* } { enumerate* } { 1 }
336   \setlist [ tabularnotes* ]
337   {
338     afterlabel = \nobreak ,
339     itemjoin = \quad ,
340     label =
341       \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
342   }
```

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>39</sup>

```
343   \NewDocumentCommand \tabularnote { m }
344   {
345     \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
346     { \@@_error:n { tabularnote-forbidden } }
347     {
```

`\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. <sup>a,b,c</sup>).

```
348       \int_incr:N \l_@@_number_of_notes_int
```

<sup>39</sup>We should try to find a solution to that problem.

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

```

349         \seq_gput_right:Nn \g_@@_tabularnotes_seq { #1 }
350         \peek_meaning:NF \tabularnote
351     {

```

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.

```

352         \hbox_set:Nn \l_tmpa_box
353     {

```

We remind that it is the command `@@_notes_label_in_tabular:n` that will (most of the time) put the labels in a `\textsuperscript`.

```

354         @@_notes_label_in_tabular:n
355     {
356         \stepcounter { tabularnote }
357         @@_notes_style:n { tabularnote }
358         \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
359     {
360         ,
361         \stepcounter { tabularnote }
362         @@_notes_style:n { tabularnote }
363     }
364 }
365 }

```

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.

```

366         \addtocounter { tabularnote } { -1 }
367         \refstepcounter { tabularnote }
368         \int_zero:N \l_@@_number_of_notes_int
369         \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).

```

370         \skip_horizontal:n { \box_wd:N \l_tmpa_box }
371     }
372 }
373 }
374 }
375 }

```

## 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.

```

376 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
377 {
378     \begin { pgfscope }
379     \pgfset
380     {
381         outer~sep = \c_zero_dim ,
382         inner~sep = \c_zero_dim ,
383         minimum~size = \c_zero_dim
384     }
385     \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
386     \pgfnode
387     { rectangle }

```

```

388     { center }
389     {
390         \vbox_to_ht:nn
391         { \dim_abs:n { #5 - #3 } }
392         {
393             \vfill
394             \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
395         }
396     }
397     { #1 }
398     { }
399 \end { pgfscope }
400 }

```

The command `\@@_pgf_rect_node:nnn` is a variant of `\@@_pgr_rect_node:nnnn`: it takes two PGF points as arguments instead of the four dimensions which are the coordinates.

```

401 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
402 {
403     \begin { pgfscope }
404     \pgfset
405     {
406         outer~sep = \c_zero_dim ,
407         inner~sep = \c_zero_dim ,
408         minimum~size = \c_zero_dim
409     }
410     \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
411     \pgfpointdiff { #3 } { #2 }
412     \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
413     \pgfnode
414     { rectangle }
415     { center }
416     {
417         \vbox_to_ht:nn
418         { \dim_abs:n \l_tmpb_dim }
419         { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
420     }
421     { #1 }
422     { }
423 \end { pgfscope }
424 }

```

## 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.

```

425 \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`.

```

426 \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`).

```

427 \dim_new:N \l_@@_cell_space_top_limit_dim
428 \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.

```
429 \dim_new:N \l_@@_inter_dots_dim
430 \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).

```
431 \dim_new:N \l_@@_xdots_shorten_dim
432 \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.

```
433 \dim_new:N \l_@@_radius_dim
434 \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.

```
435 \tl_new:N \l_@@_xdots_line_style_tl
436 \tl_const:Nn \c_@@_standard_tl { standard }
437 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

The boolean `\l_@@_light_syntax_bool` corresponds to the option `light-syntax`.

```
438 \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).

```
439 \tl_new:N \l_@@_baseline_tl
440 \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`).

```
441 \bool_new:N \l_@@_exterior_arraycolsep_bool
```

The flag `\l_@@_parallelize_diags_bool` controls whether the diagonals are parallelized. The initial value is `true`.

```
442 \bool_new:N \l_@@_parallelize_diags_bool
443 \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.

```
444 \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.

```
445 \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`.

```
446 \clist_new:N \l_@@_except_corners_clist
447 \dim_new:N \l_@@_notes_above_space_dim
448 \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.).

```
449 \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).

```
450 \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.

```
451 \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”.

```
452 \bool_new:N \l_@@_medium_nodes_bool
453 \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`).

```
454 \dim_new:N \l_@@_left_margin_dim
455 \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`.

```
456 \dim_new:N \l_@@_extra_left_margin_dim
457 \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.

```
458 \tl_new:N \l_@@_end_of_row_tl
459 \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 “:”.

```
460 \tl_new:N \l_@@_xdots_color_tl
```

The following token list corresponds to the key `delimiters-color`.

```
461 \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 function 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.

```

462 \bool_new:N \l_@@_max_delimiter_width_bool

463 \keys_define:nn { NiceMatrix / xdots }
464 {
465   line-style .code:n =
466     {
467       \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.

```

468     { \cs_if_exist_p:N \tikzpicture }
469     { \str_if_eq_p:nn { #1 } { standard } }
470     { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
471     { @@_error:n { bad-option-for~line-style } }
472   } ,
473   line-style .value_required:n = true ,
474   color .tl_set:N = \l_@@_xdots_color_tl ,
475   color .value_required:n = true ,
476   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
477   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 `_`.

```

478   down .tl_set:N = \l_@@_xdots_down_tl ,
479   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).

```

480   draw-first .code:n = \prg_do_nothing: ,
481   unknown .code:n = @@_error:n { Unknown-option-for~xdots }
482 }

```

```

483 \keys_define:nn { NiceMatrix / rules }
484 {
485   color .tl_set:N = \l_@@_rules_color_tl ,
486   color .value_required:n = true ,
487   width .dim_set:N = \arrayrulewidth ,
488   width .value_required:n = true
489 }

```

First, we define a set of keys “NiceMatrix / Global” which will be used (with the mechanism of `.inherit:n`) by other sets of keys.

```

490 \keys_define:nn { NiceMatrix / Global }
491 {
492   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
493   standard-cline .default:n = true ,
494   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
495   cell-space-top-limit .value_required:n = true ,
496   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
497   cell-space-bottom-limit .value_required:n = true ,
498   cell-space-limits .meta:n =
499     {
500       cell-space-top-limit = #1 ,

```

```

501     cell-space-bottom-limit = #1 ,
502   } ,
503   cell-spaces-limits .value_required:n = true ,
504   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
505   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
506   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
507   light-syntax .default:n = true ,
508   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
509   end-of-row .value_required:n = true ,
510   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
511   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
512   last-row .int_set:N = \l_@@_last_row_int ,
513   last-row .default:n = -1 ,
514   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
515   code-for-first-col .value_required:n = true ,
516   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
517   code-for-last-col .value_required:n = true ,
518   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
519   code-for-first-row .value_required:n = true ,
520   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
521   code-for-last-row .value_required:n = true ,
522   hlines .bool_set:N = \l_@@_hlines_bool ,
523   vlines .bool_set:N = \l_@@_vlines_bool ,
524   hvlines .code:n =
525   {
526     \bool_set_true:N \l_@@_vlines_bool
527     \bool_set_true:N \l_@@_hlines_bool
528   } ,
529   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.

```

530   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
531   renew-dots .value_forbidden:n = true ,
532   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
533   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
534   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
535   create-extra-nodes .meta:n =
536   { create-medium-nodes , create-large-nodes } ,
537   left-margin .dim_set:N = \l_@@_left_margin_dim ,
538   left-margin .default:n = \arraycolsep ,
539   right-margin .dim_set:N = \l_@@_right_margin_dim ,
540   right-margin .default:n = \arraycolsep ,
541   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
542   margin .default:n = \arraycolsep ,
543   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
544   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
545   extra-margin .meta:n =
546   { extra-left-margin = #1 , extra-right-margin = #1 } ,
547   extra-margin .value_required:n = true ,
548 }

```

We define a set of keys used by the environments of `nicematrix` (but not by the command `\NiceMatrixOptions`).

```

549 \keys_define:nn { NiceMatrix / Env }
550 {
551   except-corners .clist_set:N = \l_@@_except_corners_clist ,
552   except-corners .default:n = { NW , SW , NE , SE } ,
553   hvlines-except-corners .code:n =
554   {
555     \clist_set:Nn \l_@@_except_corners_clist { #1 }
556     \bool_set_true:N \l_@@_vlines_bool

```

```

557     \bool_set_true:N \l_@@_hlines_bool
558   } ,
559   hvlines-except-corners .default:n = { NW , SW , NE , SE } ,
560   code-before .code:n =
561   {
562     \tl_if_empty:nF { #1 }
563     {
564       \tl_put_right:Nn \l_@@_code_before_tl { #1 }
565       \bool_set_true:N \l_@@_code_before_bool
566     }
567   } ,

```

The options `c`, `t` and `b` of the environment `{NiceArray}` have the same meaning as the option of the classical environment `{array}`.

```

568   c .code:n = \tl_set:Nn \l_@@_baseline_tl c ,
569   t .code:n = \tl_set:Nn \l_@@_baseline_tl t ,
570   b .code:n = \tl_set:Nn \l_@@_baseline_tl b ,
571   baseline .tl_set:N = \l_@@_baseline_tl ,
572   baseline .value_required:n = true ,
573   columns-width .code:n =
574   { \tl_if_eq:nnTF { #1 } { auto }
575     { \bool_set_true:N \l_@@_auto_columns_width_bool }
576     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
577   columns-width .value_required:n = true ,
578   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.

```

579   \legacy_if:nF { measuring@ }
580   {
581     \str_set:Nn \l_tmpa_str { #1 }
582     \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
583     { \@@_error:nn { Duplicate~name } { #1 } }
584     { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
585     \str_set_eq:NN \l_@@_name_str \l_tmpa_str
586   } ,
587   name .value_required:n = true ,
588   code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
589   code-after .value_required:n = true ,
590   colortbl-like .code:n =
591   { \bool_set_true:N \l_@@_colortbl_like_bool
592     \bool_set_true:N \l_@@_code_before_bool ,
593     colortbl-like .value_forbidden:n = true
594   }
595   \keys_define:nn { NiceMatrix / notes }
596   {
597     para .bool_set:N = \l_@@_notes_para_bool ,
598     para .default:n = true ,
599     code-before .tl_set:N = \l_@@_notes_code_before_tl ,
600     code-before .value_required:n = true ,
601     code-after .tl_set:N = \l_@@_notes_code_after_tl ,
602     code-after .value_required:n = true ,
603     bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
604     bottomrule .default:n = true ,
605     style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
606     style .value_required:n = true ,
607     label-in-tabular .code:n =
608     { \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
609       label-in-tabular .value_required:n = true ,
610     label-in-list .code:n =
611     { \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
612       label-in-list .value_required:n = true ,
613     enumitem-keys .code:n =

```



```

614 {
615   \bool_if:NTF \c_@@_in_preamble_bool
616   {
617     \AtBeginDocument
618     {
619       \bool_if:NT \c_@@_enumitem_loaded_bool
620       { \setlist* [ tabularnotes ] { #1 } }
621     }
622   }
623   {
624     \bool_if:NT \c_@@_enumitem_loaded_bool
625     { \setlist* [ tabularnotes ] { #1 } }
626   }
627 },
628 enumitem-keys .value_required:n = true ,
629 enumitem-keys-para .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-para .value_required:n = true ,
645 unknown .code:n = \@@_error:n { Unknown~key~for~notes }
646 }

```

We begin the construction of the major sets of keys (used by the different user commands and environments).

```

647 \keys_define:nn { NiceMatrix }
648 {
649   NiceMatrixOptions .inherit:n =
650   { NiceMatrix / Global } ,
651   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
652   NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
653   NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
654   NiceMatrixOptions / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
655   CodeAfter / xdots .inherit:n = NiceMatrix / xdots ,
656   NiceMatrix .inherit:n =
657   {
658     NiceMatrix / Global ,
659     NiceMatrix / Env ,
660   } ,
661   NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
662   NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
663   NiceTabular .inherit:n =
664   {
665     NiceMatrix / Global ,
666     NiceMatrix / Env
667   } ,
668   NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
669   NiceTabular / rules .inherit:n = NiceMatrix / rules ,
670   NiceArray .inherit:n =
671   {
672     NiceMatrix / Global ,
673     NiceMatrix / Env ,

```

```

674     } ,
675     NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
676     NiceArray / rules .inherit:n = NiceMatrix / rules ,
677     pNiceArray .inherit:n =
678     {
679         NiceMatrix / Global ,
680         NiceMatrix / Env ,
681     } ,
682     pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
683     pNiceArray / rules .inherit:n = NiceMatrix / rules ,
684 }

```

We finalise the definition of the set of keys “NiceMatrix / NiceMatrixOptions” with the options specific to \NiceMatrixOptions.

```

685 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
686 {
687     delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
688     delimiters-color .value_required:n = true ,
689     last-col .code:n = \tl_if_empty:nF { #1 }
690         { \@@_error:n { last-col-non-empty-for-NiceMatrixOptions } }
691         \int_zero:N \l_@@_last_col_int ,
692     small .bool_set:N = \l_@@_small_bool ,
693     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.

```

694     renew-matrix .code:n = \@@_renew_matrix: ,
695     renew-matrix .value_forbidden:n = true ,

```

The key `transparent` is now considered as obsolete (because its name is ambiguous).

```

696     transparent .code:n =
697     {
698         \@@_renew_matrix:
699         \bool_set_true:N \l_@@_renew_dots_bool
700         \@@_error:n { Key~transparent }
701     } ,
702     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}`.

```

703     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.

```

704     columns-width .code:n =
705     \tl_if_eq:nnTF { #1 } { auto }
706     { \@@_error:n { Option~auto~for~columns-width } }
707     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

Usually, an error is raised when the user tries to give the same name to two distincts environments of `nicematrix` (theses names are global and not local to the current TeX scope). However, the option `allow-duplicate-names` disables this feature.

```

708     allow-duplicate-names .code:n =
709     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
710     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`).

```

711 letter-for-dotted-lines .code:n =
712 {
713   \tl_if_single_token:nTF { #1 }
714   { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
715   { \@@_error:n { Bad-value-for-letter-for-dotted-lines } }
716 },
717 letter-for-dotted-lines .value_required:n = true ,
718 notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
719 notes .value_required:n = true ,
720 sub-matrix .code:n =
721   \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
722 sub-matrix .value_required:n = true ,
723 unknown .code:n = \@@_error:n { Unknown-key-for-NiceMatrixOptions }
724 }

725 \str_new:N \l_@@_letter_for_dotted_lines_str
726 \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.

```

727 \NewDocumentCommand \NiceMatrixOptions { m }
728 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

We finalise the definition of the set of keys “NiceMatrix / NiceMatrix” with the options specific to `{NiceMatrix}`.

```

729 \keys_define:nn { NiceMatrix / NiceMatrix }
730 {
731   last-col .code:n = \tl_if_empty:nTF {#1}
732   {
733     \bool_set_true:N \l_@@_last_col_without_value_bool
734     \int_set:Nn \l_@@_last_col_int { -1 }
735   }
736   { \int_set:Nn \l_@@_last_col_int { #1 } } ,
737   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
738   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
739   small .bool_set:N = \l_@@_small_bool ,
740   small .value_forbidden:n = true ,
741   delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
742   delimiters-color .value_required:n = true ,
743   unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
744 }

```

We finalise the definition of the set of keys “NiceMatrix / NiceArray” with the options specific to `{NiceArray}`.

```

745 \keys_define:nn { NiceMatrix / NiceArray }
746 {

```

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.

```

747   small .bool_set:N = \l_@@_small_bool ,
748   small .value_forbidden:n = true ,
749   last-col .code:n = \tl_if_empty:nF { #1 }
750   { \@@_error:n { last-col-non-empty-for-NiceArray } }
751   \int_zero:N \l_@@_last_col_int ,
752   notes / para .bool_set:N = \l_@@_notes_para_bool ,
753   notes / para .default:n = true ,
754   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
755   notes / bottomrule .default:n = true ,
756   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
757   tabularnote .value_required:n = true ,
758   delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,

```

```

759     delimiters-color .value_required:n = true ,
760     unknown .code:n = \@@_error:n { Unknown-option-for-NiceArray }
761 }
762 \keys_define:nn { NiceMatrix / pNiceArray }
763 {
764     first-col .code:n = \int_zero:N \l_@@_first_col_int ,
765     last-col .code:n = \tl_if_empty:nF {#1}
766         { \@@_error:n { last-col-non-empty-for-NiceArray } }
767         \int_zero:N \l_@@_last_col_int ,
768     first-row .code:n = \int_zero:N \l_@@_first_row_int ,
769     small .bool_set:N = \l_@@_small_bool ,
770     small .value_forbidden:n = true ,
771     unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
772 }

```

We finalise the definition of the set of keys “NiceMatrix / NiceTabular” with the options specific to {NiceTabular}.

```

773 \keys_define:nn { NiceMatrix / NiceTabular }
774 {
775     notes / para .bool_set:N = \l_@@_notes_para_bool ,
776     notes / para .default:n = true ,
777     notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
778     notes / bottomrule .default:n = true ,
779     tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
780     tabularnote .value_required:n = true ,
781     last-col .code:n = \tl_if_empty:nF {#1}
782         { \@@_error:n { last-col-non-empty-for-NiceArray } }
783         \int_zero:N \l_@@_last_col_int ,
784     unknown .code:n = \@@_error:n { Unknown-option-for-NiceTabular }
785 }

```

## 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}).

```

786 \cs_new_protected:Npn \@@_Cell:
787 {

```

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).

```

788     \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n

```

We increment \c@jCol, which is the counter of the columns.

```

789     \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.

```

790     \int_compare:nNnT \c@jCol = 1
791     { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
792     \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).

```

793     \hbox_set:Nw \l_@@_cell_box
794     \bool_if:NF \l_@@_NiceTabular_bool
795     {
796         \c_math_toggle_token
797         \bool_if:NT \l_@@_small_bool \scriptstyle
798     }

```

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.

```

799   \int_compare:nNnTF \c@iRow = 0
800   {
801     \int_compare:nNnT \c@jCol > 0
802     {
803       \l_@@_code_for_first_row_tl
804       \xglobal \colorlet { nicematrix-first-row } { . }
805     }
806   }
807   {
808     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
809     {
810       \l_@@_code_for_last_row_tl
811       \xglobal \colorlet { nicematrix-last-row } { . }
812     }
813   }
814 }

```

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.

```

815 \cs_new_protected:Npn \@@_begin_of_row:
816 {
817   \int_gincr:N \c@iRow
818   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
819   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
820   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
821   \pgfpicture
822   \pgfrememberpicturepositiononpagetrue
823   \pgfcoordinate
824   { \@@_env: - row - \int_use:N \c@iRow - base }
825   { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
826   \str_if_empty:NF \l_@@_name_str
827   {
828     \pgfnodealias
829     { \l_@@_name_str - row - \int_use:N \c@iRow - base }
830     { \@@_env: - row - \int_use:N \c@iRow - base }
831   }
832   \endpgfpicture
833 }

```

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.

```

834 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
835 {
836   \int_compare:nNnTF \c@iRow = 0
837   {
838     \dim_gset:Nn \g_@@_dp_row_zero_dim
839     { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
840     \dim_gset:Nn \g_@@_ht_row_zero_dim
841     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
842   }
843   {
844     \int_compare:nNnT \c@iRow = 1
845     {
846       \dim_gset:Nn \g_@@_ht_row_one_dim
847       { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
848     }
849   }
850 }

```

```

849     }
850 }

851 \cs_new_protected:Npn \@@_rotate_cell_box:
852 {
853     \box_rotate:Nn \l_@@_cell_box { 90 }
854     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
855     {
856         \vbox_set_top:Nn \l_@@_cell_box
857         {
858             \vbox_to_zero:n { }
859             \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
860             \box_use:N \l_@@_cell_box
861         }
862     }
863     \bool_gset_false:N \g_@@_rotate_bool
864 }

865 \cs_new_protected:Npn \@@_adjust_size_box:
866 {
867     \dim_compare:nNnT \g_@@_blocks_wd_dim > \c_zero_dim
868     {
869         \box_set_wd:Nn \l_@@_cell_box
870         { \dim_max:nn { \box_wd:N \l_@@_cell_box } \g_@@_blocks_wd_dim }
871         \dim_gzero:N \g_@@_blocks_wd_dim
872     }
873     \dim_compare:nNnT \g_@@_blocks_dp_dim > \c_zero_dim
874     {
875         \box_set_dp:Nn \l_@@_cell_box
876         { \dim_max:nn { \box_dp:N \l_@@_cell_box } \g_@@_blocks_dp_dim }
877         \dim_gzero:N \g_@@_blocks_dp_dim
878     }
879     \dim_compare:nNnT \g_@@_blocks_ht_dim > \c_zero_dim
880     {
881         \box_set_ht:Nn \l_@@_cell_box
882         { \dim_max:nn { \box_ht:N \l_@@_cell_box } \g_@@_blocks_ht_dim }
883         \dim_gzero:N \g_@@_blocks_ht_dim
884     }
885 }

886 \cs_new_protected:Npn \@@_end_Cell:
887 {
888     \@@_math_toggle_token:
889     \hbox_set_end:
890     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
891     \@@_adjust_size_box:

892     \box_set_ht:Nn \l_@@_cell_box
893     { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
894     \box_set_dp:Nn \l_@@_cell_box
895     { \box_dp:N \l_@@_cell_box + \l_@@_cell_space_bottom_limit_dim }

```

We want to compute in  $\text{\g\_@@\_max\_cell\_width\_dim}$  the width of the widest cell of the array (except the cells of the “first column” and the “last column”).

```

896     \dim_gset:Nn \g_@@_max_cell_width_dim
897     { \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”.

```

898     \@@_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 of 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 `code-after`); 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.

```

899   \bool_if:NTF \g_@@_empty_cell_bool
900     { \box_use_drop:N \l_@@_cell_box }
901     {
902       \bool_lazy_or:nnTF
903         \g_@@_not_empty_cell_bool
904         { \dim_compare_p:nNn { \box_wd:N \l_@@_cell_box } > \c_zero_dim }
905         \@@_node_for_the_cell:
906         { \box_use_drop:N \l_@@_cell_box }
907     }
908   \bool_gset_false:N \g_@@_empty_cell_bool
909   \bool_gset_false:N \g_@@_not_empty_cell_bool
910 }

```

The following command creates the PGF name of the node with, of course, `\l_@@_cell_box` as the content.

```

911 \cs_new_protected:Npn \@@_node_for_the_cell:
912 {
913   \pgfpicture
914   \pgfsetbaseline \c_zero_dim
915   \pgfrememberpicturepositiononpagetrue
916   \pgfset
917   {
918     inner~sep = \c_zero_dim ,
919     minimum~width = \c_zero_dim
920   }
921   \pgfnode
922   { rectangle }
923   { base }
924   { \box_use_drop:N \l_@@_cell_box }
925   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
926   { }
927   \str_if_empty:NF \l_@@_name_str
928   {
929     \pgfnodealias
930     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
931     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
932   }
933   \endpgfpicture
934 }

```

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.

```
935 \cs_new_protected:Npn \@@_instruction_of_type:nnn #1 #2 #3  
936 {
```

It's important to use a `\tl_gput_right:cx` and not a `\tl_gput_left:cx` because we want the `\Ddots` lines to be drawn in the order of appearance in the array (for parallelisation).

```
937   \bool_if:NTF { #1 } \tl_gput_left:cx \tl_gput_right:cx  
938   { g_@@_ #2 _ lines _ tl }  
939   {  
940     \use:c { @@ _ draw _ #2 : nnn }  
941     { \int_use:N \c@iRow }  
942     { \int_use:N \c@jCol }  
943     { \exp_not:n { #3 } }  
944   }  
945 }
```

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`.

```
946 \cs_new_protected:Npn \@@_revtex_array:  
947 {  
948   \cs_set_eq:NN \@acol1 \@arrayacol  
949   \cs_set_eq:NN \@acolr \@arrayacol  
950   \cs_set_eq:NN \@acol \@arrayacol  
951   \cs_set_nopar:Npn \@halignto { }  
952   \@array@array  
953 }  
  
954 \cs_new_protected:Npn \@@_array:  
955 {  
956   \bool_if:NTF \c_@@_revtex_bool  
957   \@@_revtex_array:  
958   {  
959     \bool_if:NTF \l_@@_NiceTabular_bool  
960     { \dim_set_eq:NN \col@sep \tabcolsep }  
961     { \dim_set_eq:NN \col@sep \arraycolsep }  
962     \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim  
963     { \cs_set_nopar:Npn \@halignto { } }  
964     { \cs_set_nopar:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }
```

It `colortbl` is loaded, `\@tabarray` has been redefined to incorporate `\CT@start`.

```
965   \@tabarray  
966 }
```

`\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.

```
967   [ \str_if_eq:VnTF \l_@@_baseline_tl c c t ]  
968 }
```

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.

```
969 \cs_set_eq:NN \@@_old_ialign: \ialign
```

The following command creates a `row` node (and not a row of nodes!).

```
970 \cs_new_protected:Npn \@@_create_row_node:  
971 {
```



The `\hbox:n` (or `\hbox`) is mandatory.

```

972   \hbox
973   {
974     \bool_if:NT \l_@@_code_before_bool
975     {
976       \vtop
977       {
978         \skip_vertical:N 0.5\arrayrulewidth
979         \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
980         \skip_vertical:N -0.5\arrayrulewidth
981       }
982     }
983     \pgfpicture
984     \pgfrememberpicturerepositiononpagetrue
985     \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
986     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
987     \str_if_empty:NF \l_@@_name_str
988     {
989       \pgfnodealias
990       { \l_@@_name_str - row - \int_use:N \c@iRow }
991       { \@@_env: - row - \int_use:N \c@iRow }
992     }
993     \endpgfpicture
994   }
995 }

```

The following must *not* be protected because it begins with `\noalign`.

```

996 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
997 \cs_new_protected:Npn \@@_everycr_i:
998 {
999   \int_gzero:N \c@jCol
1000   \bool_gset_false:N \g_@@_after_col_zero_bool
1001   \bool_if:NF \g_@@_row_of_col_done_bool
1002   {
1003     \@@_create_row_node:

```

We don't draw the rules of the key `hlines` (or `hvlines`) but we reserve the vertical space for theses rules.

```

1004   \bool_if:NT \l_@@_hlines_bool
1005   {

```

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.

```

1006     \int_compare:nNnT \c@iRow > { -1 }
1007     {
1008       \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@`.

```

1009       { \hrule height \arrayrulewidth width \c_zero_dim }
1010     }
1011   }
1012 }
1013 }

```

The command `\@@_newcolumnntype` is the command `\newcolumnntype` of `array` without the warnings for redefinitions of columns types (we will use it to redefine the columns types `w` and `W`).

```

1014 \cs_set_protected:Npn \@@_newcolumnntype #1
1015 {
1016   \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
1017   \peek_meaning:NTF [

```

```

1018     { \newcol@ #1 }
1019     { \newcol@ #1 [ 0 ] }
1020 }

```

When the key `renew-dots` is used, the following code will be executed.

```

1021 \cs_set_protected:Npn \@@_renew_dots:
1022 {
1023     \cs_set_eq:NN \ldots \@@_Ldots
1024     \cs_set_eq:NN \cdots \@@_Cdots
1025     \cs_set_eq:NN \vdots \@@_Vdots
1026     \cs_set_eq:NN \ddots \@@_Ddots
1027     \cs_set_eq:NN \iddots \@@_Iddots
1028     \cs_set_eq:NN \dots \@@_Ldots
1029     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
1030 }

```

When the key `colortbl-like` is used, the following code will be executed.

```

1031 \cs_new_protected:Npn \@@_colortbl_like:
1032 {
1033     \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
1034     \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
1035     \cs_set_eq:NN \columncolor \@@_columncolor_preamble
1036 }

```

The following code `\@@_pre_array:` is used in `{NiceArrayWithDelims}`. It exists as a standalone macro only for legibility.

```

1037 \cs_new_protected:Npn \@@_pre_array:
1038 {

```

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>40</sup>.

```

1039     \bool_if:NT \c_@@_booktabs_loaded_bool
1040     { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
1041     \box_clear_new:N \l_@@_cell_box
1042     \cs_if_exist:NT \theiRow
1043     { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
1044     \int_gzero_new:N \c@iRow
1045     \cs_if_exist:NT \thejCol
1046     { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
1047     \int_gzero_new:N \c@jCol
1048     \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}`).

```

1049     \bool_if:NT \l_@@_small_bool
1050     {
1051         \cs_set_nopar:Npn \arraystretch { 0.47 }
1052         \dim_set:Nn \arraycolsep { 1.45 pt }
1053     }

```

---

<sup>40</sup>cf. `\nicematrix@redefine@check@rerun`

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`.

```

1054 \cs_set_nopar:Npn \ialign
1055 {
1056   \bool_if:NTF \c_@@_colortbl_loaded_bool
1057   {
1058     \CT@everycr
1059     {
1060       \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
1061       \@@_everycr:
1062     }
1063   }
1064   { \everycr { \@@_everycr: } }
1065   \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>41</sup> and `\extrarowheight` (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`.

```

1066 \dim_gzero_new:N \g_@@_dp_row_zero_dim
1067 \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
1068 \dim_gzero_new:N \g_@@_ht_row_zero_dim
1069 \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
1070 \dim_gzero_new:N \g_@@_ht_row_one_dim
1071 \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
1072 \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
1073 \dim_gzero_new:N \g_@@_ht_last_row_dim
1074 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
1075 \dim_gzero_new:N \g_@@_dp_last_row_dim
1076 \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`.

```

1077 \cs_set_eq:NN \ialign \@@_old_ialign:
1078 \halign
1079 }

```

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).

```

1080 \cs_set_eq:NN \@@_old_ldots \ldots
1081 \cs_set_eq:NN \@@_old_cdots \cdots
1082 \cs_set_eq:NN \@@_old_vdots \vdots
1083 \cs_set_eq:NN \@@_old_ddots \ddots
1084 \cs_set_eq:NN \@@_old_iddots \iddots
1085 \bool_if:NTF \l_@@_standard_cline_bool
1086 { \cs_set_eq:NN \cline \@@_standard_cline }
1087 { \cs_set_eq:NN \cline \@@_cline }
1088 \cs_set_eq:NN \Ldots \@@_Ldots
1089 \cs_set_eq:NN \Cdots \@@_Cdots
1090 \cs_set_eq:NN \Vdots \@@_Vdots
1091 \cs_set_eq:NN \Ddots \@@_Ddots
1092 \cs_set_eq:NN \Iddots \@@_Iddots
1093 \cs_set_eq:NN \hdottedline \@@_hdottedline:
1094 \cs_set_eq:NN \Hline \@@_Hline:
1095 \cs_set_eq:NN \Hspace \@@_Hspace:
1096 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
1097 \cs_set_eq:NN \Vdotsfor \@@_Vdotsfor:

```

<sup>41</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

```

1098 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
1099 \cs_set_eq:NN \Block \@@_Block:
1100 \cs_set_eq:NN \rotate \@@_rotate:
1101 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
1102 \cs_set_eq:NN \dotfill \@@_old_dotfill:
1103 \cs_set_eq:NN \CodeAfter \@@_CodeAfter:
1104 \cs_set_eq:NN \diagbox \@@_diagbox:nn
1105 \cs_set_eq:NN \NotEmpty \@@_NotEmpty:
1106 \bool_if:NT \l_@@_colortbl_like_bool \@@_colortbl_like:
1107 \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  $n$ ) correspondent will be stored. These lists will be used for the creation of the “medium nodes” (if they are created).

```

1108 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
1109 \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).

```

1110 \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`).

```

1111 \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.

```

1112 \int_gzero_new:N \g_@@_col_total_int
1113 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1114 \@@_renew_NC@rewrite@S:
1115 \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.

```

1116 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1117 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1118 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1119 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1120 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1121 \tl_gclear_new:N \g_@@_HVdotsfor_lines_tl

```

```

1122 \tl_gclear_new:N \g_nicematrix_code_before_tl
1123 }

```

This is the end of `\@@_pre_array:`.

## The environment `{NiceArrayWithDelims}`

```

1124 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
1125 {
1126 \@@_provide_pgfsyspdfmark:
1127 \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.

```

1128 \bgroup

1129 \tl_set:Nn \l_@@_left_delim_tl { #1 }

```

```

1130 \tl_set:Nn \l_@@_right_delim_tl { #2 }
1131 \tl_gset:Nn \g_@@_preamble_tl { #4 }

1132 \int_gzero:N \g_@@_block_box_int
1133 \dim_zero:N \g_@@_width_last_col_dim
1134 \dim_zero:N \g_@@_width_first_col_dim
1135 \bool_gset_false:N \g_@@_row_of_col_done_bool
1136 \str_if_empty:NT \g_@@_name_env_str
1137 { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1138 \@@_adapt_S_column:
1139 \bool_if:NTF \l_@@_NiceTabular_bool
1140 \mode_leave_vertical:
1141 \@@_test_if_math_mode:
1142 \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet-in-env } }
1143 \bool_set_true:N \l_@@_in_env_bool

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>42</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 beginning of arrays done by `colortbl`. Of course, we restore the value of `\CT@arc@` at the end of our environment.

```

1144 \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.

```

1145 \cs_if_exist:NT \tikz@library@external@loaded
1146 {
1147   \tikzexternaldisable
1148   \cs_if_exist:NT \ifstandalone
1149     { \tikzset { external / optimize = false } }
1150 }

```

We increment the counter `\g_@@_env_int` which counts the environments of the package.

```

1151 \int_gincr:N \g_@@_env_int
1152 \bool_if:NF \l_@@_block_auto_columns_width_bool
1153 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

The sequence `\g_@@_blocks_seq` will contain the carateristics 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.

```

1154 \seq_gclear:N \g_@@_blocks_seq
1155 \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`.

```

1156 \seq_gclear:N \g_@@_pos_of_stroken_blocks_seq
1157 \seq_gclear:N \g_@@_pos_of_xdots_seq

1158 \tl_if_exist:cT { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1159 {
1160   \bool_set_true:N \l_@@_code_before_bool
1161   \exp_args:NNv \tl_put_right:Nn \l_@@_code_before_tl
1162     { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1163 }

```

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`.

```

1164 \bool_if:NTF \l_@@_NiceArray_bool
1165 { \keys_set:nn { NiceMatrix / NiceArray } }
1166 { \keys_set:nn { NiceMatrix / pNiceArray } }
1167 { #3 , #5 }

```

---

<sup>42</sup>e.g. `\color[rgb]{0.5,0.5,0}`

```

1168 \tl_if_empty:NF \l_@@_rules_color_tl
1169 { \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.

```

1170 \bool_if:NT \l_@@_code_before_bool
1171 {
1172   \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1173   {

```

First, we give values to the LaTeX counters `iRow` and `jCol`. We remind that, in the `code-before` (and in the `code-after`) they represent the numbers of rows and columns of the array (without the potential last row and last column).

```

1174     \int_zero_new:N \c@iRow
1175     \int_set:Nn \c@iRow
1176     { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1177     \int_zero_new:N \c@jCol
1178     \int_set:Nn \c@jCol
1179     { \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.

```

1180     \int_compare:nNnF \l_@@_last_row_int = { -2 }
1181     { \int_decr:N \c@iRow }
1182     \int_compare:nNnF \l_@@_last_col_int = { -2 }
1183     { \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.

```

1184     \pgfsys@markposition { \@@_env: - position }
1185     \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1186     \pgfpicture

```

First, the creation of the `row` nodes.

```

1187     \int_step_inline:nnn
1188     { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1189     { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1190     {
1191       \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1192       \pgfcoordinate { \@@_env: - row - ##1 }
1193       { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1194     }

```

Now, the creation of the `col` nodes.

```

1195     \int_step_inline:nnn
1196     { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1197     { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1198     {
1199       \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1200       \pgfcoordinate { \@@_env: - col - ##1 }
1201       { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1202     }
1203   \endpgfpicture
1204   \group_begin:
1205     \bool_if:NT \c_@@_tikz_loaded_bool
1206     {
1207       \tikzset
1208       {
1209         every~picture / .style =
1210         { overlay , name~prefix = \@@_env: - }
1211       }
1212     }
1213     \cs_set_eq:NN \cellcolor \@@_cellcolor
1214     \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor

```

```

1215         \cs_set_eq:NN \rowcolor \@@_rowcolor
1216         \cs_set_eq:NN \rowcolors \@@_rowcolors
1217         \cs_set_eq:NN \columncolor \@@_columncolor
1218         \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`.

```

1219         \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1220         \seq_clear_new:N \l_@@_colors_seq
1221         \l_@@_code_before_tl
1222         \@@_actually_color:
1223         \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1224     \group_end:
1225 }
1226 }

```

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).

```

1227     \int_compare:nNnT \l_@@_last_row_int > { -2 }
1228     {
1229         \tl_put_right:Nn \@@_update_for_first_and_last_row:
1230         {
1231             \dim_gset:Nn \g_@@_ht_last_row_dim
1232             { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1233             \dim_gset:Nn \g_@@_dp_last_row_dim
1234             { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1235         }
1236     }
1237     \int_compare:nNnT \l_@@_last_row_int = { -1 }
1238     {
1239         \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.

```

1240     \str_if_empty:NTF \l_@@_name_str
1241     {
1242         \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1243         {
1244             \int_set:Nn \l_@@_last_row_int
1245             { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1246         }
1247     }
1248     {
1249         \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1250         {
1251             \int_set:Nn \l_@@_last_row_int
1252             { \use:c { @@_last_row_ \l_@@_name_str } }
1253         }
1254     }
1255 }

```

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).

```

1256     \int_compare:nNnT \l_@@_last_col_int = { -1 }
1257     {
1258         \str_if_empty:NTF \l_@@_name_str
1259         {
1260             \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1261             {
1262                 \int_set:Nn \l_@@_last_col_int
1263                 { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1264             }
1265         }
1266         {
1267             \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }

```

```

1268         {
1269             \int_set:Nn \l_@@_last_col_int
1270             { \use:c { @@_last_col_ \l_@@_name_str } }
1271         }
1272     }
1273 }

```

The code in `\@@_pre_array:` is used only by `{NiceArrayWithDelims}`.

```

1274 \@@_pre_array:

```

We compute the width of both delimiters.

```

1275 \dim_zero_new:N \l_@@_left_delim_dim
1276 \dim_zero_new:N \l_@@_right_delim_dim
1277 \bool_if:NTF \l_@@_NiceArray_bool
1278 {
1279     \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1280     \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1281 }
1282 {

```

The command `\bBigg@` is a command of `amsmath`.

```

1283 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \l_@@_left_delim_tl $ }
1284 \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1285 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \l_@@_right_delim_tl $ }
1286 \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1287 }

```

The array will be composed in a box (named `\l_@@_the_array_box`) because we have to do manipulations concerning the potential exterior rows.

```

1288 \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).

```

1289 \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:

```

The preamble will be constructed in `\g_@@_preamble_tl`.

```

1290 \@@_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).

```

1291 \hbox_set:Nw \l_@@_the_array_box
1292 \skip_horizontal:N \l_@@_left_margin_dim
1293 \skip_horizontal:N \l_@@_extra_left_margin_dim
1294 \c_math_toggle_token
1295 \bool_if:NTF \l_@@_light_syntax_bool
1296 { \use:c { @@-light-syntax } }
1297 { \use:c { @@-normal-syntax } }
1298 }
1299 {
1300     \bool_if:NTF \l_@@_light_syntax_bool
1301     { \use:c { end @@-light-syntax } }
1302     { \use:c { end @@-normal-syntax } }
1303     \c_math_toggle_token
1304     \skip_horizontal:N \l_@@_right_margin_dim
1305     \skip_horizontal:N \l_@@_extra_right_margin_dim
1306     \hbox_set_end:

```

End of the construction of the array (in the box `\l_@@_the_array_box`).



It 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).

```

1307 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1308 {
1309   \bool_if:NF \l_@@_last_row_without_value_bool
1310   {
1311     \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1312     {
1313       \@@_error:n { Wrong~last~row }
1314       \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1315     }
1316   }
1317 }

```

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>43</sup>

```

1318 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1319 \bool_if:nTF \g_@@_last_col_found_bool
1320 { \int_gdecr:N \c@jCol }
1321 {
1322   \int_compare:nNnT \l_@@_last_col_int > { -1 }
1323   { \@@_error:n { last~col~not~used } }
1324 }

```

We fix also the value of `\c@iRow` and `\g_@@_row_total_int` with the same principle.

```

1325 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1326 \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).

```

1327 \int_compare:nNnT \l_@@_first_col_int = 0
1328 {
1329   \skip_horizontal:N \col@sep
1330   \skip_horizontal:N \g_@@_width_first_col_dim
1331 }

```

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.

```

1332 \bool_if:NTF \l_@@_NiceArray_bool
1333 {
1334   \str_case:VnF \l_@@_baseline_tl
1335   {
1336     b \@@_use_arraybox_with_notes_b:
1337     c \@@_use_arraybox_with_notes_c:
1338   }
1339   \@@_use_arraybox_with_notes:
1340 }

```

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).

```

1341 {
1342   \int_compare:nNnTF \l_@@_first_row_int = 0
1343   {
1344     \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1345     \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1346   }
1347   { \dim_zero:N \l_tmpa_dim }

```

---

<sup>43</sup>We remind that the potential “first column” (exterior) has the number 0.

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>44</sup>

```

1348 \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1349 {
1350     \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1351     \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1352 }
1353 { \dim_zero:N \l_tmpb_dim }
1354 \hbox_set:Nn \l_tmpa_box
1355 {
1356     \c_math_toggle_token
1357     \tl_if_empty:NF \l_@@_delimiters_color_tl
1358     { \color { \l_@@_delimiters_color_tl } }
1359     \exp_after:wN \left \l_@@_left_delim_tl
1360     \vcenter
1361     {

```

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.

```

1362     \skip_vertical:N -\l_tmpa_dim
1363     \hbox
1364     {
1365         \bool_if:NTF \l_@@_NiceTabular_bool
1366         { \skip_horizontal:N -\tabcolsep }
1367         { \skip_horizontal:N -\arraycolsep }
1368         \@@_use_arraybox_with_notes_c:
1369         \bool_if:NTF \l_@@_NiceTabular_bool
1370         { \skip_horizontal:N -\tabcolsep }
1371         { \skip_horizontal:N -\arraycolsep }
1372     }

```

We take into account the “last row” (we have previously computed its total height in `\l_tmpb_dim`).

```

1373     \skip_vertical:N -\l_tmpb_dim
1374 }
1375 \exp_after:wN \right \l_@@_right_delim_tl
1376 \c_math_toggle_token
1377 }

```

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.

```

1378 \bool_if:NTF \l_@@_max_delimiter_width_bool
1379 { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1380 \@@_put_box_in_flow:
1381 }

```

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).

```

1382 \bool_if:NT \g_@@_last_col_found_bool
1383 {
1384     \skip_horizontal:N \g_@@_width_last_col_dim
1385     \skip_horizontal:N \col@sep
1386 }
1387 \bool_if:NF \l_@@_Matrix_bool
1388 {
1389     \int_compare:nNnT \c@jCol < \g_@@_static_num_of_col_int
1390     { \@@_error:n { columns-not-used } }
1391 }
1392 \group_begin:
1393 \globaldefs = 1
1394 \@@_msg_redirect_name:nn { columns-not-used } { error }
1395 \group_end:

```

---

<sup>44</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).

```
1396 \@@_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.

```
1397 \egroup
1398 \bool_if:NT \c_@@_footnote_bool \endsavenotes
1399 }
```

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.

```
1400 \cs_new_protected:Npn \@@_construct_preamble:
1401 {
```

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 programmation 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.

```
1402 \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).

```
1403 \bool_if:NF \l_@@_Matrix_bool
1404 {
1405   \@@_newcolumntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1406   \@@_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`).

```
1407 \exp_args:NV \@temptokena \g_@@_preamble_tl
```

Initialisation of a flag used by `array` to detect the end of the expansion.

```
1408 \@tempswatrue
```

The following line actually does the expansion (it’s has been copied from `array.sty`).

```
1409 \@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`.

```
1410 \int_gzero_new:N \c@jCol
1411 \bool_if:NTF \l_@@_vlines_bool
1412 {
1413   \tl_gset:Nn \g_@@_preamble_tl
1414     { ! { \skip_horizontal:N \arrayrulewidth } }
1415 }
1416 { \tl_gclear:N \g_@@_preamble_tl }
```

The counter `\l_tmpa_int` will count the number of consecutive occurrences of the symbol `|`.

```
1417 \int_zero:N \l_tmpa_int
```

Now, we actually patch the preamble (and it is constructed in `\g_@@_preamble_tl`).

```
1418 \exp_after:wN \@@_patch_preamble:n \the \temptokena \q_stop
1419 \int_gset_eq:NN \g_@@_static_num_of_col_int \c@jCol
1420 }
```

Now, we replace `\columncolor` by `\@@_columncolor_preamble`.

```
1421 \bool_if:NT \l_@@_colortbl_like_bool
1422 {
1423   \regex_replace_all:NnN
1424   \c_@@_columncolor_regex
1425   { \c { @@_columncolor_preamble } }
1426   \g_@@_preamble_tl
1427 }
```

We complete the preamble with the potential “exterior columns”.

```
1428 \int_compare:nNnTF \l_@@_first_col_int = 0
1429 { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1430 {
1431   \bool_lazy_all:nT
1432   {
1433     \l_@@_NiceArray_bool
1434     { \bool_not_p:n \l_@@_NiceTabular_bool }
1435     { \bool_not_p:n \l_@@_vlines_bool }
1436     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1437   }
1438   { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1439 }
1440 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1441 { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }
1442 {
1443   \bool_lazy_all:nT
1444   {
1445     \l_@@_NiceArray_bool
1446     { \bool_not_p:n \l_@@_NiceTabular_bool }
1447     { \bool_not_p:n \l_@@_vlines_bool }
1448     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1449   }
1450   { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1451 }
```

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`).

```
1452 \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1453 {
1454   \tl_gput_right:Nn
1455   \g_@@_preamble_tl
1456   { > { \@@_error_too_much_cols: } 1 }
1457 }
```

Now, we have to close the TeX group which was opened for the redefinition of the columns of type `w` and `W`.

```
1458 \group_end:
1459 }
```

```
1460 \cs_new_protected:Npn \@@_patch_preamble:n #1
1461 {
1462   \str_case:nnF { #1 }
1463   {
```

```

1464 c { \@@_patch_preamble_i:n #1 }
1465 l { \@@_patch_preamble_i:n #1 }
1466 r { \@@_patch_preamble_i:n #1 }
1467 > { \@@_patch_preamble_ii:nn #1 }
1468 ! { \@@_patch_preamble_ii:nn #1 }
1469 @ { \@@_patch_preamble_ii:nn #1 }
1470 | { \@@_patch_preamble_iii:n #1 }
1471 p { \@@_patch_preamble_iv:nnn t #1 }
1472 m { \@@_patch_preamble_iv:nnn c #1 }
1473 b { \@@_patch_preamble_iv:nnn b #1 }
1474 \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1475 \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1476 \@@_true_c: { \@@_patch_preamble_vi:n #1 }
1477 ( { \@@_patch_preamble_vii:n #1 }
1478 [ { \@@_patch_preamble_vii:n #1 }
1479 \{ { \@@_patch_preamble_vii:n #1 }
1480 ) { \@@_patch_preamble_viii:n #1 }
1481 ] { \@@_patch_preamble_viii:n #1 }
1482 \} { \@@_patch_preamble_viii:n #1 }
1483 C { \@@_error:nn { old~column~type } #1 }
1484 L { \@@_error:nn { old~column~type } #1 }
1485 R { \@@_error:nn { old~column~type } #1 }
1486 \q_stop { }
1487 }
1488 {
1489 \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1490 { \@@_patch_preamble_xi:n #1 }
1491 {
1492 \bool_lazy_and:nnTF
1493 { \str_if_eq_p:nn { : } { #1 } }
1494 \c_@@_arydshln_loaded_bool
1495 {
1496 \tl_gput_right:Nn \g_@@_preamble_tl { : }
1497 \@@_patch_preamble:n
1498 }
1499 { \@@_fatal:nn { unknown~column~type } { #1 } }
1500 }
1501 }
1502 }

```

For c, l and r

```

1503 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1504 {
1505 \tl_gput_right:Nn \g_@@_preamble_tl
1506 {
1507 > { \@@_Cell: \tl_set:Nn \l_@@_cell_type_tl { #1 } }
1508 #1
1509 < \@@_end_Cell:
1510 }

```

We increment the counter of columns and then we test for the presence of a <.

```

1511 \int_gincr:N \c@jCol
1512 \@@_patch_preamble_x:n
1513 }

```

For >, ! and @

```

1514 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1515 {
1516 \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
1517 \@@_patch_preamble:n
1518 }

```

For |

```

1519 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1520 {
\l_tmpa_int is the number of successive occurrences of |
1521 \int_incr:N \l_tmpa_int
1522 \@@_patch_preamble_iii_i:n
1523 }
1524 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1525 {
1526 \str_if_eq:nnTF { #1 } |
1527 { \@@_patch_preamble_iii:n | }
1528 {
1529 \tl_gput_right:Nx \g_@@_preamble_tl
1530 {
1531 \exp_not:N !
1532 {
1533 \skip_horizontal:n
1534 {
1535 \dim_eval:n
1536 {
1537 \arrayrulewidth * \l_tmpa_int
1538 + \doublerulesep * ( \l_tmpa_int - 1)
1539 }
1540 }
1541 }
1542 }
1543 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1544 { \@@_vline:nn { \@@_succ:n \c@jCol } { \int_use:N \l_tmpa_int } }
1545 \int_zero:N \l_tmpa_int
1546 \@@_patch_preamble:n #1
1547 }
1548 }

```

For p, m and b

```

1549 \cs_new_protected:Npn \@@_patch_preamble_iv:nnn #1 #2 #3
1550 {
1551 \tl_gput_right:Nn \g_@@_preamble_tl
1552 {
1553 > {
1554 \@@_Cell:
1555 \begin { minipage } [ #1 ] { #3 }
1556 \mode_leave_vertical:
1557 \arraybackslash % added in the version 5.8
1558 \box_use:N \@arstrutbox
1559 }
1560 c
1561 < { \box_use:N \@arstrutbox \end { minipage } \@@_end_Cell: }
1562 }

```

We increment the counter of columns, and then we test for the presence of a <.

```

1563 \int_gincr:N \c@jCol
1564 \@@_patch_preamble_x:n
1565 }

```

For w and W

```

1566 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1567 {
1568 \tl_gput_right:Nn \g_@@_preamble_tl
1569 {
1570 > {
1571 \hbox_set:Nw \l_@@_cell_box
1572 \@@_Cell:
1573 \tl_set:Nn \l_@@_cell_type_tl { #3 }
1574 }

```

```

1575         c
1576     < {
1577         \@@_end_Cell:
1578         #1
1579         \hbox_set_end:
1580         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1581         \@@_adjust_size_box:
1582         \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
1583     }
1584 }

```

We increment the counter of columns and then we test for the presence of a <.

```

1585     \int_gincr:N \c@jCol
1586     \@@_patch_preamble_x:n
1587 }

```

For \@@\_true\_c: which will appear in our redefinition of the columns of type S (of siunitx).

```

1588 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1589 {
1590     \tl_gput_right:Nn \g_@@_preamble_tl { c }

```

We increment the counter of columns and then we test for the presence of a <.

```

1591     \int_gincr:N \c@jCol
1592     \@@_patch_preamble_x:n
1593 }

```

For (, [ and \{.

```

1594 \cs_new_protected:Npn \@@_patch_preamble_vii:n #1
1595 {
1596     \bool_if:NT \l_@@_small_bool
1597     { \@@_fatal:n { Delimiter~with~small } }

```

If we are before the column 1, we reserve space for the left delimiter.

```

1598     \int_compare:nNnT \c@jCol = \c_zero_int
1599     { \tl_gput_right:Nx \g_@@_preamble_tl { ! { \enskip } } }
1600     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1601     { \@@_delimiter:nnn #1 { \@@_succ:n \c@jCol } \c_true_bool }
1602     \@@_patch_preamble:n
1603 }

```

For ), ] and \}.

```

1604 \cs_new_protected:Npn \@@_patch_preamble_viii:n #1
1605 {
1606     \bool_if:NT \l_@@_small_bool
1607     { \@@_fatal:n { Delimiter~with~small } }
1608     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1609     { \@@_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).

```

1610     \@@_patch_preamble_viii_i:n
1611 }

1612 \cs_new_protected:Npn \@@_patch_preamble_viii_i:n #1
1613 {
1614     \bool_lazy_any:nT
1615     {
1616         { \str_if_eq_p:nn { #1 } ( }
1617         { \str_if_eq_p:nn { #1 } [ }
1618         { \str_if_eq_p:nn { #1 } \{ }
1619         { \str_if_eq_p:nn { #1 } \q_stop }
1620     }
1621     { \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } } }
1622     \@@_patch_preamble:n #1
1623 }

```

```

1624 \cs_new_protected:Npn \@@_patch_preamble_xi:n #1
1625 {
1626   \tl_gput_right:Nn \g_@@_preamble_tl
1627   { ! { \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`.

```

1628   \tl_gput_right:Nx \g_@@_internal_code_after_tl
1629   { \@@_vdottedline:n { \int_use:N \c@jCol } }
1630   \@@_patch_preamble:n
1631 }

```

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.

```

1632 \cs_new_protected:Npn \@@_patch_preamble_x:n #1
1633 {
1634   \str_if_eq:nnTF { #1 } { < }
1635   \@@_patch_preamble_ix:n
1636   {
1637     \bool_if:NT \l_@@_vlines_bool
1638     {
1639       \tl_gput_right:Nn \g_@@_preamble_tl
1640       { ! { \skip_horizontal:N \arrayrulewidth } }
1641     }
1642     \@@_patch_preamble:n { #1 }
1643   }
1644 }
1645 \cs_new_protected:Npn \@@_patch_preamble_ix:n #1
1646 {
1647   \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
1648   \@@_patch_preamble_x:n
1649 }

```

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`).

```

1650 \cs_new_protected:Npn \@@_put_box_in_flow:
1651 {
1652   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1653   \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1654   \tl_if_eq:NnTF \l_@@_baseline_tl { c }
1655   { \box_use_drop:N \l_tmpa_box }
1656   \@@_put_box_in_flow_i:
1657 }

```

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).

```

1658 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1659 {
1660   \pgfpicture
1661   \@@_qpoint:n { row - 1 }
1662   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1663   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1664   \dim_gadd:Nn \g_tmpa_dim \pgf@y
1665   \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).

```

1666   \str_if_in:NnTF \l_@@_baseline_tl { line- }
1667   {
1668     \int_set:Nn \l_tmpa_int
1669     {

```



```

1670         \str_range:Nnn
1671         \l_@@_baseline_tl
1672         6
1673         { \tl_count:V \l_@@_baseline_tl }
1674     }
1675     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1676 }
1677 {
1678     \str_case:VnF \l_@@_baseline_tl
1679     {
1680         { t } { \int_set:Nn \l_tmpa_int 1 }
1681         { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1682     }
1683     { \int_set:Nn \l_tmpa_int \l_@@_baseline_tl }
1684     \bool_lazy_or:nnT
1685     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1686     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1687     {
1688         \@@_error:n { bad~value~for~baseline }
1689         \int_set:Nn \l_tmpa_int 1
1690     }
1691     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

We take into account the position of the mathematical axis.

```

1692     \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
1693 }
1694 \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

Now, `\g_tmpa_dim` contains the value of the  $y$  translation we have to to.

```

1695     \endpgfpicture
1696     \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1697     \box_use_drop:N \l_tmpa_box
1698 }

```

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).

```

1699 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
1700 {

```

We need a `{minipage}` because we will insert a LaTeX list for the tabular notes (that means that a `\vtop{\hspace=...}` is not enough).

```

1701     \begin { minipage } [ t ] { \box_wd:N \l_@@_the_array_box }
1702     \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.

```

1703     \@@_create_extra_nodes:
1704     \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
1705     \bool_lazy_or:nnT
1706     { \int_compare_p:nNn \c@tabularnote > 0 }
1707     { ! \tl_if_empty_p:V \l_@@_tabularnote_tl }
1708     \@@_insert_tabularnotes:
1709     \end { minipage }
1710 }

```

```

1711 \cs_new_protected:Npn \@@_insert_tabularnotes:
1712 {
1713     \skip_vertical:N 0.65ex

```

The TeX group is for potential specifications in the `\l_@@_notes_code_before_tl`.

```

1714     \group_begin:
1715     \l_@@_notes_code_before_tl
1716     \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.

```

1717 \int_compare:nNnT \c@tabularnote > 0
1718 {
1719   \bool_if:NTF \l_@@_notes_para_bool
1720   {
1721     \begin { tabularnotes* }
1722     \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1723     \end { tabularnotes* }

```

The following `\par` is mandatory for the event that the user has put `\footnotesize` (for example) in the notes/code-before.

```

1724   \par
1725 }
1726 {
1727   \tabularnotes
1728   \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1729   \endtabularnotes
1730 }
1731 }
1732 \unskip
1733 \group_end:
1734 \bool_if:NT \l_@@_notes_bottomrule_bool
1735 {
1736   \bool_if:NTF \c_@@_booktabs_loaded_bool
1737   {

```

The two dimensions `\aboverulesep` et `\heavyrulewidth` are parameters defined by `booktabs`.

```

1738   \skip_vertical:N \aboverulesep
\CT@arc@ is the specification of color defined by colortbl but you use it even if colortbl is not loaded.
1739   { \CT@arc@ \hrule height \heavyrulewidth }
1740   }
1741   { @@_error:n { bottomrule-without-booktabs } }
1742   }
1743   \l_@@_notes_code_after_tl
1744   \seq_gclear:N \g_@@_tabularnotes_seq
1745   \int_gzero:N \c@tabularnote
1746 }

```

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`.

```

1747 \cs_new_protected:Npn @@_use_arraybox_with_notes_b:
1748 {
1749   \pgfpicture
1750   @@_qpoint:n { row - 1 }
1751   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1752   @@_qpoint:n { row - \int_use:N \c@iRow - base }
1753   \dim_gsub:Nn \g_tmpa_dim \pgf@y
1754   \endpgfpicture
1755   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1756   \int_compare:nNnT \l_@@_first_row_int = 0
1757   {
1758     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1759     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1760   }
1761   \box_move_up:nn \g_tmpa_dim { \hbox { @@_use_arraybox_with_notes_c: } }
1762 }

```

Now, the general case.

```

1763 \cs_new_protected:Npn @@_use_arraybox_with_notes:
1764 {

```

We convert a value of `t` to a value of 1.

```

1765 \tl_if_eq:NnT \l_@@_baseline_tl { t }
1766 { \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`.

```

1767 \pgfpicture
1768 \@@_qpoint:n { row - 1 }
1769 \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1770 \str_if_in:NnTF \l_@@_baseline_tl { line- }
1771 {
1772   \int_set:Nn \l_tmpa_int
1773   {
1774     \str_range:Nnn
1775     \l_@@_baseline_tl
1776     6
1777     { \tl_count:V \l_@@_baseline_tl }
1778   }
1779   \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1780 }
1781 {
1782   \int_set:Nn \l_tmpa_int \l_@@_baseline_tl
1783   \bool_lazy_or:nnT
1784   { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1785   { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1786   {
1787     \@@_error:n { bad~value~for~baseline }
1788     \int_set:Nn \l_tmpa_int 1
1789   }
1790   \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1791 }
1792 \dim_gsub:Nn \g_tmpa_dim \pgf@y
1793 \endpgfpicture
1794 \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1795 \int_compare:nNnT \l_@@_first_row_int = 0
1796 {
1797   \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1798   \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1799 }
1800 \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
1801 }

```

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.

```

1802 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1803 {

```

We will compute the real width of both delimiters used.

```

1804 \dim_zero_new:N \l_@@_real_left_delim_dim
1805 \dim_zero_new:N \l_@@_real_right_delim_dim
1806 \hbox_set:Nn \l_tmpb_box
1807 {
1808   \c_math_toggle_token
1809   \left #1
1810   \vcenter
1811   {
1812     \vbox_to_ht:nn
1813     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1814     { }
1815   }
1816   \right .
1817   \c_math_toggle_token

```

```

1818     }
1819     \dim_set:Nn \l_@@_real_left_delim_dim
1820     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
1821     \hbox_set:Nn \l_tmpb_box
1822     {
1823       \c_math_toggle_token
1824       \left .
1825       \vbox_to_ht:nn
1826       { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1827       { }
1828       \right #2
1829       \c_math_toggle_token
1830     }
1831     \dim_set:Nn \l_@@_real_right_delim_dim
1832     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }

```

Now, we can put the box in the TeX flow with the horizontal adjustments on both sides.

```

1833     \skip_horizontal:N \l_@@_left_delim_dim
1834     \skip_horizontal:N -\l_@@_real_left_delim_dim
1835     \@@_put_box_in_flow:
1836     \skip_horizontal:N \l_@@_right_delim_dim
1837     \skip_horizontal:N -\l_@@_real_right_delim_dim
1838   }

```

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).

```

1839 \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).

```

1840   {
1841     \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`).

```

1842     { \exp_args:NV \@@_array: \g_@@_preamble_tl }
1843   }
1844   {
1845     \@@_create_col_nodes:
1846     \endarray
1847   }

```

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`).

```

1848 \NewDocumentEnvironment { @@-light-syntax } { b }
1849   {

```

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.

```

1850     \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1851     \tl_map_inline:nn { #1 }
1852     {
1853       \str_if_eq:nnT { ##1 } { & }
1854       { \@@_fatal:n { ampersand-in-light-syntax } }
1855       \str_if_eq:nnT { ##1 } { \ }
1856       { \@@_fatal:n { double-backslash-in-light-syntax } }
1857     }

```

Now, you extract the `code-after` 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`.

```
1858   \@@_light_syntax_i #1 \CodeAfter \q_stop
1859 }
```

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`.

```
1860 { }
1861 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1862 {
1863   \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).

```
1864   \seq_gclear_new:N \g_@@_rows_seq
1865   \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1866   \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.

```
1867   \int_compare:nNnT \l_@@_last_row_int = { -1 }
1868     { \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`).

```
1869   \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.

```
1870   \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1871   \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1872   \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1873   \@@_create_col_nodes:
1874   \endarray
1875 }
1876 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1877 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }
1878 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1879 {
1880   \seq_gclear_new:N \g_@@_cells_seq
1881   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1882   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1883   \l_tmpa_tl
1884   \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1885 }
```

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).

```
1886 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1887 {
1888   \str_if_eq:VnT \g_@@_name_env_str { #2 }
1889     { \@@_fatal:n { empty-environment } } }
```

We repute in the stream the `\end{...}` we have extracted and the user will have an error for incorrect nested environments.

```
1890   \end { #2 }
1891 }
```

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).

```

1892 \cs_new:Npn \@@_create_col_nodes:
1893 {
1894   \crcr
1895   \int_compare:nNnT \l_@@_first_col_int = 0
1896   {
1897     \omit
1898     \hbox_overlap_left:n
1899     {
1900       \bool_if:NT \l_@@_code_before_bool
1901       { \pgfsys@markposition { \@@_env: - col - 0 } }
1902       \pgfpicture
1903       \pgfrememberpicturerepositiononpagetrue
1904       \pgfcoordinate { \@@_env: - col - 0 } \pgfpintorigin
1905       \str_if_empty:NF \l_@@_name_str
1906       { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1907       \endpgfpicture
1908       \skip_horizontal:N 2\col@sep
1909       \skip_horizontal:N \g_@@_width_first_col_dim
1910     }
1911     &
1912   }
1913   \omit

```

The following instruction must be put after the instruction `\omit`.

```

1914   \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`).

```

1915   \int_compare:nNnTF \l_@@_first_col_int = 0
1916   {
1917     \bool_if:NT \l_@@_code_before_bool
1918     {
1919       \hbox
1920       {
1921         \skip_horizontal:N -0.5\arrayrulewidth
1922         \pgfsys@markposition { \@@_env: - col - 1 }
1923         \skip_horizontal:N 0.5\arrayrulewidth
1924       }
1925     }
1926     \pgfpicture
1927     \pgfrememberpicturerepositiononpagetrue
1928     \pgfcoordinate { \@@_env: - col - 1 }
1929     { \pgfpint { - 0.5 \arrayrulewidth } \c_zero_dim }
1930     \str_if_empty:NF \l_@@_name_str
1931     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1932     \endpgfpicture
1933   }
1934   {
1935     \bool_if:NT \l_@@_code_before_bool
1936     {
1937       \hbox
1938       {
1939         \skip_horizontal:N 0.5\arrayrulewidth
1940         \pgfsys@markposition { \@@_env: - col - 1 }
1941         \skip_horizontal:N -0.5\arrayrulewidth
1942       }
1943     }
1944     \pgfpicture
1945     \pgfrememberpicturerepositiononpagetrue
1946     \pgfcoordinate { \@@_env: - col - 1 }
1947     { \pgfpint { 0.5 \arrayrulewidth } \c_zero_dim }

```

```

1948 \str_if_empty:NF \l_@@_name_str
1949 { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1950 \endpgfpicture
1951 }

```

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.

```

1952 \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1953 \bool_if:NF \l_@@_auto_columns_width_bool
1954 { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
1955 {
1956   \bool_lazy_and:nnTF
1957     \l_@@_auto_columns_width_bool
1958     { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1959     { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1960     { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1961   \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
1962 }
1963 \skip_horizontal:N \g_tmpa_skip
1964 \hbox
1965 {
1966   \bool_if:NT \l_@@_code_before_bool
1967   {
1968     \hbox
1969     {
1970       \skip_horizontal:N -0.5\arrayrulewidth
1971       \pgfsys@markposition { \@@_env: - col - 2 }
1972       \skip_horizontal:N 0.5\arrayrulewidth
1973     }
1974   }
1975   \pgfpicture
1976   \pgfrememberpicturepositiononpagetrue
1977   \pgfcoordinate { \@@_env: - col - 2 }
1978   { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1979   \str_if_empty:NF \l_@@_name_str
1980   { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
1981   \endpgfpicture
1982 }

```

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.

```

1983 \int_gset:Nn \g_tmpa_int 1
1984 \bool_if:NTF \g_@@_last_col_found_bool
1985 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1986 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1987 {
1988   &
1989   \omit
1990   \int_gincr:N \g_tmpa_int

```

The incrementation of the counter `\g_tmpa_int` must be done after the `\omit` of the cell.

```

1991 \skip_horizontal:N \g_tmpa_skip
1992 \bool_if:NT \l_@@_code_before_bool
1993 {
1994   \hbox
1995   {
1996     \skip_horizontal:N -0.5\arrayrulewidth
1997     \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1998     \skip_horizontal:N 0.5\arrayrulewidth
1999   }

```

2000            }

We create the col node on the right of the current column.

```

2001       \pgfpicture
2002       \pgfrememberpicturepositiononpagetrue
2003       \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2004       { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2005       \str_if_empty:NF \l_@@_name_str
2006       {
2007        \pgfnodealias
2008        { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
2009        { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2010       }
2011       \endpgfpicture
2012     }
2013   \bool_if:NT \g_@@_last_col_found_bool
2014   {
2015     \hbox_overlap_right:n
2016     {
2017       % \skip_horizontal:N \col@sep
2018       \skip_horizontal:N \g_@@_width_last_col_dim
2019       \bool_if:NT \l_@@_code_before_bool
2020       {
2021         \pgfsys@markposition
2022         { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2023       }
2024       \pgfpicture
2025       \pgfrememberpicturepositiononpagetrue
2026       \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2027       \pgfpointorigin
2028       \str_if_empty:NF \l_@@_name_str
2029       {
2030         \pgfnodealias
2031         { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
2032         { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2033       }
2034       \endpgfpicture
2035     }
2036   }
2037   \cr
2038 }

```

Here is the preamble for the “first column” (if the user uses the key `first-col`)

```

2039 \tl_const:Nn \c_@@_preamble_first_col_tl
2040 {
2041   >
2042   {
2043     \bool_gset_true:N \g_@@_after_col_zero_bool
2044     \@@_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.

```

2045       \hbox_set:Nw \l_@@_cell_box
2046       \@@_math_toggle_token:
2047       \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”.

```

2048       \bool_lazy_and:nnT
2049       { \int_compare_p:nNn \c@iRow > 0 }
2050       {
2051         \bool_lazy_or_p:nn
2052         { \int_compare_p:nNn \l_@@_last_row_int < 0 }

```



```

2053         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2054     }
2055     {
2056         \l_@@_code_for_first_col_tl
2057         \xglobal \colorlet { nicematrix-first-col } { . }
2058     }
2059 }

```

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`.

```

2060     l
2061     <
2062     {
2063         \@@_math_toggle_token:
2064         \hbox_set_end:
2065         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2066         \@@_adjust_size_box:
2067         \@@_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.

```

2068         \dim_gset:Nn \g_@@_width_first_col_dim
2069         { \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.

```

2070         \hbox_overlap_left:n
2071         {
2072             \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2073             \@@_node_for_the_cell:
2074             { \box_use_drop:N \l_@@_cell_box }
2075             \skip_horizontal:N \l_@@_left_delim_dim
2076             \skip_horizontal:N \l_@@_left_margin_dim
2077             \skip_horizontal:N \l_@@_extra_left_margin_dim
2078         }
2079         \bool_gset_false:N \g_@@_empty_cell_bool
2080         \skip_horizontal:N -2\col@sep
2081     }
2082 }

```

Here is the preamble for the “last column” (if the user uses the key `last-col`).

```

2083 \tl_const:Nn \c_@@_preamble_last_col_tl
2084 {
2085     >
2086     {

```

With the flag `\g_@@_last_col_found_bool`, we will know that the “last column” is really used.

```

2087         \bool_gset_true:N \g_@@_last_col_found_bool
2088         \int_gincr:N \c@jCol
2089         \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.

```

2090         \hbox_set:Nw \l_@@_cell_box
2091         \@@_math_toggle_token:
2092         \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”.

```

2093         \int_compare:nNnT \c@iRow > 0
2094         {
2095             \bool_lazy_or:nnT
2096             { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2097             { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2098             {
2099                 \l_@@_code_for_last_col_tl
2100                 \xglobal \colorlet { nicematrix-last-col } { . }

```

```

2101     }
2102   }
2103 }
2104 l
2105 <
2106 {
2107   \@@_math_toggle_token:
2108   \hbox_set_end:
2109   \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2110   \@@_adjust_size_box:
2111   \@@_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.

```

2112   \dim_gset:Nn \g_@@_width_last_col_dim
2113   { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
2114   \skip_horizontal:N -2\col@sep

```

The content of the cell is inserted in an overlapping position.

```

2115   \hbox_overlap_right:n
2116   {
2117     \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2118     {
2119       \skip_horizontal:N \l_@@_right_delim_dim
2120       \skip_horizontal:N \l_@@_right_margin_dim
2121       \skip_horizontal:N \l_@@_extra_right_margin_dim
2122       \@@_node_for_the_cell:
2123     }
2124   }
2125   \bool_gset_false:N \g_@@_empty_cell_bool
2126 }
2127 }

```

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.

```

2128 \NewDocumentEnvironment { NiceArray } { }
2129 {
2130   \bool_set_true:N \l_@@_NiceArray_bool
2131   \str_if_empty:NT \g_@@_name_env_str
2132   { \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).

```

2133   \NiceArrayWithDelims . .
2134 }
2135 { \endNiceArrayWithDelims }

```

We create the variants of the environment `{NiceArrayWithDelims}`.

```

2136 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
2137 {
2138   \NewDocumentEnvironment { #1 NiceArray } { }
2139   {
2140     \str_if_empty:NT \g_@@_name_env_str
2141     { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
2142     \@@_test_if_math_mode:
2143     \NiceArrayWithDelims #2 #3
2144   }
2145   { \endNiceArrayWithDelims }
2146 }

```

```

2147 \@@_def_env:nnn p ( )
2148 \@@_def_env:nnn b [ ]
2149 \@@_def_env:nnn B \{ \}
2150 \@@_def_env:nnn v | |
2151 \@@_def_env:nnn V \| \|

```

## The environment {NiceMatrix} and its variants

```

2152 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
2153 {
2154   \bool_set_true:N \l_@@_Matrix_bool
2155   \use:c { #1 NiceArray }
2156   {
2157     *
2158     {
2159       \int_compare:nNnTF \l_@@_last_col_int < 0
2160         \c@MaxMatrixCols
2161         { \@@_pred:n \l_@@_last_col_int }
2162     }
2163     { > \@@_Cell: #2 < \@@_end_Cell: }
2164   }
2165 }
2166 \clist_map_inline:nn { { } , p , b , B , v , V }
2167 {
2168   \NewDocumentEnvironment { #1 NiceMatrix } { ! O { } }
2169   {
2170     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
2171     \tl_set:Nn \l_@@_type_of_col_tl c
2172     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
2173     \exp_args:Nne \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
2174   }
2175   { \use:c { end #1 NiceArray } }
2176 }

```

The following command will be linked to \NotEmpty in the environments of nicematrix.

```

2177 \cs_new_protected:Npn \@@_NotEmpty:
2178 { \bool_gset_true:N \g_@@_not_empty_cell_bool }

```

## The environments {NiceTabular} and {NiceTabular\*}

```

2179 \NewDocumentEnvironment { NiceTabular } { O { } m ! O { } }
2180 {
2181   \str_gset:Nn \g_@@_name_env_str { NiceTabular }
2182   \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
2183   \bool_set_true:N \l_@@_NiceTabular_bool
2184   \NiceArray { #2 }
2185 }
2186 { \endNiceArray }

2187 \NewDocumentEnvironment { NiceTabular* } { m O { } m ! O { } }
2188 {
2189   \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
2190   \dim_set:Nn \l_@@_tabular_width_dim { #1 }
2191   \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2192   \bool_set_true:N \l_@@_NiceTabular_bool
2193   \NiceArray { #3 }
2194 }
2195 { \endNiceArray }

```

## After the construction of the array

```

2196 \cs_new_protected:Npn \@@_after_array:
2197 {
2198   \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.

```

2199   \bool_if:NT \g_@@_last_col_found_bool
2200   { \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`.

```

2201   \bool_if:NT \l_@@_last_col_without_value_bool
2202   {
2203     \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
2204     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2205     \iow_shipout:Nx \@mainaux
2206     {
2207       \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
2208       { \int_use:N \g_@@_col_total_int }
2209     }
2210     \str_if_empty:NF \l_@@_name_str
2211     {
2212       \iow_shipout:Nx \@mainaux
2213       {
2214         \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
2215         { \int_use:N \g_@@_col_total_int }
2216       }
2217     }
2218     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2219   }

```

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.

```

2220   \bool_if:NT \l_@@_last_row_without_value_bool
2221   {
2222     \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.

```

2223   \bool_if:NF \l_@@_light_syntax_bool
2224   {
2225     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2226     \iow_shipout:Nx \@mainaux
2227     {
2228       \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
2229       { \int_use:N \g_@@_row_total_int }
2230     }

```

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.

```

2231     \str_if_empty:NF \l_@@_name_str
2232     {
2233       \iow_shipout:Nx \@mainaux
2234       {
2235         \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
2236         { \int_use:N \g_@@_row_total_int }
2237       }
2238     }
2239     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2240   }

```

```
2241 }
```

If the key `code-before` is used, we have to write on the `aux` file the actual size of the array.

```
2242 \bool_if:NT \l_@@_code_before_bool
2243 {
2244   \iow_now:Nn \@mainaux \ExplSyntaxOn
2245   \iow_now:Nx \@mainaux
2246   { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
2247   \iow_now:Nx \@mainaux
2248   {
2249     \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
2250     {
2251       \int_use:N \l_@@_first_row_int ,
2252       \int_use:N \g_@@_row_total_int ,
2253       \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`.

```
2254       \bool_lazy_and:nnTF
2255       { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
2256       { \bool_not_p:n \g_@@_last_col_found_bool }
2257       \@@_succ:n
2258       \int_use:N
2259       \g_@@_col_total_int
2260     }
```

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`).

```
2261       \seq_gset_from_clist:cn
2262       { c_@@_pos_of_blocks _ \int_use:N \g_@@_env_int _ seq }
2263       { \seq_use:Nnnn \g_@@_pos_of_blocks_seq , , , }
2264     }
2265   \iow_now:Nn \@mainaux \ExplSyntaxOff
2266 }
```

By default, the diagonal lines will be parallelized<sup>45</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.

```
2267 \bool_if:NT \l_@@_parallelize_diags_bool
2268 {
2269   \int_gzero_new:N \g_@@_ddots_int
2270   \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` 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.

```
2271   \dim_gzero_new:N \g_@@_delta_x_one_dim
2272   \dim_gzero_new:N \g_@@_delta_y_one_dim
2273   \dim_gzero_new:N \g_@@_delta_x_two_dim
2274   \dim_gzero_new:N \g_@@_delta_y_two_dim
2275 }
2276 \int_zero_new:N \l_@@_initial_i_int
2277 \int_zero_new:N \l_@@_initial_j_int
2278 \int_zero_new:N \l_@@_final_i_int
2279 \int_zero_new:N \l_@@_final_j_int
2280 \bool_set_false:N \l_@@_initial_open_bool
2281 \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.

```
2282 \bool_if:NT \l_@@_small_bool
```

<sup>45</sup>It's possible to use the option `parallelize-diags` to disable this parallelization.

```

2283 {
2284   \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2285   \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.

```

2286   \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2287 }

```

Now, we actually draw the dotted lines (specified by `\Cdots`, `\Vdots`, etc.).

```

2288   \@@_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.

```

2289   \@@_compute_corners:

```

The sequence `\g_@@_pos_of_blocks_seq` must be “adjusted” (for the case where the user have written something like `\Block{1-}`).

```

2290   \@@_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`.

```

2291   \bool_lazy_all:nT
2292   {
2293     { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
2294     { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
2295     { \seq_if_empty_p:N \l_@@_empty_corner_cells_seq }
2296   }
2297   {
2298     \cs_set_eq:NN \@@_vline_i:nn \@@_vline_i_complete:nn
2299     \cs_set_eq:NN \@@_hline_i:nn \@@_hline_i_complete:nn
2300   }
2301   \bool_if:NT \l_@@_hlines_bool \@@_draw_hlines:
2302   \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
2303   \g_@@_internal_code_after_tl
2304   \tl_gclear:N \g_@@_internal_code_after_tl

```

Now, the `code-after`.

```

2305   \bool_if:NT \c_@@_tikz_loaded_bool
2306   {
2307     \tikzset
2308     {
2309       every~picture / .style =
2310       {
2311         overlay ,
2312         remember~picture ,
2313         name~prefix = \@@_env: -
2314       }
2315     }
2316   }
2317   \cs_set_eq:NN \line \@@_line
2318   \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.

```

2319   \cs_set_eq:NN \CodeAfter \prg_do_nothing:

```

We clear the list of the names of the potential `\SubMatrix` that will appear in the `code-after` (unfortunately, that list has to be global).

```
2320 \seq_gclear:N \g_@@_submatrix_names_seq
```

And here's the code-after. Since the code-after may begin with an “argument” between square brackets of the options, we extract and treat that potential “argument” with the command `\@@_CodeAfter_keys:`.

```
2321 \exp_last_unbraced:N \@@_CodeAfter_keys: \g_nicematrix_code_after_tl
2322 \scan_stop:
2323 \tl_gclear:N \g_nicematrix_code_after_tl
2324 \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.

```
2325 \tl_if_empty:N \g_nicematrix_code_before_tl
2326 {
```

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`.

```
2327 \cs_set_protected:Npn \rectanglecolor { }
2328 \cs_set_protected:Npn \columncolor { }
2329 \iow_now:Nn \@mainaux \ExplSyntaxOn
2330 \iow_now:Nx \@mainaux
2331 {
2332   \tl_gset:cn
2333     { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
2334     { \exp_not:V \g_nicematrix_code_before_tl }
2335 }
2336 \iow_now:Nn \@mainaux \ExplSyntaxOff
2337 \bool_set_true:N \l_@@_code_before_bool
2338 }
```

```
2339 \str_gclear:N \g_@@_name_env_str
2340 \@@_restore_iRow_jCol:
```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>46</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`.

```
2341 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
2342 }
```

The following command will extract the potential options (between square brackets) at the beginning of the code-after (that is to say, when `\CodeAfter` is used, the options of that “command” `\CodeAfter`).

```
2343 \NewDocumentCommand \@@_CodeAfter_keys: { 0 { } }
2344 { \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).

```
2345 \cs_new_protected:Npn \@@_adjust_pos_of_blocks_seq:
2346 {
2347   \seq_gset_map_x:NNn \g_@@_pos_of_blocks_seq \g_@@_pos_of_blocks_seq
2348     { \@@_adjust_pos_of_blocks_seq_i:nnnn ##1 }
2349 }
```

---

<sup>46</sup>e.g. `\color[rgb]{0.5,0.5,0}`

The following command must *not* be protected.

```

2350 \cs_new:Npn \@@_adjust_pos_of_blocks_seq_i:nnnn #1 #2 #3 #4
2351 {
2352   { #1 }
2353   { #2 }
2354   {
2355     \int_compare:nNnTF { #3 } > { 99 }
2356     { \int_use:N \c@iRow }
2357     { #3 }
2358   }
2359   {
2360     \int_compare:nNnTF { #4 } > { 99 }
2361     { \int_use:N \c@jCol }
2362     { #4 }
2363   }
2364 }

```

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).

```

2365 \AtBeginDocument
2366 {
2367   \cs_new_protected:Npx \@@_draw_dotted_lines:
2368   {
2369     \c_@@_pgfortikzpicture_tl
2370     \@@_draw_dotted_lines_i:
2371     \c_@@_endpgfortikzpicture_tl
2372   }
2373 }

```

The following command *must* be protected because it will appear in the construction of the command `\@@_draw_dotted_lines:`.

```

2374 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2375 {
2376   \pgfrememberpicturepositiononpagetrue
2377   \pgf@relevantforpicturesizefalse
2378   \g_@@_HVdotsfor_lines_tl
2379   \g_@@_Vdots_lines_tl
2380   \g_@@_Ddots_lines_tl
2381   \g_@@_Iddots_lines_tl
2382   \g_@@_Cdots_lines_tl
2383   \g_@@_Ldots_lines_tl
2384 }

2385 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2386 {
2387   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2388   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2389 }

```

## 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.

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

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.

```
2390 \cs_new_protected:Npn \l_@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2391 {
```

First, we declare the current cell as “dotted” because we forbid intersections of dotted lines.

```
2392 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
2393 \int_set:Nn \l_@@_initial_i_int { #1 }
2394 \int_set:Nn \l_@@_initial_j_int { #2 }
2395 \int_set:Nn \l_@@_final_i_int { #1 }
2396 \int_set:Nn \l_@@_final_j_int { #2 }
```

We will do two loops: one when determining the initial cell and the other when determining 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.

```
2397 \bool_set_false:N \l_@@_stop_loop_bool
2398 \bool_do_until:Nn \l_@@_stop_loop_bool
2399 {
2400   \int_add:Nn \l_@@_final_i_int { #3 }
2401   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
2402   \bool_set_false:N \l_@@_final_open_bool
2403   \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
2404   {
2405     \int_compare:nNnTF { #3 } = 1
2406     { \bool_set_true:N \l_@@_final_open_bool }
2407     {
2408       \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2409       { \bool_set_true:N \l_@@_final_open_bool }
2410     }
2411   }
2412   {
2413     \int_compare:nNnTF \l_@@_final_j_int < 1
2414     {
2415       \int_compare:nNnT { #4 } = { -1 }
2416       { \bool_set_true:N \l_@@_final_open_bool }
2417     }
2418     {
2419       \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2420       {
2421         \int_compare:nNnT { #4 } = 1
2422         { \bool_set_true:N \l_@@_final_open_bool }
2423       }
2424     }
2425   }
2426   \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.

```

2427     {
We do a step backwards.
2428         \int_sub:Nn \l_@@_final_i_int { #3 }
2429         \int_sub:Nn \l_@@_final_j_int { #4 }
2430         \bool_set_true:N \l_@@_stop_loop_bool
2431     }

```

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`.

```

2432     {
2433         \cs_if_exist:cTF
2434         {
2435             @@ _ dotted _
2436             \int_use:N \l_@@_final_i_int -
2437             \int_use:N \l_@@_final_j_int
2438         }
2439         {
2440             \int_sub:Nn \l_@@_final_i_int { #3 }
2441             \int_sub:Nn \l_@@_final_j_int { #4 }
2442             \bool_set_true:N \l_@@_final_open_bool
2443             \bool_set_true:N \l_@@_stop_loop_bool
2444         }
2445         {
2446             \cs_if_exist:cTF
2447             {
2448                 pgf @ sh @ ns @ \@@_env:
2449                 - \int_use:N \l_@@_final_i_int
2450                 - \int_use:N \l_@@_final_j_int
2451             }
2452             { \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.

```

2453         {
2454             \cs_set:cpn
2455             {
2456                 @@ _ dotted _
2457                 \int_use:N \l_@@_final_i_int -
2458                 \int_use:N \l_@@_final_j_int
2459             }
2460             { }
2461         }
2462     }
2463 }
2464

```

For `\l_@@_initial_i_int` and `\l_@@_initial_j_int` the programmation is similar to the previous one.

```

2465     \bool_set_false:N \l_@@_stop_loop_bool
2466     \bool_do_until:Nn \l_@@_stop_loop_bool
2467     {
2468         \int_sub:Nn \l_@@_initial_i_int { #3 }
2469         \int_sub:Nn \l_@@_initial_j_int { #4 }
2470         \bool_set_false:N \l_@@_initial_open_bool
2471         \int_compare:nNnTF \l_@@_initial_i_int < 1
2472         {
2473             \int_compare:nNnTF { #3 } = 1

```

```

2474         { \bool_set_true:N \l_@@_initial_open_bool }
2475     {
2476         \int_compare:nNnT \l_@@_initial_j_int = 0
2477         { \bool_set_true:N \l_@@_initial_open_bool }
2478     }
2479 }
2480 {
2481     \int_compare:nNnTF \l_@@_initial_j_int < 1
2482     {
2483         \int_compare:nNnT { #4 } = 1
2484         { \bool_set_true:N \l_@@_initial_open_bool }
2485     }
2486     {
2487         \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
2488         {
2489             \int_compare:nNnT { #4 } = { -1 }
2490             { \bool_set_true:N \l_@@_initial_open_bool }
2491         }
2492     }
2493 }
2494 \bool_if:NNTF \l_@@_initial_open_bool
2495 {
2496     \int_add:Nn \l_@@_initial_i_int { #3 }
2497     \int_add:Nn \l_@@_initial_j_int { #4 }
2498     \bool_set_true:N \l_@@_stop_loop_bool
2499 }
2500 {
2501     \cs_if_exist:cTF
2502     {
2503         @@ _ dotted _
2504         \int_use:N \l_@@_initial_i_int -
2505         \int_use:N \l_@@_initial_j_int
2506     }
2507     {
2508         \int_add:Nn \l_@@_initial_i_int { #3 }
2509         \int_add:Nn \l_@@_initial_j_int { #4 }
2510         \bool_set_true:N \l_@@_initial_open_bool
2511         \bool_set_true:N \l_@@_stop_loop_bool
2512     }
2513     {
2514         \cs_if_exist:cTF
2515         {
2516             pgf @ sh @ ns @ \@@_env:
2517             - \int_use:N \l_@@_initial_i_int
2518             - \int_use:N \l_@@_initial_j_int
2519         }
2520         { \bool_set_true:N \l_@@_stop_loop_bool }
2521         {
2522             \cs_set:cpn
2523             {
2524                 @@ _ dotted _
2525                 \int_use:N \l_@@_initial_i_int -
2526                 \int_use:N \l_@@_initial_j_int
2527             }
2528             { }
2529         }
2530     }
2531 }
2532 }

```

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.

```

2533 \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2534 {

```

```

2535     { \int_use:N \l_@@_initial_i_int }
2536     { \int_use:N \l_@@_initial_j_int }
2537     { \int_use:N \l_@@_final_i_int }
2538     { \int_use:N \l_@@_final_j_int }
2539   }
2540 }

2541 \cs_new_protected:Npn \@@_set_initial_coords:
2542 {
2543   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2544   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2545 }
2546 \cs_new_protected:Npn \@@_set_final_coords:
2547 {
2548   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2549   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2550 }
2551 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2552 {
2553   \pgfpointanchor
2554   {
2555     \@@_env:
2556     - \int_use:N \l_@@_initial_i_int
2557     - \int_use:N \l_@@_initial_j_int
2558   }
2559   { #1 }
2560   \@@_set_initial_coords:
2561 }
2562 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2563 {
2564   \pgfpointanchor
2565   {
2566     \@@_env:
2567     - \int_use:N \l_@@_final_i_int
2568     - \int_use:N \l_@@_final_j_int
2569   }
2570   { #1 }
2571   \@@_set_final_coords:
2572 }

```

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.

```

2573 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
2574 {
2575   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2576   {
2577     \@@_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.

```

2578   \group_begin:
2579     \int_compare:nNnTF { #1 } = 0
2580     { \color { nicematrix-first-row } }
2581     {

```

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.

```

2582       \int_compare:nNnT { #1 } = \l_@@_last_row_int
2583       { \color { nicematrix-last-row } }
2584     }
2585     \keys_set:nn { NiceMatrix / xdots } { #3 }
2586     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }

```

```

2587         \@@_actually_draw_Ldots:
2588     \group_end:
2589 }
2590 }

```

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`.

```

2591 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2592 {
2593     \bool_if:NTF \l_@@_initial_open_bool
2594     {
2595         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2596         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2597         \dim_add:Nn \l_@@_x_initial_dim \col@sep
2598         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2599         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2600     }
2601     { \@@_set_initial_coords_from_anchor:n { base-east } }
2602     \bool_if:NTF \l_@@_final_open_bool
2603     {
2604         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2605         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2606         \dim_sub:Nn \l_@@_x_final_dim \col@sep
2607         \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2608         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2609     }
2610     { \@@_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 text. Of course, maybe we should not do that when the option `line-style` is used (?).

```

2611     \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2612     \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2613     \@@_draw_line:
2614 }

```

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.

```

2615 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2616 {
2617     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2618     {
2619         \@@_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.

```

2620     \group_begin:
2621     \int_compare:nNnTF { #1 } = 0
2622     { \color { nicematrix-first-row } }
2623     {

```

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.

```

2624         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2625         { \color { nicematrix-last-row } }
2626     }
2627     \keys_set:nn { NiceMatrix / xdots } { #3 }
2628     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2629     \@@_actually_draw_Cdots:
2630 \group_end:
2631 }
2632 }

```

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`.

```

2633 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2634 {
2635     \bool_if:NTF \l_@@_initial_open_bool
2636     {
2637         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2638         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2639         \dim_add:Nn \l_@@_x_initial_dim \col@sep
2640     }
2641     { \@@_set_initial_coords_from_anchor:n { mid-east } }
2642     \bool_if:NTF \l_@@_final_open_bool
2643     {
2644         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2645         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2646         \dim_sub:Nn \l_@@_x_final_dim \col@sep
2647     }
2648     { \@@_set_final_coords_from_anchor:n { mid-west } }
2649     \bool_lazy_and:nnTF
2650     \l_@@_initial_open_bool
2651     \l_@@_final_open_bool
2652     {
2653         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2654         \dim_set_eq:NN \l_tmpa_dim \pgf@y
2655         \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2656         \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2657         \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2658     }
2659     {
2660         \bool_if:NT \l_@@_initial_open_bool
2661         { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2662         \bool_if:NT \l_@@_final_open_bool
2663         { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2664     }
2665     \@@_draw_line:
2666 }

```

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.

```

2667 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2668 {

```

```

2669 \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2670 {
2671     \@@_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.

```

2672     \group_begin:
2673     \int_compare:nNnTF { #2 } = 0
2674     { \color { nicematrix-first-col } }
2675     {
2676         \int_compare:nNnT { #2 } = \l_@@_last_col_int
2677         { \color { nicematrix-last-col } }
2678     }
2679     \keys_set:nn { NiceMatrix / xdots } { #3 }
2680     \tl_if_empty:VF \l_@@_xdots_color_tl
2681     { \color { \l_@@_xdots_color_tl } }
2682     \@@_actually_draw_Vdots:
2683 \group_end:
2684 }
2685 }

```

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`.

```

2686 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2687 {

```

The boolean `\l_tmpa_bool` indicates whether the column is of type 1 or may be considered as if.

```

2688     \bool_set_false:N \l_tmpa_bool
2689     \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2690     {
2691         \@@_set_initial_coords_from_anchor:n { south-west }
2692         \@@_set_final_coords_from_anchor:n { north-west }
2693         \bool_set:Nn \l_tmpa_bool
2694         { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2695     }

```

Now, we try to determine whether the column is of type c or may be considered as if.

```

2696     \bool_if:NTF \l_@@_initial_open_bool
2697     {
2698         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2699         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2700     }
2701     { \@@_set_initial_coords_from_anchor:n { south } }
2702     \bool_if:NTF \l_@@_final_open_bool
2703     {
2704         \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2705         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2706     }
2707     { \@@_set_final_coords_from_anchor:n { north } }
2708     \bool_if:NTF \l_@@_initial_open_bool
2709     {
2710         \bool_if:NTF \l_@@_final_open_bool

```

```

2711 {
2712   \@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2713   \dim_set_eq:NN \l_tmpa_dim \pgf@x
2714   \@_qpoint:n { col - \@_succ:n \l_@@_initial_j_int }
2715   \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2716   \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.

```

2717   \int_compare:nNnT \l_@@_last_col_int > { -2 }
2718   {
2719     \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2720     {
2721       \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2722       \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2723       \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2724       \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2725     }
2726   }
2727 }
2728 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2729 }
2730 {
2731   \bool_if:NTF \l_@@_final_open_bool
2732   { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2733   {

```

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.

```

2734   \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2735   {
2736     \dim_set:Nn \l_@@_x_initial_dim
2737     {
2738       \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2739       \l_@@_x_initial_dim \l_@@_x_final_dim
2740     }
2741     \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2742   }
2743 }
2744 }
2745 \@_draw_line:
2746 }

```

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.

```

2747 \cs_new_protected:Npn \@_draw_Ddots:nnn #1 #2 #3
2748 {
2749   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2750   {
2751     \@_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.

```

2752   \group_begin:
2753   \keys_set:nn { NiceMatrix / xdots } { #3 }
2754   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2755   \@_actually_draw_Ddots:
2756   \group_end:
2757 }
2758 }

```



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.`

```

2759 \cs_new_protected:Npn \@@_actually_draw_Ddots:
2760 {
2761   \bool_if:NTF \l_@@_initial_open_bool
2762   {
2763     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2764     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2765     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2766     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2767   }
2768   { \@@_set_initial_coords_from_anchor:n { south-east } }
2769   \bool_if:NTF \l_@@_final_open_bool
2770   {
2771     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2772     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2773     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2774     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2775   }
2776   { \@@_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.

```

2777   \bool_if:NT \l_@@_parallelize_diags_bool
2778   {
2779     \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).

```

2780     \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.

```

2781     {
2782       \dim_gset:Nn \g_@@_delta_x_one_dim
2783       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2784       \dim_gset:Nn \g_@@_delta_y_one_dim
2785       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2786     }

```

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`.

```

2787     {
2788       \dim_set:Nn \l_@@_y_final_dim
2789       {
2790         \l_@@_y_initial_dim +
2791         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2792         \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
2793       }
2794     }
2795   }
2796   \@@_draw_line:
2797 }

```

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.

```

2798 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2799 {
2800   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2801   {
2802     \@@_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.

```

2803     \group_begin:
2804       \keys_set:nn { NiceMatrix / xdots } { #3 }
2805       \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2806       \@@_actually_draw_Iddots:
2807     \group_end:
2808   }
2809 }

```

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`.

```

2810 \cs_new_protected:Npn \@@_actually_draw_Iddots:
2811 {
2812   \bool_if:NTF \l_@@_initial_open_bool
2813   {
2814     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2815     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2816     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2817     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2818   }
2819   { \@@_set_initial_coords_from_anchor:n { south-west } }
2820   \bool_if:NTF \l_@@_final_open_bool
2821   {
2822     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2823     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2824     \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2825     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2826   }
2827   { \@@_set_final_coords_from_anchor:n { north-east } }
2828   \bool_if:NT \l_@@_parallelize_diags_bool
2829   {
2830     \int_gincr:N \g_@@_iddots_int
2831     \int_compare:nNnTF \g_@@_iddots_int = 1
2832     {
2833       \dim_gset:Nn \g_@@_delta_x_two_dim
2834       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2835       \dim_gset:Nn \g_@@_delta_y_two_dim
2836       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2837     }
2838     {
2839       \dim_set:Nn \l_@@_y_final_dim
2840       {
2841         \l_@@_y_initial_dim +

```

```

2842             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2843             \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2844         }
2845     }
2846 }
2847 \@@_draw_line:
2848 }

```

## 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`

```

2849 \cs_new_protected:Npn \@@_draw_line:
2850 {
2851     \pgfrememberpicturepositiononpagetrue
2852     \pgf@relevantforpicturesizefalse
2853     \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
2854         \@@_draw_standard_dotted_line:
2855         \@@_draw_non_standard_dotted_line:
2856 }

```

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.

```

2857 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2858 {
2859     \begin { scope }
2860     \exp_args:No \@@_draw_non_standard_dotted_line:n
2861         { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2862 }

```

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.

```

2863 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2864 {
2865     \@@_draw_non_standard_dotted_line:nVV
2866     { #1 }
2867     \l_@@_xdots_up_tl
2868     \l_@@_xdots_down_tl
2869 }
2870 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:nnn #1 #2 #3
2871 {
2872     \draw
2873     [
2874         #1 ,
2875         shorten-> = \l_@@_xdots_shorten_dim ,
2876         shorten-< = \l_@@_xdots_shorten_dim ,
2877     ]
2878     ( \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).

```

2879     -- node [ sloped , above ] { $ \scriptstyle #2 $ }
2880     node [ sloped , below ] { $ \scriptstyle #3 $ }
2881     ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2882 \end { scope }
2883 }
2884 \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).

```

2885 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2886 {
2887   \bool_lazy_and:nnF
2888   { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2889   { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2890   {
2891     \pgfscope
2892     \pgftransformshift
2893     {
2894       \pgfpointlineattime { 0.5 }
2895       { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2896       { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2897     }
2898     \pgftransformrotate
2899     {
2900       \fp_eval:n
2901       {
2902         atand
2903         (
2904           \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2905           \l_@@_x_final_dim - \l_@@_x_initial_dim
2906         )
2907       }
2908     }
2909     \pgfnode
2910     { rectangle }
2911     { south }
2912     {
2913       \c_math_toggle_token
2914       \scriptstyle \l_@@_xdots_up_tl
2915       \c_math_toggle_token
2916     }
2917     { }
2918     { \pgfusepath { } }
2919     \pgfnode
2920     { rectangle }
2921     { north }
2922     {
2923       \c_math_toggle_token
2924       \scriptstyle \l_@@_xdots_down_tl
2925       \c_math_toggle_token
2926     }
2927     { }
2928     { \pgfusepath { } }
2929     \endpgfscope
2930   }
2931   \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.

```

2932   \dim_zero_new:N \l_@@_l_dim
2933   \dim_set:Nn \l_@@_l_dim

```

```

2934 {
2935   \fp_to_dim:n
2936   {
2937     sqrt
2938     (
2939       ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
2940       +
2941       ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2942     )
2943   }
2944 }

```

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.

```

2945   \bool_lazy_or:nnF
2946   { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2947   { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2948   \@@_draw_standard_dotted_line_i:
2949 \group_end:
2950 }
2951 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2952 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2953 {

```

The integer `\l_tmpa_int` is the number of dots of the dotted line.

```

2954   \bool_if:NTF \l_@@_initial_open_bool
2955   {
2956     \bool_if:NTF \l_@@_final_open_bool
2957     {
2958       \int_set:Nn \l_tmpa_int
2959       { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2960     }
2961     {
2962       \int_set:Nn \l_tmpa_int
2963       {
2964         \dim_ratio:nn
2965         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2966         \l_@@_inter_dots_dim
2967       }
2968     }
2969   }
2970   {
2971     \bool_if:NTF \l_@@_final_open_bool
2972     {
2973       \int_set:Nn \l_tmpa_int
2974       {
2975         \dim_ratio:nn
2976         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2977         \l_@@_inter_dots_dim
2978       }
2979     }
2980     {
2981       \int_set:Nn \l_tmpa_int
2982       {
2983         \dim_ratio:nn
2984         { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2985         \l_@@_inter_dots_dim
2986       }
2987     }
2988   }

```

The dimensions `\l_tmpa_dim` and `\l_tmpb_dim` are the coordinates of the vector between two dots in the dotted line.

```

2989   \dim_set:Nn \l_tmpa_dim
2990   {
2991     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2992     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2993   }
2994   \dim_set:Nn \l_tmpb_dim
2995   {
2996     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2997     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2998   }

```

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`.

```

2999   \int_set:Nn \l_tmpb_int
3000   {
3001     \bool_if:NTF \l_@@_initial_open_bool
3002     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
3003     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
3004   }

```

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.

```

3005   \dim_gadd:Nn \l_@@_x_initial_dim
3006   {
3007     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3008     \dim_ratio:nn
3009     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3010     { 2 \l_@@_l_dim }
3011     * \l_tmpb_int
3012   }
3013   \dim_gadd:Nn \l_@@_y_initial_dim
3014   {
3015     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
3016     \dim_ratio:nn
3017     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3018     { 2 \l_@@_l_dim }
3019     * \l_tmpb_int
3020   }
3021   \pgf@relevantforpicturesizefalse
3022   \int_step_inline:nnn 0 \l_tmpa_int
3023   {
3024     \pgfpathcircle
3025     { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
3026     { \l_@@_radius_dim }
3027     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
3028     \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
3029   }
3030   \pgfusepathqfill
3031 }

```

## 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.

```

3032 \AtBeginDocument
3033 {
3034   \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
3035   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3036   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
3037   {
3038     \int_compare:nNnTF \c@jCol = 0
3039     { \@@_error:nn { in~first~col } \Ldots }
3040     {
3041       \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3042       { \@@_error:nn { in~last~col } \Ldots }
3043       {
3044         \@@_instruction_of_type:nnn \c_false_bool { Ldots }
3045         { #1 , down = #2 , up = #3 }
3046       }
3047     }
3048     \bool_if:NF \l_@@_nullify_dots_bool
3049     { \phantom { \ensuremath { \@@_old_ldots } } }
3050     \bool_gset_true:N \g_@@_empty_cell_bool
3051   }

3052   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
3053   {
3054     \int_compare:nNnTF \c@jCol = 0
3055     { \@@_error:nn { in~first~col } \Cdots }
3056     {
3057       \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3058       { \@@_error:nn { in~last~col } \Cdots }
3059       {
3060         \@@_instruction_of_type:nnn \c_false_bool { Cdots }
3061         { #1 , down = #2 , up = #3 }
3062       }
3063     }
3064     \bool_if:NF \l_@@_nullify_dots_bool
3065     { \phantom { \ensuremath { \@@_old_cdots } } }
3066     \bool_gset_true:N \g_@@_empty_cell_bool
3067   }

3068   \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
3069   {
3070     \int_compare:nNnTF \c@iRow = 0
3071     { \@@_error:nn { in~first~row } \Vdots }
3072     {
3073       \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
3074       { \@@_error:nn { in~last~row } \Vdots }
3075       {
3076         \@@_instruction_of_type:nnn \c_false_bool { Vdots }
3077         { #1 , down = #2 , up = #3 }
3078       }
3079     }
3080     \bool_if:NF \l_@@_nullify_dots_bool
3081     { \phantom { \ensuremath { \@@_old_vdots } } }
3082     \bool_gset_true:N \g_@@_empty_cell_bool
3083   }

```

```

3084 \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
3085 {
3086   \int_case:nnF \c@iRow
3087   {
3088     0 { \@@_error:nn { in~first~row } \Ddots }
3089     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
3090   }
3091   {
3092     \int_case:nnF \c@jCol
3093     {
3094       0 { \@@_error:nn { in~first~col } \Ddots }
3095       \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
3096     }
3097     {
3098       \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3099       \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Ddots }
3100       { #1 , down = #2 , up = #3 }
3101     }
3102   }
3103 }
3104 \bool_if:NF \l_@@_nullify_dots_bool
3105 { \phantom { \ensuremath { \@@_old_ddots } } }
3106 \bool_gset_true:N \g_@@_empty_cell_bool
3107 }

3108 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
3109 {
3110   \int_case:nnF \c@iRow
3111   {
3112     0 { \@@_error:nn { in~first~row } \Iddots }
3113     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
3114   }
3115   {
3116     \int_case:nnF \c@jCol
3117     {
3118       0 { \@@_error:nn { in~first~col } \Iddots }
3119       \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
3120     }
3121     {
3122       \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3123       \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Iddots }
3124       { #1 , down = #2 , up = #3 }
3125     }
3126   }
3127   \bool_if:NF \l_@@_nullify_dots_bool
3128   { \phantom { \ensuremath { \@@_old_iddots } } }
3129   \bool_gset_true:N \g_@@_empty_cell_bool
3130 }
3131 }

```

End of the \AtBeginDocument.

Despite its name, the following set of keys will be used for \Ddots but also for \Iddots.

```

3132 \keys_define:nn { NiceMatrix / Ddots }
3133 {
3134   draw-first .bool_set:N = \l_@@_draw_first_bool ,
3135   draw-first .default:n = true ,
3136   draw-first .value_forbidden:n = true
3137 }

```

The command \@@\_Hspace: will be linked to \hspace in {NiceArray}.

```

3138 \cs_new_protected:Npn \@@_Hspace:

```



```

3139 {
3140   \bool_gset_true:N \g_@@_empty_cell_bool
3141   \hspace
3142 }

```

In the environment `{NiceArray}`, the command `\multicolumn` will be linked to the following command `\@@_multicolumn:nnn`.

```

3143 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
3144 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
3145 {

```

We have to act in an expandable way since it will begin by a `\multicolumn`.

```

3146   \exp_args:NNe
3147   \@@_old_multicolumn
3148   { #1 }

```

We will have to replace `\tl_lower_case:n` in the future since it seems to be deprecated.

```

3149   {
3150     \exp_args:Ne \str_case:nn { \str_foldcase:n { #2 } }
3151     {
3152       l { > \@@_Cell: l < \@@_end_Cell: }
3153       r { > \@@_Cell: r < \@@_end_Cell: }
3154       c { > \@@_Cell: c < \@@_end_Cell: }
3155       { l | } { > \@@_Cell: l < \@@_end_Cell: | }
3156       { r | } { > \@@_Cell: r < \@@_end_Cell: | }
3157       { c | } { > \@@_Cell: c < \@@_end_Cell: | }
3158       { | l } { | > \@@_Cell: l < \@@_end_Cell: }
3159       { | r } { | > \@@_Cell: r < \@@_end_Cell: }
3160       { | c } { | > \@@_Cell: c < \@@_end_Cell: }
3161       { | l | } { | > \@@_Cell: l < \@@_end_Cell: | }
3162       { | r | } { | > \@@_Cell: r < \@@_end_Cell: | }
3163       { | c | } { | > \@@_Cell: c < \@@_end_Cell: | }
3164     }
3165   }
3166   { #3 }

```

The `\peek_remove_spaces:n` is mandatory.

```

3167   \peek_remove_spaces:n
3168   {
3169     \int_compare:nNnT #1 > 1
3170     {
3171       \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
3172       { \int_use:N \c@iRow - \int_use:N \c@jCol }
3173       \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
3174       \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
3175       {
3176         { \int_use:N \c@iRow }
3177         { \int_use:N \c@jCol }
3178         { \int_use:N \c@iRow }
3179         { \int_eval:n { \c@jCol + #1 - 1 } }
3180       }
3181     }
3182     \int_gadd:Nn \c@jCol { #1 - 1 }
3183     \int_compare:nNnT \c@jCol > \g_@@_col_total_int
3184     { \int_gset_eq:NN \g_@@_col_total_int \c@jCol }
3185   }
3186 }

```

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`.

```

3187 \cs_new:Npn \@@_Hdotsfor:
3188 {

```

```

3189 \bool_lazy_and:nnTF
3190 { \int_compare_p:nNn \c@jCol = 0 }
3191 { \int_compare_p:nNn \l_@@_first_col_int = 0 }
3192 {
3193   \bool_if:NTF \g_@@_after_col_zero_bool
3194   {
3195     \multicolumn { 1 } { c } { }
3196     \@@_Hdotsfor_i
3197   }
3198   { \@@_fatal:n { Hdotsfor~in~col~0 } }
3199 }
3200 {
3201   \multicolumn { 1 } { c } { }
3202   \@@_Hdotsfor_i
3203 }
3204 }

```

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:`).

```

3205 \AtBeginDocument
3206 {
3207   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3208   \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.

```

3209 \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
3210 {
3211   \tl_gput_right:Nx \g_@@_HVDotsfor_lines_tl
3212   {
3213     \@@_Hdotsfor:nnnn
3214     { \int_use:N \c@iRow }
3215     { \int_use:N \c@jCol }
3216     { #2 }
3217     {
3218       #1 , #3 ,
3219       down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3220     }
3221   }
3222   \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }
3223 }
3224 }

```

Enf of `\AtBeginDocument`.

```

3225 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
3226 {
3227   \bool_set_false:N \l_@@_initial_open_bool
3228   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

3229 \int_set:Nn \l_@@_initial_i_int { #1 }
3230 \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

For the column, it's a bit more complicated.

```

3231 \int_compare:nNnTF #2 = 1
3232 {
3233   \int_set:Nn \l_@@_initial_j_int 1
3234   \bool_set_true:N \l_@@_initial_open_bool
3235 }
3236 {
3237   \cs_if_exist:cTF
3238   {
3239     pgf @ sh @ ns @ \@@_env:

```

```

3240         - \int_use:N \l_@@_initial_i_int
3241         - \int_eval:n { #2 - 1 }
3242     }
3243     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
3244     {
3245         \int_set:Nn \l_@@_initial_j_int { #2 }
3246         \bool_set_true:N \l_@@_initial_open_bool
3247     }
3248 }
3249 \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
3250 {
3251     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3252     \bool_set_true:N \l_@@_final_open_bool
3253 }
3254 {
3255     \cs_if_exist:cTF
3256     {
3257         pgf @ sh @ ns @ \@@_env:
3258         - \int_use:N \l_@@_final_i_int
3259         - \int_eval:n { #2 + #3 }
3260     }
3261     { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
3262     {
3263         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3264         \bool_set_true:N \l_@@_final_open_bool
3265     }
3266 }
3267 \group_begin:
3268 \int_compare:nNnTF { #1 } = 0
3269 { \color { nicematrix-first-row } }
3270 {
3271     \int_compare:nNnT { #1 } = \g_@@_row_total_int
3272     { \color { nicematrix-last-row } }
3273 }
3274 \keys_set:nn { NiceMatrix / xdots } { #4 }
3275 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3276 \@@_actually_draw_ldots:
3277 \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).

```

3278     \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
3279     { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
3280 }

3281 \AtBeginDocument
3282 {
3283     \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3284     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3285     \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
3286     {
3287         \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
3288         {
3289             \@@_Vdotsfor:nnnn
3290             { \int_use:N \c@iRow }
3291             { \int_use:N \c@jCol }
3292             { #2 }
3293             {
3294                 #1 , #3 ,
3295                 down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3296             }
3297         }
3298     }
3299 }

```

```

3297     }
3298   }
3299 }

```

Enf of \AtBeginDocument.

```

3300 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
3301 {
3302   \bool_set_false:N \l_@@_initial_open_bool
3303   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it's easy.

```

3304   \int_set:Nn \l_@@_initial_j_int { #2 }
3305   \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

For the row, it's a bit more complicated.

```

3306   \int_compare:nNnTF #1 = 1
3307   {
3308     \int_set:Nn \l_@@_initial_i_int 1
3309     \bool_set_true:N \l_@@_initial_open_bool
3310   }
3311   {
3312     \cs_if_exist:cTF
3313     {
3314       pgf @ sh @ ns @ \@@_env:
3315       - \int_eval:n { #1 - 1 }
3316       - \int_use:N \l_@@_initial_j_int
3317     }
3318     { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
3319     {
3320       \int_set:Nn \l_@@_initial_i_int { #1 }
3321       \bool_set_true:N \l_@@_initial_open_bool
3322     }
3323   }
3324   \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
3325   {
3326     \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3327     \bool_set_true:N \l_@@_final_open_bool
3328   }
3329   {
3330     \cs_if_exist:cTF
3331     {
3332       pgf @ sh @ ns @ \@@_env:
3333       - \int_eval:n { #1 + #3 }
3334       - \int_use:N \l_@@_final_j_int
3335     }
3336     { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
3337     {
3338       \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3339       \bool_set_true:N \l_@@_final_open_bool
3340     }
3341   }
3342   \group_begin:
3343   \int_compare:nNnTF { #2 } = 0
3344   { \color { nicematrix-first-col } }
3345   {
3346     \int_compare:nNnT { #2 } = \g_@@_col_total_int
3347     { \color { nicematrix-last-col } }
3348   }
3349   \keys_set:nn { NiceMatrix / xdots } { #4 }
3350   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3351   \@@_actually_draw_Vdots:
3352   \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).

```

3353 \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
3354 { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
3355 }

```

The command `\@@_rotate:` will be linked to `\rotate` in `{NiceArrayWithDelims}`.

```

3356 \cs_new_protected:Npn \@@_rotate: { \bool_gset_true:N \g_@@_rotate_bool }

```

## The command `\line` accessible in code-after

In the `code-after`, 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>47</sup>

```

3357 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
3358 { \int_eval:n { #1 } - \int_eval:n { #2 } }

```

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).

```

3359 \AtBeginDocument
3360 {
3361   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
3362   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3363   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
3364   {
3365     \group_begin:
3366     \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
3367     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3368     \use:e
3369     {
3370       \@@_line_i:nn
3371       { \@@_double_int_eval:n #2 \q_stop }
3372       { \@@_double_int_eval:n #3 \q_stop }
3373     }
3374     \group_end:
3375   }
3376 }
3377 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3378 {
3379   \bool_set_false:N \l_@@_initial_open_bool
3380   \bool_set_false:N \l_@@_final_open_bool
3381   \bool_if:nTF
3382   {
3383     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3384     ||
3385     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3386   }
3387   {
3388     \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
3389   }

```

---

<sup>47</sup>Indeed, we want that the user may use the command `\line` in `code-after` with LaTeX counters in the arguments — with the command `\value`.

```

3390     { \@@_draw_line_ii:nn { #1 } { #2 } }
3391   }
3392 \AtBeginDocument
3393 {
3394   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3395   {

```

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:`.

```

3396     \c_@@_pgfortikzpicture_tl
3397     \@@_draw_line_iii:nn { #1 } { #2 }
3398     \c_@@_endpgfortikzpicture_tl
3399   }
3400 }

```

The following command *must* be protected (it’s used in the construction of `\@@_draw_line_ii:nn`).

```

3401 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3402 {
3403   \pgfrememberpicturepositiononpagetrue
3404   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3405   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3406   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3407   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3408   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3409   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3410   \@@_draw_line:
3411 }

```

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`.

```

3412 \cs_new_protected:Npn \@@_add_to_colors_seq:nn #1 #2
3413 {

```

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.

```

3414   \int_zero:N \l_tmpa_int
3415   \seq_map_indexed_inline:Nn \l_@@_colors_seq
3416   { \tl_if_eq:nnT { #1 } { ##2 } { \int_set:Nn \l_tmpa_int { ##1 } } }
3417   \int_compare:nNnTF \l_tmpa_int = \c_zero_int

```

First, the case where the color is a *new* color (not in the sequence).

```

3418 {
3419   \seq_put_right:Nn \l_@@_colors_seq { #1 }
3420   \tl_set:cn { l_@@_color _ \seq_count:N \l_@@_colors_seq _ tl } { #2 }
3421 }

```

Now, the case where the color is *not* a new color (the color is in the sequence at the position `\l_tmpa_int`).

```

3422 { \tl_put_right:cn { l_@@_color _ \int_use:N \l_tmpa_int _tl } { #2 } }
3423 }
3424 \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`).

```

3425 \cs_new_protected:Npn \@@_actually_color:
3426 {
3427   \pgfpicture
3428   \pgf@relevantforpicturesizefalse
3429   \seq_map_indexed_inline:Nn \l_@@_colors_seq
3430   {
3431     \color ##2
3432     \use:c { l_@@_color _ ##1 _tl }
3433     \pgfusepathqfill
3434   }
3435   \endpgfpicture
3436 }
3437 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3438 {
3439   \tl_set:Nn \l_tmpa_tl { #1 }
3440   \tl_set:Nn \l_tmpb_tl { #2 }
3441 }

```

Here is an example : `\@@_rowcolor {red!15} {1,3,5-7,10-}`

```

3442 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3443 {
3444   \tl_if_blank:nF { #2 }
3445   {
3446     \@@_add_to_colors_seq:xn
3447     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3448     { \@@_rowcolor:n { #3 } }
3449   }
3450 }
3451 \cs_new_protected:Npn \@@_rowcolor:n #1
3452 {
3453   \tl_set:Nn \l_@@_rows_tl { #1 }
3454   \tl_set:Nn \l_@@_cols_tl { - }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3455 \@@_cartesian_path:
3456 }

```

Here an example : `\@@_columncolor:nn {red!15} {1,3,5-7,10-}`

```

3457 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3458 {
3459   \tl_if_blank:nF { #2 }
3460   {
3461     \@@_add_to_colors_seq:xn
3462     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3463     { \@@_columncolor:n { #3 } }
3464   }
3465 }

```

```

3466 \cs_new_protected:Npn \@@_columncolor:n #1
3467 {
3468   \tl_set:Nn \l_@@_rows_tl { - }
3469   \tl_set:Nn \l_@@_cols_tl { #1 }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3470   \@@_cartesian_path:
3471 }

```

Here is an example : `\@@_rectanglecolor{red!15}{2-3}{5-6}`

```

3472 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3473 {
3474   \tl_if_blank:nF { #2 }
3475   {
3476     \@@_add_to_colors_seq:xn
3477     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3478     { \@@_rectanglecolor:nn { #3 } { #4 } }
3479   }
3480 }

```

```

3481 \cs_new_protected:Npn \@@_rectanglecolor:nn #1 #2
3482 {
3483   \@@_cut_on_hyphen:w #1 \q_stop
3484   \tl_clear_new:N \l_tmpc_tl
3485   \tl_clear_new:N \l_tmpd_tl
3486   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
3487   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
3488   \@@_cut_on_hyphen:w #2 \q_stop
3489   \tl_set:Nx \l_@@_rows_tl { \l_tmpc_tl - \l_tmpa_tl }
3490   \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`.

```

3491   \@@_cartesian_path:
3492 }

```

Here is an example : `\@@_cellcolor[rgb]{0.5,0.5,0}{2-3,3-4,4-5,5-6}`

```

3493 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3494 {
3495   \clist_map_inline:nn { #3 }
3496   { \@@_rectanglecolor [ #1 ] { #2 } { ##1 } { ##1 } }
3497 }

```

```

3498 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3499 {
3500   \int_step_inline:nn { \int_use:N \c@iRow }
3501   {
3502     \int_step_inline:nn { \int_use:N \c@jCol }
3503     {
3504       \int_if_even:nTF { ####1 + ##1 }
3505       { \@@_cellcolor [ #1 ] { #2 } }
3506       { \@@_cellcolor [ #1 ] { #3 } }
3507     } { ##1 - ####1 }
3508   }
3509 }
3510 }

```



```

3511 \keys_define:nn { NiceMatrix / rowcolors }
3512 {
3513   respect-blocks .bool_set:N = \l_@@_respect_blocks_bool ,
3514   respect-blocks .default:n = true ,
3515   cols .tl_set:N = \l_@@_cols_tl ,
3516   restart .bool_set:N = \l_@@_rowcolors_restart_bool ,
3517   restart .default:n = true ,
3518   unknown .code:n = \@@_error:n { Unknown~option~for~rowcolors }
3519 }

```

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.

```

3520 \NewDocumentCommand \@@_rowcolors { 0 { } m m m 0 { } }
3521 {

```

The group is for the options.

```

3522   \group_begin:
3523   \tl_clear_new:N \l_@@_cols_tl
3524   \tl_set:Nn \l_@@_cols_tl { - }
3525   \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.

```

3526   \bool_set_true:N \l_tmpa_bool
3527   \bool_lazy_and:nnT
3528     \l_@@_respect_blocks_bool
3529     {
3530       \cs_if_exist_p:c
3531         { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq } }
3532   {

```

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`).

```

3533     \seq_set_eq:Nc \l_tmpb_seq
3534     { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3535     \seq_set_filter:Nn \l_tmpa_seq \l_tmpb_seq
3536     { \@@_not_in_exterior_p:nnnn ##1 }
3537   }
3538   \pgfpicture
3539   \pgf@relevantforpicturesizefalse
3540   \clist_map_inline:nn { #2 }
3541   {
3542     \tl_set:Nn \l_tmpa_tl { ##1 }
3543     \tl_if_in:NnTF \l_tmpa_tl { - }
3544       { \@@_cut_on_hyphen:w ##1 \q_stop }
3545       { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }

```

The counter `\l_tmpa_int` will be the index of the loop.

```

3546     \int_set:Nn \l_tmpa_int \l_tmpa_tl
3547     \bool_if:NnTF \l_@@_rowcolors_restart_bool
3548       { \bool_set_true:N \l_tmpa_bool }
3549       { \bool_set:Nn \l_tmpa_bool { \int_if_odd_p:n { \l_tmpa_tl } } }
3550     \int_zero_new:N \l_tmppc_int
3551     \int_set:Nn \l_tmppc_int \l_tmppb_tl
3552     \int_do_until:nNnn \l_tmpa_int > \l_tmppc_int
3553     {

```

We will compute in `\l_tmppb_int` the last row of the “block”.

```

3554     \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).

```

3555         \bool_lazy_and:nnT
3556         \l_@@_respect_blocks_bool
3557         {
3558             \cs_if_exist_p:c
3559             { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3560         }
3561         {
3562             \seq_set_filter:Nn \l_tmpb_seq \l_tmpa_seq
3563             { \@@_intersect_our_row_p:nnnn ###1 }
3564             \seq_map_inline:Nn \l_tmpb_seq { \@@_rowcolors_i:nnnn ###1 }

```

Now, the last row of the block is computed in `\l_tmpb_int`.

```

3565     }
3566     \tl_set:Nx \l_@@_rows_tl
3567     { \int_use:N \l_tmpa_int - \int_use:N \l_tmpb_int }
3568     \bool_if:NTF \l_tmpa_bool
3569     {
3570         \tl_if_blank:nF { #3 }
3571         {
3572             \tl_if_empty:nTF { #1 }
3573             \color
3574             { \color [ #1 ] }
3575             { #3 }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3576         \@@_cartesian_path:
3577         \pgfusepathqfill
3578     }
3579     \bool_set_false:N \l_tmpa_bool
3580 }
3581 {
3582     \tl_if_blank:nF { #4 }
3583     {
3584         \tl_if_empty:nTF { #1 }
3585         \color
3586         { \color [ #1 ] }
3587         { #4 }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_row_tl`.

```

3588         \@@_cartesian_path:
3589         \pgfusepathqfill
3590     }
3591     \bool_set_true:N \l_tmpa_bool
3592 }
3593     \int_set:Nn \l_tmpa_int { \l_tmpb_int + 1 }
3594 }
3595 }
3596 \endpgfpicture
3597 \group_end:
3598 }

```

```

3599 \cs_new_protected:Npn \@@_rowcolors_i:nnnn #1 #2 #3 #4
3600 {
3601     \int_compare:nNnT { #3 } > \l_tmpb_int
3602     { \int_set:Nn \l_tmpb_int { #3 } }
3603 }

```

```

3604 \prg_new_conditional:Nnn \@@_not_in_exterior:nnnn p
3605 {
3606     \bool_lazy_or:nnTF

```

```

3607     { \int_compare_p:nNn { #4 } = \c_zero_int }
3608     { \int_compare_p:nNn { #2 } = { \@@_succ:n { \c@jCol } } }
3609     \prg_return_false:
3610     \prg_return_true:
3611 }

```

The following command return true when the block intersects the row \l\_tmpa\_int.

```

3612 \prg_new_conditional:Nnn \@@_intersect_our_row:nnnn p
3613 {
3614     \bool_if:nTF
3615     {
3616         \int_compare_p:n { #1 <= \l_tmpa_int }
3617         &&
3618         \int_compare_p:n { \l_tmpa_int <= #3 }
3619     }
3620     \prg_return_true:
3621     \prg_return_false:
3622 }

```

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.

```

3623 \cs_new_protected:Npn \@@_cartesian_path:
3624 {

```

We begin the loop over the columns.

```

3625     \clist_map_inline:Nn \l_@@_cols_tl
3626     {
3627         \tl_set:Nn \l_tmpa_tl { ##1 }
3628         \tl_if_in:NnTF \l_tmpa_tl { - }
3629         { \@@_cut_on_hyphen:w ##1 \q_stop }
3630         { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3631         \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3632         \tl_if_empty:NT \l_tmpb_tl
3633         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3634         \int_compare:nNnT \l_tmpb_tl > \c@jCol
3635         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3636         \@@_qpoint:n { col - \l_tmpa_tl }
3637         \int_compare:nNnTF \l_@@_first_col_int = \l_tmpa_tl
3638         { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3639         { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3640         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3641         \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }

```

We begin the loop over the rows

```

3642     \clist_map_inline:Nn \l_@@_rows_tl
3643     {
3644         \tl_set:Nn \l_tmpa_tl { ####1 }
3645         \tl_if_in:NnTF \l_tmpa_tl { - }
3646         { \@@_cut_on_hyphen:w ####1 \q_stop }
3647         { \@@_cut_on_hyphen:w ####1 - ####1 \q_stop }
3648         \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3649         \tl_if_empty:NT \l_tmpb_tl
3650         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3651         \int_compare:nNnT \l_tmpb_tl > \c@iRow
3652         { \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.

```

3653         \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3654         \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3655         \@@_qpoint:n { row - \l_tmpa_tl }
3656         \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3657         \pgfpathrectanglecorners

```

```

3658         { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3659         { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3660     }
3661 }
3662 }

```

When the user uses the key `colortbl-like`, the following command will be linked to `\cellcolor` in the tabular.

```

3663 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
3664 {
3665     \tl_gput_right:Nx \g_nicematrix_code_before_tl
3666     { \cellcolor [ #1 ] { #2 } { \int_use:N \c@iRow - \int_use:N \c@jCol } }
3667 }

```

When the user uses the key `colortbll-like`, the following command will be linked to `\rowcolor` in the tabular.

```

3668 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
3669 {
3670     \tl_gput_right:Nx \g_nicematrix_code_before_tl
3671     {
3672         \exp_not:N \rectanglecolor [ #1 ] { #2 }
3673         { \int_use:N \c@iRow - \int_use:N \c@jCol }
3674         { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
3675     }
3676 }

```

```

3677 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
3678 {
3679     \int_compare:nNnT \c@iRow = 1
3680     {

```

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).

```

3681         \tl_gput_left:Nx \g_nicematrix_code_before_tl
3682         { \exp_not:N \columncolor [ #1 ] { #2 } { \int_use:N \c@jCol } }
3683     }
3684 }

```

## 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`.

```

3685 \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`.

```

3686 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
3687 {
3688     \int_compare:nNnTF \l_@@_first_col_int = 0
3689     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3690     {
3691         \int_compare:nNnTF \c@jCol = 0
3692         {

```

```

3693         \int_compare:nNnF \c@iRow = { -1 }
3694         { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
3695     }
3696     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3697 }
3698 }

```

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* an 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.

```

3699 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
3700 {
3701     \int_compare:nNnF \c@iRow = 0
3702     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
3703 }

```

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:nNnT \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`. `#2` is the number of consecutive occurrences of `|`.

```

3704 \cs_new_protected:Npn \@@_vline:nn #1 #2
3705 {

```

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).

```

3706     \int_compare:nNnT { #1 } < { \c@jCol + 2 }
3707     {
3708         \pgfpicture
3709         \@@_vline_i:nn { #1 } { #2 }
3710         \endpgfpicture
3711     }
3712 }
3713 \cs_new_protected:Npn \@@_vline_i:nn #1 #2
3714 {

```

`\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_tmppc_tl`.

```

3715     \tl_set:Nx \l_tmpb_tl { #1 }
3716     \tl_clear_new:N \l_tmppc_tl
3717     \int_step_variable:nNn \c@iRow \l_tmpa_tl
3718     {

```

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.

```

3719         \bool_gset_true:N \g_tmpa_bool
3720         \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3721         { \@@_test_if_vline_in_block:nnnn #1 }
3722         \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3723         { \@@_test_if_vline_in_block:nnnn #1 }
3724         \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
3725         { \@@_test_if_vline_in_stroken_block:nnnn #1 }
3726         \clist_if_empty:NF \l_@@_except_corners_clist
3727         \@@_test_in_corner_v:
3728         \bool_if:NTF \g_tmpa_bool
3729         {
3730             \tl_if_empty:NT \l_tmppc_tl

```

We keep in memory that we have a rule to draw.

```

3731         { \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl }
3732     }
3733     {
3734         \tl_if_empty:NF \l_tmpc_tl
3735         {
3736             \@@_vline_ii:nnnn
3737             { #1 }
3738             { #2 }
3739             \l_tmpc_tl
3740             { \int_eval:n { \l_tmpa_tl - 1 } }
3741             \tl_clear:N \l_tmpc_tl
3742         }
3743     }
3744 }
3745 \tl_if_empty:NF \l_tmpc_tl
3746 {
3747     \@@_vline_ii:nnnn
3748     { #1 }
3749     { #2 }
3750     \l_tmpc_tl
3751     { \int_use:N \c@iRow }
3752     \tl_clear:N \l_tmpc_tl
3753 }
3754 }

3755 \cs_new_protected:Npn \@@_test_in_corner_v:
3756 {
3757     \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
3758     {
3759         \seq_if_in:NxT
3760         \l_@@_empty_corner_cells_seq
3761         { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3762         { \bool_set_false:N \g_tmpa_bool }
3763     }
3764     {
3765         \seq_if_in:NxT
3766         \l_@@_empty_corner_cells_seq
3767         { \l_tmpa_tl - \l_tmpb_tl }
3768         {
3769             \int_compare:nNnTF \l_tmpb_tl = 1
3770             { \bool_set_false:N \g_tmpa_bool }
3771             {
3772                 \seq_if_in:NxT
3773                 \l_@@_empty_corner_cells_seq
3774                 { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3775                 { \bool_set_false:N \g_tmpa_bool }
3776             }
3777         }
3778     }
3779 }

```

#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 number of the rows between which the rule has to be drawn.

```

3780 \cs_new_protected:Npn \@@_vline_ii:nnnn #1 #2 #3 #4
3781 {
3782     \pgfrememberpicturepositiononpagetrue
3783     \pgf@relevantforpicturesizefalse
3784     \@@_qpoint:n { row - #3 }
3785     \dim_set_eq:NN \l_tmpa_dim \pgf@y
3786     \@@_qpoint:n { col - #1 }
3787     \dim_set_eq:NN \l_tmpb_dim \pgf@x

```

```

3788 \@@_qpoint:n { row - \@@_succ:n { #4 } }
3789 \dim_set_eq:NN \l_tmpc_dim \pgf@y
3790 \bool_lazy_and:nnT
3791 { \int_compare_p:nNn { #2 } > 1 }
3792 { ! \tl_if_blank_p:V \CT@drsc@ }
3793 {
3794   \group_begin:
3795   \CT@drsc@
3796   \dim_add:Nn \l_tmpa_dim { 0.5 \arrayrulewidth }
3797   \dim_sub:Nn \l_tmpc_dim { 0.5 \arrayrulewidth }
3798   \dim_set:Nn \l_tmpd_dim
3799     { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3800   \pgfpathrectanglecorners
3801     { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3802     { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
3803   \pgfusepathqfill
3804   \group_end:
3805 }
3806 \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3807 \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3808 \prg_replicate:nn { #2 - 1 }
3809 {
3810   \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3811   \dim_sub:Nn \l_tmpb_dim \doublerulesep
3812   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3813   \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3814 }
3815 \CT@arc@
3816 \pgfsetlinewidth { 1.1 \arrayrulewidth }
3817 \pgfsetrectcap
3818 \pgfusepathqstroke
3819 }

```

The following 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).

```

3820 \cs_new_protected:Npn \@@_vline_i_complete:nn #1 #2
3821 { \@@_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).

```

3822 \cs_new_protected:Npn \@@_draw_vlines:
3823 {
3824   \int_step_inline:nnn
3825     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3826     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3827     { \@@_vline:nn { ##1 } 1 }
3828 }

```

## 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`.

```

3829 \cs_new_protected:Npn \@@_hline:nn #1 #2
3830 {
3831   \pgfpicture
3832   \@@_hline_i:nn { #1 } { #2 }
3833   \endpgfpicture
3834 }

```

```

3835 \cs_new_protected:Npn \@@_hline_i:nn #1 #2
3836 {

```

\l\_tmpa\_tl is the number of row and \l\_tmpb\_tl the number of column. Whe, we have found a column corresponding to a rule to draw, we note its number in \l\_tmpc\_tl.

```

3837 \tl_set:Nn \l_tmpa_tl { #1 }
3838 \tl_clear_new:N \l_tmpc_tl
3839 \int_step_variable:nNn \c@jCol \l_tmpb_tl
3840 {

```

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.

```

3841 \bool_gset_true:N \g_tmpa_bool
3842 \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3843 { \@@_test_if_hline_in_block:nnnn ##1 }
3844 \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3845 { \@@_test_if_hline_in_block:nnnn ##1 }
3846 \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
3847 { \@@_test_if_hline_in_stroken_block:nnnn ##1 }
3848 \clist_if_empty:NF \l_@@_except_corners_clist \@@_test_in_corner_h:
3849 \bool_if:NTF \g_tmpa_bool
3850 {
3851 \tl_if_empty:NT \l_tmpc_tl

```

We keep in memory that we have a rule to draw.

```

3852 { \tl_set_eq:NN \l_tmpc_tl \l_tmpb_tl }
3853 }
3854 {
3855 \tl_if_empty:NF \l_tmpc_tl
3856 {
3857 \@@_hline_ii:nnnn
3858 { #1 }
3859 { #2 }
3860 \l_tmpc_tl
3861 { \int_eval:n { \l_tmpb_tl - 1 } }
3862 \tl_clear:N \l_tmpc_tl
3863 }
3864 }
3865 }
3866 \tl_if_empty:NF \l_tmpc_tl
3867 {
3868 \@@_hline_ii:nnnn
3869 { #1 }
3870 { #2 }
3871 \l_tmpc_tl
3872 { \int_use:N \c@jCol }
3873 \tl_clear:N \l_tmpc_tl
3874 }
3875 }

```

```

3876 \cs_new_protected:Npn \@@_test_in_corner_h:
3877 {
3878 \int_compare:nNnTF \l_tmpa_tl = { \@@_succ:n \c@iRow }
3879 {
3880 \seq_if_in:NxT
3881 \l_@@_empty_corner_cells_seq
3882 { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3883 { \bool_set_false:N \g_tmpa_bool }
3884 }
3885 {
3886 \seq_if_in:NxT
3887 \l_@@_empty_corner_cells_seq

```



```

3888     { \l_tmpa_tl - \l_tmpb_tl }
3889     {
3890         \int_compare:nNnTF \l_tmpa_tl = 1
3891         { \bool_set_false:N \g_tmpa_bool }
3892         {
3893             \seq_if_in:NxT
3894             \l_@@_empty_corner_cells_seq
3895             { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3896             { \bool_set_false:N \g_tmpa_bool }
3897         }
3898     }
3899 }
3900 }

```

#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.

```

3901 \cs_new_protected:Npn \@@_hline_ii:nnnn #1 #2 #3 #4
3902 {
3903     \pgfrememberpicturepositiononpagetrue
3904     \pgf@relevantforpicturesizefalse
3905     \@@_qpoint:n { col - #3 }
3906     \dim_set_eq:NN \l_tmpa_dim \pgf@x
3907     \@@_qpoint:n { row - #1 }
3908     \dim_set_eq:NN \l_tmpb_dim \pgf@y
3909     \@@_qpoint:n { col - \@@_succ:n { #4 } }
3910     \dim_set_eq:NN \l_tmpc_dim \pgf@x
3911     \bool_lazy_and:nnT
3912     { \int_compare_p:nNn { #2 } > 1 }
3913     { ! \tl_if_blank_p:V \CT@drsc@ }
3914     {
3915         \group_begin:
3916         \CT@drsc@
3917         \dim_set:Nn \l_tmpd_dim
3918         { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3919         \pgfpathrectanglecorners
3920         { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3921         { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3922         \pgfusepathqfill
3923         \group_end:
3924     }
3925     \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3926     \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3927     \prg_replicate:nn { #2 - 1 }
3928     {
3929         \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3930         \dim_sub:Nn \l_tmpb_dim \doublerulesep
3931         \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3932         \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3933     }
3934     \CT@arc@
3935     \pgfsetlinewidth { 1.1 \arrayrulewidth }
3936     \pgfsetrectcap
3937     \pgfusepathqstroke
3938 }

```

```

3939 \cs_new_protected:Npn \@@_hline_i_complete:nn #1 #2
3940 { \@@_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).

```

3941 \cs_new_protected:Npn \@@_draw_hlines:
3942 {
3943   \int_step_inline:nnn
3944     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3945     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3946     { \@@_hline:nn { ##1 } 1 }
3947 }

```

The command `\@@_Hline:` will be linked to `\Hline` in the environments of `nicematrix`.

```

3948 \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.

```

3949 \cs_set:Npn \@@_Hline_i:n #1
3950 {
3951   \peek_meaning_ignore_spaces:NTF \Hline
3952     { \@@_Hline_ii:nn { #1 + 1 } }
3953     { \@@_Hline_iii:n { #1 } }
3954 }
3955 \cs_set:Npn \@@_Hline_ii:nn #1 #2 { \@@_Hline_i:n { #1 } }
3956 \cs_set:Npn \@@_Hline_iii:n #1
3957 {
3958   \skip_vertical:n
3959   {
3960     \arrayrulewidth * ( #1 )
3961     + \doublerulesep * ( \int_max:nn 0 { #1 - 1 } )
3962   }
3963   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3964     { \@@_hline:nn { \@@_succ:n { \c@iRow } } { #1 } }
3965   \ifnum 0 = ` { \fi }
3966 }

```

## 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 col) 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.

```

3967 \cs_new_protected:Npn \@@_test_if_hline_in_block:nnnn #1 #2 #3 #4
3968 {
3969   \bool_lazy_all:nT
3970   {
3971     { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
3972     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3973     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3974     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3975   }
3976   { \bool_gset_false:N \g_tmpa_bool }
3977 }

```

The same for vertical rules.

```

3978 \cs_new_protected:Npn \@@_test_if_vline_in_block:nnnn #1 #2 #3 #4
3979 {
3980   \bool_lazy_all:nT
3981   {
3982     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3983     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3984     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
3985     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3986   }
3987   { \bool_gset_false:N \g_tmpa_bool }
3988 }

```

```

3989 \cs_new_protected:Npn \@@_test_if_hline_in_stroken_block:nnnn #1 #2 #3 #4
3990 {
3991   \bool_lazy_all:nT
3992   {
3993     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3994     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 2 } }
3995     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3996     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3997   }
3998   { \bool_gset_false:N \g_tmpa_bool }
3999 }

4000 \cs_new_protected:Npn \@@_test_if_vline_in_stroken_block:nnnn #1 #2 #3 #4
4001 {
4002   \bool_lazy_all:nT
4003   {
4004     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4005     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4006     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4007     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 2 } }
4008   }
4009   { \bool_gset_false:N \g_tmpa_bool }
4010 }

```

## 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.

```

4011 \cs_new_protected:Npn \@@_compute_corners:
4012 {

```

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.

```

4013   \seq_clear_new:N \l_@@_empty_corner_cells_seq
4014   \clist_map_inline:Nn \l_@@_except_corners_clist
4015   {
4016     \str_case:nnF { ##1 }
4017     {
4018       { NW }
4019       { \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol }
4020       { NE }
4021       { \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1 }
4022       { SW }
4023       { \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol }
4024       { SE }
4025       { \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1 }
4026     }
4027     { \@@_error:nn { bad~corner } { ##1 } }
4028   }
4029 }

```

“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.

```

4030 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
4031 {

```

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.

```

4032 \bool_set_false:N \l_tmpa_bool
4033 \int_zero_new:N \l_@@_last_empty_row_int
4034 \int_set:Nn \l_@@_last_empty_row_int { #1 }
4035 \int_step_inline:nnnn { #1 } { #3 } { #5 }
4036 {
4037   \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
4038   \bool_lazy_or:nnTF
4039     {
4040       \cs_if_exist_p:c
4041         { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
4042     }
4043     \l_tmpb_bool
4044     { \bool_set_true:N \l_tmpa_bool }
4045     {
4046       \bool_if:NF \l_tmpa_bool
4047       { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
4048     }
4049 }

```

Now, you determine the last empty cell in the row of number 1.

```

4050 \bool_set_false:N \l_tmpa_bool
4051 \int_zero_new:N \l_@@_last_empty_column_int
4052 \int_set:Nn \l_@@_last_empty_column_int { #2 }
4053 \int_step_inline:nnnn { #2 } { #4 } { #6 }
4054 {
4055   \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
4056   \bool_lazy_or:nnTF
4057     \l_tmpb_bool
4058     {
4059       \cs_if_exist_p:c
4060         { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
4061     }
4062     { \bool_set_true:N \l_tmpa_bool }
4063     {
4064       \bool_if:NF \l_tmpa_bool
4065       { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
4066     }
4067 }

```

Now, we loop over the rows.

```

4068 \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
4069 {

```

We treat the row number `##1` with another loop.

```

4070 \bool_set_false:N \l_tmpa_bool
4071 \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
4072 {
4073   \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
4074   \bool_lazy_or:nnTF
4075     \l_tmpb_bool
4076     {
4077       \cs_if_exist_p:c
4078         { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
4079     }
4080     { \bool_set_true:N \l_tmpa_bool }
4081     {
4082       \bool_if:NF \l_tmpa_bool

```

```

4083         {
4084             \int_set:Nn \l_@@_last_empty_column_int { ####1 }
4085             \seq_put_right:Nn
4086                 \l_@@_empty_corner_cells_seq
4087                 { ##1 - ####1 }
4088         }
4089     }
4090 }
4091 }
4092 }

```

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`).

```

4093 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
4094 {
4095     \int_set:Nn \l_tmpa_int { #1 }
4096     \int_set:Nn \l_tmpb_int { #2 }
4097     \bool_set_false:N \l_tmpb_bool
4098     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4099         { \@@_test_if_cell_in_block:nnnnnnn \l_tmpa_int \l_tmpb_int ##1 }
4100 }
4101 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnnn #1 #2 #3 #4 #5 #6
4102 {
4103     \int_compare:nNnT { #3 } < { \@@_succ:n { #1 } }
4104     {
4105         \int_compare:nNnT { #1 } < { \@@_succ:n { #5 } }
4106         {
4107             \int_compare:nNnT { #4 } < { \@@_succ:n { #2 } }
4108             {
4109                 \int_compare:nNnT { #2 } < { \@@_succ:n { #6 } }
4110                 { \bool_set_true:N \l_tmpb_bool }
4111             }
4112         }
4113     }
4114 }

```

## 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`.

```

4115 \cs_new:Npn \@@_hdottedline:
4116 {
4117     \noalign { \skip_vertical:N 2\l_@@_radius_dim }
4118     \@@_hdottedline_i:
4119 }

```

On the other side, the following command should be protected.

```

4120 \cs_new_protected:Npn \@@_hdottedline_i:
4121 {

```

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).

```

4122     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4123     { \@@_hdottedline:n { \int_use:N \c@iRow } }
4124 }

```

The command `\@@_hdottedline:n` is the command written in the `code-after` that will actually draw the dotted line. Its argument is the number of the row *before* which we will draw the row.

```
4125 \AtBeginDocument
4126 {
```

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}`).

```
4127 \cs_new_protected:Npx \@@_hdottedline:n #1
4128 {
4129   \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
4130   \bool_set_true:N \exp_not:N \l_@@_final_open_bool
4131   \c_@@_pgfortikzpicture_tl
4132   \@@_hdottedline_i:n { #1 }
4133   \c_@@_endpgfortikzpicture_tl
4134 }
4135 }
```

The following command *must* be protected since it is used in the construction of `\@@_hdottedline:n`.

```
4136 \cs_new_protected:Npn \@@_hdottedline_i:n #1
4137 {
4138   \pgfrememberpicturepositiononpagetrue
4139   \@@_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).

```
4140 \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
4141 \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
4142 \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}
```

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

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}
```

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

```
4143 \@@_qpoint:n { col - 1 }
4144 \dim_set:Nn \l_@@_x_initial_dim
4145 {
4146   \pgf@x +
```

We do a reduction by `\arraycolsep` for the environments with delimiters (and not for the other).

```
4147 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4148 - \l_@@_left_margin_dim
4149 }
4150 \@@_qpoint:n { col - \@@_succ:n \c@jCol }
4151 \dim_set:Nn \l_@@_x_final_dim
4152 {
4153   \pgf@x -
```

```

4154     \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4155     + \l_@@_right_margin_dim
4156   }

```

For reasons purely aesthetic, we do an adjustment in the case of a rounded bracket. The correction by  $0.5 \text{ } \l_@@\_inter\_dots\_dim$  is *ad hoc* for a better result.

```

4157   \tl_set:Nn \l_tmpa_tl { ( }
4158   \tl_if_eq:NNF \l_@@_left_delim_tl \l_tmpa_tl
4159   { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
4160   \tl_set:Nn \l_tmpa_tl { ) }
4161   \tl_if_eq:NNF \l_@@_right_delim_tl \l_tmpa_tl
4162   { \dim_gsub:Nn \l_@@_x_final_dim { 0.5 \l_@@_inter_dots_dim } }

```

As of 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`.

```

4163   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4164   \@@_draw_line:
4165 }

```

## Vertical dotted lines

```

4166 \cs_new_protected:Npn \@@_vdottedline:n #1
4167 {
4168   \bool_set_true:N \l_@@_initial_open_bool
4169   \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”.

```

4170   \bool_if:NTF \c_@@_tikz_loaded_bool
4171   {
4172     \tikzpicture
4173     \@@_vdottedline_i:n { #1 }
4174     \endtikzpicture
4175   }
4176   {
4177     \pgfpicture
4178     \@@_vdottedline_i:n { #1 }
4179     \endpgfpicture
4180   }
4181 }

```

```

4182 \cs_new_protected:Npn \@@_vdottedline_i:n #1
4183 {

```

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.

```

4184   \CT@arc@
4185   \pgfrememberpicturepositiononpagetrue
4186   \@@_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).

```

4187   \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
4188   \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
4189   \@@_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.

```

4190   \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
4191   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
4192   \dim_set:Nn \l_@@_y_final_dim { \pgf@y + 0.5 \l_@@_inter_dots_dim }

```

As of 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`.

```

4193     \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4194     \@@_draw_line:
4195 }

```

## 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.

```

4196 \bool_new:N \l_@@_block_auto_columns_width_bool

```

As of now, there is only one option available for the environment `{NiceMatrixBlock}`.

```

4197 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
4198 {
4199     auto-columns-width .code:n =
4200     {
4201         \bool_set_true:N \l_@@_block_auto_columns_width_bool
4202         \dim_gzero_new:N \g_@@_max_cell_width_dim
4203         \bool_set_true:N \l_@@_auto_columns_width_bool
4204     }
4205 }

4206 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
4207 {
4208     \int_gincr:N \g_@@_NiceMatrixBlock_int
4209     \dim_zero:N \l_@@_columns_width_dim
4210     \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
4211     \bool_if:NT \l_@@_block_auto_columns_width_bool
4212     {
4213         \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4214         {
4215             \exp_args:Nnc \dim_set:Nn \l_@@_columns_width_dim
4216             { @@_max_cell_width _ \int_use:N \g_@@_NiceMatrixBlock_int }
4217         }
4218     }
4219 }

```

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`).

```

4220 {
4221     \bool_if:NT \l_@@_block_auto_columns_width_bool
4222     {
4223         \iow_shipout:Nn \@mainaux \ExplSyntaxOn
4224         \iow_shipout:Nx \@mainaux
4225         {
4226             \cs_gset:cpn
4227             { @@ _ 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.

```

4228         { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
4229     }
4230     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
4231 }
4232 }

```



## The extra nodes

First, two variants of the functions `\dim_min:nn` and `\dim_max:nn`.

```
4233 \cs_generate_variant:Nn \dim_min:nn { v n }
4234 \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).

```
4235 \cs_new_protected:Npn \@@_create_extra_nodes:
4236 {
4237   \bool_if:nTF \l_@@_medium_nodes_bool
4238   {
4239     \bool_if:nTF \l_@@_large_nodes_bool
4240     \@@_create_medium_and_large_nodes:
4241     \@@_create_medium_nodes:
4242   }
4243   { \bool_if:nT \l_@@_large_nodes_bool \@@_create_large_nodes: }
4244 }
```

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`.

```
4245 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
4246 {
4247   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4248   {
4249     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
4250     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
4251     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
4252     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
4253   }
4254   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4255   {
4256     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
4257     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
4258     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
4259     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
4260   }
}
```

We begin the two nested loops over the rows and the columns of the array.

```
4261 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4262 {
4263   \int_step_variable:nnNn
4264   \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.

```

4265     {
4266         \cs_if_exist:cT
4267         { 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.

```

4268     {
4269         \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
4270         \dim_set:cn { l_@@_row _ \@@_i: _ min_dim }
4271         { \dim_min:vn { l_@@_row _ \@@_i: _ min_dim } \pgf@y }
4272         \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4273         {
4274             \dim_set:cn { l_@@_column _ \@@_j: _ min_dim }
4275             { \dim_min:vn { l_@@_column _ \@@_j: _ min_dim } \pgf@x }
4276         }

```

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.

```

4277         \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
4278         \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
4279         { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
4280         \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4281         {
4282             \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
4283             { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
4284         }
4285     }
4286 }
4287 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

4288 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4289 {
4290     \dim_compare:nNnT
4291     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
4292     {
4293         \@@_qpoint:n { row - \@@_i: - base }
4294         \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
4295         \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
4296     }
4297 }
4298 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4299 {
4300     \dim_compare:nNnT
4301     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
4302     {
4303         \@@_qpoint:n { col - \@@_j: }
4304         \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@x
4305         \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@x
4306     }
4307 }
4308 }

```

Here is the command \@@\_create\_medium\_nodes:. When this command is used, the “medium nodes” are created.

```

4309 \cs_new_protected:Npn \@@_create_medium_nodes:
4310 {
4311     \pgfpicture
4312     \pgfrememberpicturepositiononpagetrue
4313     \pgf@relevantforpicturesizefalse
4314     \@@_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”.

```

4315     \tl_set:Nn \l_@@_suffix_tl { -medium }
4316     \@@_create_nodes:
4317     \endpgfpicture
4318 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones<sup>48</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:`.

```

4319 \cs_new_protected:Npn \@@_create_large_nodes:
4320 {
4321     \pgfpicture
4322     \pgfrememberpicturepositiononpagetrue
4323     \pgf@relevantforpicturesizefalse
4324     \@@_computations_for_medium_nodes:
4325     \@@_computations_for_large_nodes:
4326     \tl_set:Nn \l_@@_suffix_tl { - large }
4327     \@@_create_nodes:
4328     \endpgfpicture
4329 }

4330 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
4331 {
4332     \pgfpicture
4333     \pgfrememberpicturepositiononpagetrue
4334     \pgf@relevantforpicturesizefalse
4335     \@@_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”.

```

4336     \tl_set:Nn \l_@@_suffix_tl { - medium }
4337     \@@_create_nodes:
4338     \@@_computations_for_large_nodes:
4339     \tl_set:Nn \l_@@_suffix_tl { - large }
4340     \@@_create_nodes:
4341     \endpgfpicture
4342 }

```

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.

```

4343 \cs_new_protected:Npn \@@_computations_for_large_nodes:
4344 {
4345     \int_set:Nn \l_@@_first_row_int 1
4346     \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`.

```

4347     \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
4348     {
4349         \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
4350         {
4351             (
4352                 \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
4353                 \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4354             )
4355             / 2
4356         }
4357         \dim_set:eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }

```

---

<sup>48</sup>If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

```

4358     { l_@@_row\_@@_i: _min_dim }
4359 }
4360 \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
4361 {
4362     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
4363     {
4364         (
4365             \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
4366             \dim_use:c
4367                 { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4368         )
4369         / 2
4370     }
4371     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4372     { l_@@_column _ \@@_j: _ max _ dim }
4373 }

```

Here, we have to use `\dim_sub:cn` because of the number 1 in the name.

```

4374     \dim_sub:cn
4375     { l_@@_column _ 1 _ min _ dim }
4376     \l_@@_left_margin_dim
4377     \dim_add:cn
4378     { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
4379     \l_@@_right_margin_dim
4380 }

```

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).

```

4381 \cs_new_protected:Npn \@@_create_nodes:
4382 {
4383     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4384     {
4385         \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4386         {

```

We draw the rectangular node for the cell (`\@@_i-\@@_j`).

```

4387         \@@_pgf_rect_node:nnnnn
4388         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4389         { \dim_use:c { l_@@_column _ \@@_j: _ min_dim } }
4390         { \dim_use:c { l_@@_row _ \@@_i: _ min_dim } }
4391         { \dim_use:c { l_@@_column _ \@@_j: _ max_dim } }
4392         { \dim_use:c { l_@@_row _ \@@_i: _ max_dim } }
4393         \str_if_empty:NF \l_@@_name_str
4394         {
4395             \pgfnodealias
4396             { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4397             { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4398         }
4399     }
4400 }

```

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  $n$ .

```

4401     \seq_mapthread_function:NNN
4402     \g_@@_multicolumn_cells_seq
4403     \g_@@_multicolumn_sizes_seq
4404     \@@_node_for_multicolumn:nn
4405 }

```

```

4406 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
4407 {
4408   \cs_set_nopar:Npn \@@_i: { #1 }
4409   \cs_set_nopar:Npn \@@_j: { #2 }
4410 }

```

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”).

```

4411 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
4412 {
4413   \@@_extract_coords_values: #1 \q_stop
4414   \@@_pgf_rect_node:nnnnn
4415     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4416     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }
4417     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } }
4418     { \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
4419     { \dim_use:c { l_@@_row _ \@@_i: _ max _ dim } }
4420   \str_if_empty:NF \l_@@_name_str
4421   {
4422     \pgfnodealias
4423       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4424       { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
4425   }
4426 }

```

## 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.

```

4427 \keys_define:nn { NiceMatrix / Block / FirstPass }
4428 {
4429   l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4430   l .value_forbidden:n = true ,
4431   r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4432   r .value_forbidden:n = true ,
4433   c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4434   c .value_forbidden:n = true ,

```

The two following lines will be uncommented when we will give to the key `color` its new definition.

```

4435   % color .tl_set:N = \l_@@_color_tl ,
4436   % color .value_required:n = true ,
4437 }

```

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.

```

4438 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
4439 {

```

If the first mandatory argument of the command (which is the size of the block with the syntax  $i$ - $j$ ) has not be provided by the user, you use 1-1 (that is to say a block of only one cell).

```

4440   \tl_if_blank:nTF { #2 } { \@@_Block_i 1-1 \q_stop } { \@@_Block_i #2 \q_stop }
4441   { #1 } { #3 } { #4 }
4442 }

```

With the following construction, we extract the values of  $i$  and  $j$  in the first mandatory argument of the command.

```
4443 \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.

```
4444 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
4445 {
```

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*).

```
4446   \bool_lazy_or:nnTF
4447     { \tl_if_blank_p:n { #1 } }
4448     { \str_if_eq_p:nn { #1 } { * } }
4449     { \int_set:Nn \l_tmpa_int { 100 } }
4450     { \int_set:Nn \l_tmpa_int { #1 } }
4451   \bool_lazy_or:nnTF
4452     { \tl_if_blank_p:n { #2 } }
4453     { \str_if_eq_p:nn { #2 } { * } }
4454     { \int_set:Nn \l_tmpb_int { 100 } }
4455     { \int_set:Nn \l_tmpb_int { #2 } }
4456   \int_compare:nNnTF \l_tmpb_int = 1
4457     {
4458       \tl_if_empty:NTF \l_@@_cell_type_tl
4459         { \tl_set:Nn \l_@@_pos_of_block_tl c }
4460         { \tl_set_eq:NN \l_@@_pos_of_block_tl \l_@@_cell_type_tl }
4461     }
4462     { \tl_set:Nn \l_@@_pos_of_block_tl c }

4463   \keys_set:known:nn { NiceMatrix / Block / FirstPass } { #3 }
4464   \tl_set:Nx \l_tmpa_tl
4465     {
4466     { \int_use:N \c@iRow }
4467     { \int_use:N \c@jCol }
4468     { \int_eval:n { \c@iRow + \l_tmpa_int - 1 } }
4469     { \int_eval:n { \c@jCol + \l_tmpb_int - 1 } }
4470   }
}
```

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).

```
4471   \bool_lazy_or:nnTF
4472     { \int_compare_p:nNn { \l_tmpa_int } = 1 }
4473     { \int_compare_p:nNn { \l_tmpb_int } = 1 }
4474     { \exp_args:Nxx \@@_Block_iv:nnnnn }
4475     { \exp_args:Nxx \@@_Block_v:nnnnn }
4476   { \l_tmpa_int } { \l_tmpb_int } { #3 } { #4 } { #5 }
4477 }
```

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).

```

4478 \cs_new_protected:Npn \l_@@_Block_iv:nnnnn #1 #2 #3 #4 #5
4479 {
4480   \int_gincr:N \g_@@_block_box_int
4481   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4482   {
4483     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4484     {
4485       \@@_actually_diagbox:nnnnnn
4486       { \int_use:N \c@iRow }
4487       { \int_use:N \c@jCol }
4488       { \int_eval:n { \c@iRow + #1 - 1 } }
4489       { \int_eval:n { \c@jCol + #2 - 1 } }
4490       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4491     }
4492   }
4493   \box_gclear_new:c
4494   { g_@@_block_box _ \int_use:N \g_@@_block_box_int _ box }
4495   \hbox_gset:cn
4496   { g_@@_block_box _ \int_use:N \g_@@_block_box_int _ box }
4497   {

```

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.

```

4498   \tl_if_empty:NTF \l_@@_color_tl
4499   { \int_compare:nNnT { #2 } = 1 \set@color }
4500   { \color { \l_@@_color_tl } }
4501   \group_begin:
4502   \cs_set:Npn \arraystretch { 1 }
4503   \dim_set_eq:NN \extrarowheight \c_zero_dim
4504   #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).

```

4505   \bool_if:NT \g_@@_rotate_bool { \tl_set:Nn \l_@@_pos_of_block_tl c }
4506   \bool_if:NNTF \l_@@_NiceTabular_bool
4507   {
4508     \exp_args:Nnx \begin { tabular }
4509     { @ { } \l_@@_pos_of_block_tl @ { } }
4510     #5
4511     \end { tabular }
4512   }
4513   {
4514     \c_math_toggle_token
4515     \exp_args:Nnx \begin { array }
4516     { @ { } \l_@@_pos_of_block_tl @ { } }
4517     #5
4518     \end { array }
4519     \c_math_toggle_token
4520   }
4521   \group_end:
4522 }
4523 \bool_if:NT \g_@@_rotate_bool
4524 {
4525   \box_grotate:cn
4526   { g_@@_block_box _ \int_use:N \g_@@_block_box_int _ box }
4527   { 90 }
4528   \bool_gset_false:N \g_@@_rotate_bool
4529 }

```

If we are in a mono-column block, we take into account the width of that block for the width of the column.

```

4530 \int_compare:nNnT { #2 } = 1
4531 {
4532   \dim_gset:Nn \g_@@_blocks_wd_dim
4533   {
4534     \dim_max:nn
4535     \g_@@_blocks_wd_dim
4536     {
4537       \box_wd:c
4538       { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4539     }
4540   }
4541 }

```

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.

```

4542 \int_compare:nNnT { #1 } = 1
4543 {
4544   \dim_gset:Nn \g_@@_blocks_ht_dim
4545   {
4546     \dim_max:nn
4547     \g_@@_blocks_ht_dim
4548     {
4549       \box_ht:c
4550       { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4551     }
4552   }
4553   \dim_gset:Nn \g_@@_blocks_dp_dim
4554   {
4555     \dim_max:nn
4556     \g_@@_blocks_dp_dim
4557     {
4558       \box_dp:c
4559       { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4560     }
4561   }
4562 }
4563 \seq_gput_right:Nx \g_@@_blocks_seq
4564 {
4565   \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.

```

4566 { \exp_not:n { #3 } , \l_@@_pos_of_block_tl }
4567 {
4568   \box_use_drop:c
4569   { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4570 }
4571 }
4572 }

```

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.

```

4573 \cs_new_protected:Npn \@@_Block_v:nnnnn #1 #2 #3 #4 #5
4574 {
4575   \seq_gput_right:Nx \g_@@_blocks_seq
4576   {
4577     \l_tmpa_tl
4578     { \exp_not:n { #3 } }

```



```

4579 \exp_not:n
4580 {
4581   {
4582     \bool_if:NTF \l_@@_NiceTabular_bool
4583     {
4584       \group_begin:
4585       \cs_set:Npn \arraystretch { 1 }
4586       \dim_set_eq:NN \extrarowheight \c_zero_dim
4587       #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).

```

4588       \bool_if:NT \g_@@_rotate_bool
4589       { \tl_set:Nn \l_@@_pos_of_block_tl c }
4590       \exp_args:Nnx \begin { tabular }
4591       { @ { } \l_@@_pos_of_block_tl @ { } }
4592       #5
4593       \end { tabular }
4594       \group_end:
4595     }
4596   {
4597     \group_begin:
4598     \cs_set:Npn \arraystretch { 1 }
4599     \dim_set_eq:NN \extrarowheight \c_zero_dim
4600     #4
4601     \bool_if:NT \g_@@_rotate_bool
4602     { \tl_set:Nn \l_@@_pos_of_block_tl c }
4603     \c_math_toggle_token
4604     \exp_args:Nnx \begin { array }
4605     { @ { } \l_@@_pos_of_block_tl @ { } } #5 \end { array }
4606     \c_math_toggle_token
4607     \group_end:
4608   }
4609 }
4610 }
4611 }
4612 }

```

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).

```

4613 \keys_define:nn { NiceMatrix / Block / SecondPass }
4614 {
4615   fill .tl_set:N = \l_@@_fill_tl ,
4616   fill .value_required:n = true ,
4617   draw .tl_set:N = \l_@@_draw_tl ,
4618   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.

```

4619   color .code:n =
4620     \@@_error:n { Key~color~for~Block }
4621     \tl_set:Nn \l_@@_fill_tl { #1 } ,
4622   % color .code:n = \color { #1 } \tl_set:Nn \l_@@_draw_tl { #1 } ,
4623   color .value_required:n = true ,
4624   line-width .dim_set:N = \l_@@_line_width_dim ,
4625   line-width .value_required:n = true ,
4626   l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4627   l .value_forbidden:n = true ,

```

```

4628   r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4629   r .value_forbidden:n = true ,
4630   c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4631   c .value_forbidden:n = true ,
4632   unknown .code:n = \@@_error:n { Unknown~key~for~Block }
4633 }

```

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.

```

4634 \cs_new_protected:Npn \@@_draw_blocks:
4635 {
4636   \cs_set_eq:NN \ialign \@@_old_ialign:
4637   \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iv:nnnnnn #1 }
4638 }
4639 \cs_new_protected:Npn \@@_Block_iv:nnnnnn #1 #2 #3 #4 #5 #6
4640 {

```

The integer `\l_@@_last_row_int` will be the last row of the block and `\l_@@_last_col_int` its last column.

```

4641   \int_zero_new:N \l_@@_last_row_int
4642   \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.

```

4643   \int_compare:nNnTF { #3 } > { 99 }
4644   { \int_set_eq:NN \l_@@_last_row_int \c@iRow }
4645   { \int_set:Nn \l_@@_last_row_int { #3 } }
4646   \int_compare:nNnTF { #4 } > { 99 }
4647   { \int_set_eq:NN \l_@@_last_col_int \c@jCol }
4648   { \int_set:Nn \l_@@_last_col_int { #4 } }
4649   \bool_lazy_or:nnTF
4650   { \int_compare_p:nNn \l_@@_last_row_int > \g_@@_row_total_int }
4651   { \int_compare_p:nNn \l_@@_last_col_int > \g_@@_col_total_int }
4652   {
4653     \int_compare:nTF
4654     { \l_@@_last_col_int <= \g_@@_static_num_of_col_int }
4655     {
4656       \msg_error:nnnn { nicematrix } { Block~too~large~2 } { #1 } { #2 }
4657       \@@_msg_redirect_name:nn { Block~too~large~2 } { none }
4658       \group_begin:
4659       \globaldefs = 1
4660       \@@_msg_redirect_name:nn { columns~not~used } { none }
4661       \group_end:
4662     }
4663     { \msg_error:nnnn { nicematrix } { Block~too~large~1 } { #1 } { #2 } }
4664   }
4665   { \@@_Block_v:nnnnnn { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } }
4666 }
4667 \cs_new_protected:Npn \@@_Block_v:nnnnnn #1 #2 #3 #4 #5 #6
4668 {

```

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).

```

4669   \seq_gput_left:Nn \g_@@_pos_of_blocks_seq { { #1 } { #2 } { #3 } { #4 } }

```

The group is for the keys.

```

4670   \group_begin:
4671   \keys_set:nn { NiceMatrix / Block / SecondPass } { #5 }

```

```

4672 \bool_lazy_or:nnT
4673 { ! \tl_if_empty_p:N \l_@@_draw_tl }
4674 { \dim_compare_p:nNn \l_@@_line_width_dim > \c_zero_dim }
4675 {
4676   \tl_gput_right:Nx \g_nicematrix_code_after_tl
4677   {
4678     \@@_stroke_block:nnn
4679     { \exp_not:n { #5 } }
4680     { #1 - #2 }
4681     { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
4682   }
4683   \seq_gput_right:Nn \g_@@_pos_of_stroken_blocks_seq
4684   { { #1 } { #2 } { #3 } { #4 } }
4685 }
4686 \tl_if_empty:NF \l_@@_fill_tl
4687 {
4688   \tl_gput_right:Nx \g_nicematrix_code_before_tl
4689   {
4690     \exp_not:N \rectanglecolor
4691     { \exp_not:V \l_@@_fill_tl }
4692     { #1 - #2 }
4693     { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
4694   }
4695 }
4696 \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4697 {
4698   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4699   {
4700     \@@_actually_diagbox:nnnnnn
4701     { #1 }
4702     { #2 }
4703     { \int_use:N \l_@@_last_row_int }
4704     { \int_use:N \l_@@_last_col_int }
4705     { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4706   }
4707 }
4708 \hbox_set:Nn \l_@@_cell_box { #6 }
4709 \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.

```

4710 \pgfpicture
4711 \pgfrememberpicturepositiononpagetrue
4712 \pgf@relevantforpicturesizefalse
4713 \@@_qpoint:n { row - #1 }
4714 \dim_set_eq:NN \l_tmpa_dim \pgf@y
4715 \@@_qpoint:n { col - #2 }
4716 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4717 \@@_qpoint:n { row - \@@_succ:n { \l_@@_last_row_int } }
4718 \dim_set_eq:NN \l_tmpc_dim \pgf@y
4719 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
4720 \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.

```

4721 \begin { pgfscope }
4722 \@@_pgf_rect_node:nnnnn
4723 { \@@_env: - #1 - #2 - block }
4724 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
4725 \end { pgfscope }

```

We construct the short node.

```

4726 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
4727 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4728 {

```

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.

```

4729 \cs_if_exist:cT
4730 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
4731 {
4732 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4733 {
4734 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
4735 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
4736 }
4737 }
4738 }

```

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.

```

4739 \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
4740 {
4741 \@@_qpoint:n { col - #2 }
4742 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4743 }
4744 \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
4745 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4746 {
4747 \cs_if_exist:cT
4748 { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
4749 {
4750 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4751 {
4752 \pgfpointanchor
4753 { \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
4754 { east }
4755 \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
4756 }
4757 }
4758 }
4759 \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
4760 {
4761 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }

```

```

4762         \dim_set_eq:NN \l_tmpd_dim \pgf@x
4763     }
4764     \@@_pgf_rect_node:nnnnn
4765     { \@@_env: - #1 - #2 - block - short }
4766     \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.

```

4767     \bool_if:NT \l_@@_medium_nodes_bool
4768     {
4769         \@@_pgf_rect_node:nnn
4770         { \@@_env: - #1 - #2 - block - medium }
4771         { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
4772         {
4773             \pgfpointanchor
4774             { \@@_env:
4775                 - \int_use:N \l_@@_last_row_int
4776                 - \int_use:N \l_@@_last_col_int - medium
4777             }
4778             { south-east }
4779         }
4780     }

```

Now, we will put the label of the block beginning with the case of a `\Block` of one row.

```

4781     \int_compare:nNnTF { #1 } = { #3 }
4782     {

```

We take into account the case of a block of one row in the “first row” or the “last row”.

```

4783         \int_compare:nNnTF { #1 } = 0
4784         { \l_@@_code_for_first_row_tl }
4785         {
4786             \int_compare:nNnT { #1 } = \l_@@_last_row_int
4787             { \l_@@_code_for_last_row_tl }
4788         }

```

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`.

```

4789         \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

We retrieve (in `\pgf@x`) the  $x$ -value of the center of the block.

```

4790         \pgfpointanchor
4791         { \@@_env: - #1 - #2 - block - short }
4792         {
4793             \str_case:Vn \l_@@_pos_of_block_tl
4794             {
4795                 c { center }
4796                 l { west }
4797                 r { east }
4798             }
4799         }

```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```

4800         \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
4801         \pgfset { inner-sep = \c_zero_dim }
4802         \pgfnode
4803         { rectangle }
4804         {
4805             \str_case:Vn \l_@@_pos_of_block_tl
4806             {
4807                 c { base }
4808                 l { base~west }
4809                 r { base~east }
4810             }
4811         }

```

```

4812     { \box_use_drop:N \l_@@_cell_box } { } { }
4813 }

```

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`).

```

4814 {

```

If we are in the first column, we must put the block as if it was with the key `r`.

```

4815     \int_compare:nNnT { #2 } = 0
4816     { \tl_set:Nn \l_@@_pos_of_block_tl r }
4817     \bool_if:nT \g_@@_last_col_found_bool
4818     {
4819         \int_compare:nNnT { #2 } = \g_@@_col_total_int
4820         { \tl_set:Nn \l_@@_pos_of_block_tl 1 }
4821     }
4822     \pgftransformshift
4823     {
4824         \pgfpointanchor
4825         { \@@_env: - #1 - #2 - block - short }
4826         {
4827             \str_case:Nn \l_@@_pos_of_block_tl
4828             {
4829                 c { center }
4830                 l { west }
4831                 r { east }
4832             }
4833         }
4834     }
4835     \pgfset { inner~sep = \c_zero_dim }
4836     \pgfnode
4837     { rectangle }
4838     {
4839         \str_case:Nn \l_@@_pos_of_block_tl
4840         {
4841             c { center }
4842             l { west }
4843             r { east }
4844         }
4845     }
4846     { \box_use_drop:N \l_@@_cell_box } { } { }
4847 }
4848 \endpgfpicture
4849 \group_end:
4850 }

```

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).

```

4851 \cs_new_protected:Npn \@@_stroke_block:nnn #1 #2 #3
4852 {
4853     \group_begin:
4854     \tl_clear:N \l_@@_draw_tl
4855     \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
4856     \keys_set_known:nn { NiceMatrix / BlockStroke } { #1 }
4857     \pgfpicture
4858     \pgfrememberpicturerepositiononpagetrue
4859     \pgf@relevantforpicturesizefalse
4860     \tl_if_empty:NF \l_@@_draw_tl
4861     {

```

If the user has used the key `color` of the command `\Block` without value, the color fixed by `\arrayrulecolor` is used.

```

4862         \str_if_eq:NnTF \l_@@_draw_tl { default }

```

```

4863         { \CT@arc@ }
4864         { \pgfsetstrokecolor { \l_@@_draw_tl } }
4865     }
4866     \@@_cut_on_hyphen:w #2 \q_stop
4867     \bool_lazy_and:nnT
4868     { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
4869     { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
4870     {
4871         \@@_qpoint:n { row - \l_tmpa_tl }
4872         \dim_set:Nn \l_tmpb_dim { \pgf@y }
4873         \@@_qpoint:n { col - \l_tmpb_tl }
4874         \dim_set:Nn \l_tmpc_dim { \pgf@x }
4875         \@@_cut_on_hyphen:w #3 \q_stop
4876         \int_compare:nNnT \l_tmpa_tl > \c@iRow
4877         { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
4878         \int_compare:nNnT \l_tmpb_tl > \c@jCol
4879         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
4880         \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
4881         \dim_set:Nn \l_tmpa_dim { \pgf@y }
4882         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
4883         \dim_set:Nn \l_tmpd_dim { \pgf@x }
4884         \pgfpathrectanglecorners
4885         { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4886         { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
4887         \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }
4888         \pgfusepathqstroke
4889     }
4890     \endpgfpicture
4891     \group_end:
4892 }

```

Here is the set of keys for the command `\@@_stroke_block:nnn`.

```

4893 \keys_define:nn { NiceMatrix / BlockStroke }
4894 {

```

We will uncomment the following line when we will give to the key `color` of the command `\Block` its new definition.

```

4895     % color .tl_set:N = \l_@@_draw_tl ,
4896     draw .tl_set:N = \l_@@_draw_tl ,
4897     draw .default:n = default ,
4898     line-width .dim_set:N = \l_@@_line_width_dim ,
4899 }

```

## How to draw the dotted lines transparently

```

4900 \cs_set_protected:Npn \@@_renew_matrix:
4901 {
4902     \RenewDocumentEnvironment { pmatrix } { } {
4903         { \pNiceMatrix }
4904         { \endpNiceMatrix }
4905     }
4906     \RenewDocumentEnvironment { vmatrix } { } {
4907         { \vNiceMatrix }
4908         { \endvNiceMatrix }
4909     }
4910     \RenewDocumentEnvironment { Vmatrix } { } {
4911         { \VNiceMatrix }
4912         { \endVNiceMatrix }
4913     }
4914     \RenewDocumentEnvironment { bmatrix } { } {
4915         { \bNiceMatrix }
4916         { \endbNiceMatrix }
4917     }
4918     \RenewDocumentEnvironment { Bmatrix } { } {
4919         { \BNiceMatrix }
4920         { \endBNiceMatrix }
4921     }

```

```
4917 }
```

## Automatic arrays

```
4918 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
4919 {
4920   \int_set:Nn \l_@@_nb_rows_int { #1 }
4921   \int_set:Nn \l_@@_nb_cols_int { #2 }
4922 }

4923 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
4924 {
4925   \int_zero_new:N \l_@@_nb_rows_int
4926   \int_zero_new:N \l_@@_nb_cols_int
4927   \@@_set_size:n #4 \q_stop
4928   \begin { NiceArrayWithDelims } { #1 } { #2 }
4929     { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
4930   \int_compare:nNnT \l_@@_first_row_int = 0
4931     {
4932       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4933       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4934       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4935     }
4936   \prg_replicate:nn \l_@@_nb_rows_int
4937     {
4938       \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`).

```
4939   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
4940   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4941 }
4942 \int_compare:nNnT \l_@@_last_row_int > { -2 }
4943 {
4944   \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4945   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4946   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4947 }
4948 \end { NiceArrayWithDelims }
4949 }

4950 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
4951 {
4952   \cs_set_protected:cpn { #1 AutoNiceMatrix }
4953   {
4954     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
4955     \AutoNiceMatrixWithDelims { #2 } { #3 }
4956   }
4957 }

4958 \@@_define_com:nnn p ( )
4959 \@@_define_com:nnn b [ ]
4960 \@@_define_com:nnn v | |
4961 \@@_define_com:nnn V \ | \ |
4962 \@@_define_com:nnn B \{ \}
```

We define also an command `\AutoNiceMatrix` similar to the environment `{NiceMatrix}`.

```
4963 \NewDocumentCommand \AutoNiceMatrix { O { } m O { } m ! O { } }
4964 {
4965   \group_begin:
4966     \bool_set_true:N \l_@@_NiceArray_bool
4967     \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
4968   \group_end:
4969 }
```



## The redefinition of the command `\dotfill`

```
4970 \cs_set_eq:NN \@@_old_dotfill \dotfill
4971 \cs_new_protected:Npn \@@_dotfill:
4972 {
```

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}`).

```
4973   \@@_old_dotfill
4974   \bool_if:NT \l_@@_NiceTabular_bool
4975     { \group_insert_after:N \@@_dotfill_ii: }
4976     { \group_insert_after:N \@@_dotfill_i: }
4977   }
4978 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
4979 \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 its 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`.

```
4980 \cs_new_protected:Npn \@@_dotfill_iii:
4981 { \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`.

```
4982 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
4983 {
4984   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4985   {
4986     \@@_actually_diagbox:nnnnnn
4987     { \int_use:N \c@iRow }
4988     { \int_use:N \c@jCol }
4989     { \int_use:N \c@iRow }
4990     { \int_use:N \c@jCol }
4991     { \exp_not:n { #1 } }
4992     { \exp_not:n { #2 } }
4993   }
```

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`.

```
4994   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
4995   {
4996     { \int_use:N \c@iRow }
4997     { \int_use:N \c@jCol }
4998     { \int_use:N \c@iRow }
4999     { \int_use:N \c@jCol }
5000   }
5001 }
```

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.

```
5002 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
5003 {
5004   \pgfpicture
5005   \pgf@relevantforpicturesizefalse
5006   \pgfrememberpicturepositiononpagetrue
5007   \@@_qpoint:n { row - #1 }
5008   \dim_set_eq:NN \l_tmpa_dim \pgf@y
5009   \@@_qpoint:n { col - #2 }
5010   \dim_set_eq:NN \l_tmpb_dim \pgf@x
5011   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
```

```

5012 \@@_qpoint:n { row - \@@_succ:n { #3 } }
5013 \dim_set_eq:NN \l_tmpc_dim \pgf@y
5014 \@@_qpoint:n { col - \@@_succ:n { #4 } }
5015 \dim_set_eq:NN \l_tmpd_dim \pgf@x
5016 \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
5017 {

```

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.

```

5018 \CT@arc@
5019 \pgfsetroundcap
5020 \pgfusepathqstroke
5021 }
5022 \pgfset { inner~sep = 1 pt }
5023 \pgfscope
5024 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
5025 \pgfnode { rectangle } { south-west }
5026 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
5027 \endpgfscope
5028 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
5029 \pgfnode { rectangle } { north-east }
5030 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
5031 \endpgfpicture
5032 }

```

## The keyword `\CodeAfter`

The `code-after` (inserted with the key `code-after` or after the keyword `\CodeAfter` can always begin with a list of pairs *key-value* between square brackets). Here the corresponding set of keys.

```

5033 \keys_define:nn { NiceMatrix / CodeAfter }
5034 {
5035   sub-matrix .code:n =
5036   \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
5037   sub-matrix .value_required:n = true ,
5038   delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
5039   delimiters-color .value_required:n = true ,
5040   unknown .code:n = \@@_error:n { Unknown-key-for-CodeAfter }
5041 }

```

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`.

```

5042 \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`.

```

5043 \cs_new_protected:Npn \@@_CodeAfter_i:n #1 \end
5044 {
5045   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
5046   \@@_CodeAfter_ii:n
5047 }

```

We catch the argument of the command `\end` (in `#1`).

```

5048 \cs_new_protected:Npn \@@_CodeAfter_ii:n #1
5049 {

```

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.

```
5050 \str_if_eq:eeTF \currentenvir { #1 }
5051 { \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`.

```
5052 {
5053   \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
5054   \@@_CodeAfter_i:n
5055 }
5056 }
```

## 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 columnn. 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.

```
5057 \cs_new_protected:Npn \@@_delimiter:nnn #1 #2 #3
5058 {
5059   \pgfpicture
5060   \pgfrememberpicturerepositiononpagetrue
5061   \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.

```
5062 \@@_qpoint:n { row - 1 }
5063 \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
5064 \@@_qpoint:n { row - \@@_succ:n \c@iRow }
5065 \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).

```
5066 \bool_if:nTF { #3 }
5067 { \dim_set_eq:NN \l_tmpa_dim \c_max_dim }
5068 { \dim_set:Nn \l_tmpa_dim { - \c_max_dim } }
5069 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5070 {
5071   \cs_if_exist:cT
5072   { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
5073   {
5074     \pgfpictureanchor
5075     { \@@_env: - ##1 - #2 }
5076     { \bool_if:nTF { #3 } { west } { east } }
5077     \dim_set:Nn \l_tmpa_dim
5078     { \bool_if:nTF { #3 } \dim_min:nn \dim_max:nn \l_tmpa_dim \pgf@x }
5079   }
5080 }
```

Now we can put the delimiter with a node of PGF.

```
5081 \pgfset { inner~sep = \c_zero_dim }
5082 \dim_zero:N \nulldelimiterspace
5083 \pgftransformshift
5084 {
```

```

5085     \pgfpoint
5086     { \l_tmpa_dim }
5087     { ( \l_@@_y_initial_dim + \l_@@_y_final_dim + \arrayrulewidth ) / 2 }
5088   }
5089   \pgfnode
5090   { rectangle }
5091   { \bool_if:nTF { #3 } { east } { west } }
5092   {

```

Here is the content of the PGF node, that is to say the delimiter, constructed with its right size.

```

5093     \nullfont
5094     \c_math_toggle_token
5095     \tl_if_empty:NF \l_@@_delimiters_color_tl
5096     { \color { \l_@@_delimiters_color_tl } }
5097     \bool_if:nTF { #3 } { \left #1 } { \left . }
5098     \vcenter
5099     {
5100         \nullfont
5101         \hrule \@height
5102             \dim_eval:n { \l_@@_y_initial_dim - \l_@@_y_final_dim }
5103             \@depth \c_zero_dim
5104             \@width \c_zero_dim
5105     }
5106     \bool_if:nTF { #3 } { \right . } { \right #1 }
5107     \c_math_toggle_token
5108   }
5109   { }
5110   { }
5111 \endpgfpicture
5112 }

```

## The command SubMatrix

```

5113 \keys_define:nn { NiceMatrix / sub-matrix }
5114 {
5115     extra-height .dim_set:N = \l_@@_submatrix_extra_height_dim ,
5116     extra-height .value_required:n = true ,
5117     left-xshift .dim_set:N = \l_@@_submatrix_left_xshift_dim ,
5118     left-xshift .value_required:n = true ,
5119     right-xshift .dim_set:N = \l_@@_submatrix_right_xshift_dim ,
5120     right-xshift .value_required:n = true ,
5121     xshift .meta:n = { left-xshift = #1, right-xshift = #1 } ,
5122     xshift .value_required:n = true ,
5123     delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
5124     delimiters-color .value_required:n = true ,
5125     slim .bool_set:N = \l_@@_submatrix_slim_bool ,
5126     slim .default:n = true ,
5127 }
5128 \keys_define:nn { NiceMatrix }
5129 {
5130     SubMatrix .inherit:n = NiceMatrix /sub-matrix ,
5131     CodeAfter / sub-matrix .inherit:n = NiceMatrix / sub-matrix
5132 }

```

The following keys set is for the command `\SubMatrix` itself (not the tuning of `\SubMatrix` that can be done elsewhere).

```

5133 \keys_define:nn { NiceMatrix / SubMatrix }
5134 {
5135     name .code:n =
5136         \tl_if_empty:nTF { #1 }
5137         { \@@_error:n { Invalid-name-format } }
5138         {
5139             \regex_match:nnTF { \A[A-Za-z][A-Za-z0-9]*\Z } { #1 }

```

```

5140     {
5141       \seq_if_in:NnTF \g_@@_submatrix_names_seq { #1 }
5142       { \@@_error:nn { Duplicate-name-for-SubMatrix } { #1 } }
5143       {
5144         \str_set:Nn \l_@@_submatrix_name_str { #1 }
5145         \seq_gput_right:Nn \g_@@_submatrix_names_seq { #1 }
5146       }
5147     }
5148     { \@@_error:n { Invalid-name-format } }
5149   } ,
5150   name .value_required:n = true ,
5151   unknown .code:n = \@@_error:n { Unknown-key-for-SubMatrix }
5152 }

```

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  $i-j$ ;
- #2 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.

```

5153 \NewDocumentCommand \@@_SubMatrix { m m m m ! O { } }
5154 {
5155   \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`.

```

5156   \@@_cut_on_hyphen:w #3 \q_stop
5157   \tl_clear_new:N \l_tmpc_tl
5158   \tl_clear_new:N \l_tmpd_tl
5159   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
5160   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
5161   \@@_cut_on_hyphen:w #2 \q_stop
5162   \bool_lazy_or:nnTF
5163     { \int_compare_p:nNn \l_tmpc_tl > \g_@@_row_total_int }
5164     { \int_compare_p:nNn \l_tmpd_tl > \g_@@_col_total_int }
5165     { \@@_error:n { SubMatrix-too-large } }
5166   {
5167     \dim_zero_new:N \l_@@_submatrix_extra_height_dim
5168     \dim_zero_new:N \l_@@_submatrix_left_xshift_dim
5169     \dim_zero_new:N \l_@@_submatrix_right_xshift_dim
5170     \str_clear_new:N \l_@@_submatrix_name_str
5171     \keys_set:nn { NiceMatrix / SubMatrix } { #5 }
5172     \pgfpicture
5173     \pgfrememberpicturepositiononpagetrue
5174     \pgf@relevantforpicturesizefalse
5175     \pgfset { inner-sep = \c_zero_dim }
5176     \dim_set_eq:NN \l_@@_x_initial_dim \c_max_dim
5177     \dim_set:Nn \l_@@_x_final_dim { - \c_max_dim }

```

The last value of `\int_step_inline:nnn` is provided by currying.

```

5178   \bool_if:NTF \l_@@_submatrix_slim_bool
5179   { \int_step_inline:nnn \l_tmpa_tl \l_tmpc_tl }
5180   { \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int }
5181   {
5182     \cs_if_exist:cT
5183       { pgf @ sh @ ns @ \@@_env: - ##1 - \l_tmpb_tl }
5184     {
5185       \pgfpointanchor { \@@_env: - ##1 - \l_tmpb_tl } { west }
5186       \dim_set:Nn \l_@@_x_initial_dim
5187         { \dim_min:nn \l_@@_x_initial_dim \pgf@x }

```

```

5188     }
5189     \cs_if_exist:cT
5190     { pgf @ sh @ ns @ \@@_env: - ##1 - \l_tmpd_tl }
5191     {
5192         \pgfpointanchor { \@@_env: - ##1 - \l_tmpd_tl } { east }
5193         \dim_set:Nn \l_@@_x_final_dim
5194         { \dim_max:nn \l_@@_x_final_dim \pgf@x }
5195     }
5196 }
5197 \@@_qpoint:n { row - \l_tmpa_tl - base }
5198 \dim_set:Nn \l_@@_y_initial_dim
5199 { \pgf@y + ( \box_ht:N \strutbox + \extrarowheight ) * \arraystretch }
5200 \@@_qpoint:n { row - \l_tmpc_tl - base }
5201 \dim_set:Nn \l_@@_y_final_dim
5202 { \pgf@y - ( \box_dp:N \strutbox ) * \arraystretch }
5203 \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
5204 {
5205     \cs_if_exist:cT
5206     { pgf @ sh @ ns @ \@@_env: - \l_tmpa_tl - ##1 }
5207     {
5208         \pgfpointanchor { \@@_env: - \l_tmpa_tl - ##1 } { north }
5209         \dim_set:Nn \l_@@_y_initial_dim
5210         { \dim_max:nn \l_@@_y_initial_dim \pgf@y }
5211     }
5212     \cs_if_exist:cT
5213     { pgf @ sh @ ns @ \@@_env: - \l_tmpc_tl - ##1 }
5214     {
5215         \pgfpointanchor { \@@_env: - \l_tmpc_tl - ##1 } { south }
5216         \dim_set:Nn \l_@@_y_final_dim
5217         { \dim_min:nn \l_@@_y_final_dim \pgf@y }
5218     }
5219 }
5220 \dim_set:Nn \l_tmpa_dim
5221 {
5222     \l_@@_y_initial_dim - \l_@@_y_final_dim +
5223     \l_@@_submatrix_extra_height_dim - \arrayrulewidth
5224 }
5225 \dim_set_eq:NN \nulldelimiterspace \c_zero_dim

```

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).

```

5226     \str_if_empty:NF \l_@@_submatrix_name_str
5227     {
5228         \@@_pgf_rect_node:nnnnn \l_@@_submatrix_name_str
5229         \l_@@_x_initial_dim \l_@@_y_initial_dim
5230         \l_@@_x_final_dim \l_@@_y_final_dim
5231     }

```

Now, we deal with the left delimiter. Of course, the environment `{pgfscope}` is for the `\pgftransformshift`.

```

5232     \begin { pgfscope }
5233     \pgftransformshift
5234     {
5235         \pgfpoint
5236         { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
5237         { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
5238     }
5239     \str_if_empty:NTF \l_@@_submatrix_name_str
5240     { \@@_node_left:nn #1 { } }
5241     { \@@_node_left:nn #1 { \@@_env: - \l_@@_submatrix_name_str - left } }
5242     \end { pgfscope }

```

Now, we deal with the right delimiter.

```

5243     \pgftransformshift
5244     {
5245         \pgfpoint
5246         { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
5247         { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
5248     }
5249     \str_if_empty:NTF \l_@@_submatrix_name_str
5250     { \@@_node_right:nn #4 { } }
5251     {
5252         \@@_node_right:nn #4 { \@@_env: - \l_@@_submatrix_name_str - right }
5253     }
5254     \endpgfpicture
5255 }
5256 \group_end:
5257 }

```

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`).

```

5258 \cs_new_protected:Npn \@@_node_left:nn #1 #2
5259 {
5260     \pgfnode
5261     { rectangle }
5262     { east }
5263     {
5264         \nullfont
5265         \c_math_toggle_token
5266         \tl_if_empty:NF \l_@@_delimiters_color_tl
5267         { \color { \l_@@_delimiters_color_tl } }
5268         \left #1
5269         \vcenter
5270         {
5271             \nullfont
5272             \hrule \@height \l_tmpa_dim
5273             \c_zero_dim
5274             \@width \c_zero_dim
5275         }
5276         \right .
5277         \c_math_toggle_token
5278     }
5279     { #2 }
5280     { }
5281 }

```

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`).

```

5282 \cs_new_protected:Npn \@@_node_right:nn #1 #2
5283 {
5284     \pgfnode
5285     { rectangle }
5286     { west }
5287     {
5288         \nullfont
5289         \c_math_toggle_token
5290         \tl_if_empty:NF \l_@@_delimiters_color_tl
5291         { \color { \l_@@_delimiters_color_tl } }
5292         \left .
5293         \vcenter
5294         {
5295             \nullfont
5296             \hrule \@height \l_tmpa_dim
5297             \c_zero_dim

```

```

5298             \@width \c_zero_dim
5299         }
5300         \right #1
5301         \c_math_toggle_token
5302     }
5303     { #2 }
5304     { }
5305 }

```

## 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.

```

5306 \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.

```

5307 \bool_new:N \c_@@_footnote_bool

5308 \@@_msg_new:nnn { Unknown~option~for~package }
5309 {
5310   The~key~'\l_keys_key_str'~is~unknown.  \
5311   If~you~go~on,~it~will~be~ignored.  \
5312   For~a~list~of~the~available~keys,~type~H~<return>.
5313 }
5314 {
5315   The~available~keys~are~(in~alphabetic~order):~
5316   define-L-C-R,~
5317   footnote,~
5318   footnotehyper,~
5319   renew-dots,~and
5320   renew-matrix.
5321 }

5322 \@@_msg_new:nn { Key~transparent }
5323 {
5324   The~key~'transparent'~is~now~obsolete~(because~it's~name~
5325   is~not~clear).~You~should~use~the~conjunction~of~'renew-dots'~
5326   and~'renew-matrix'.~However,~you~can~go~on.
5327 }

5328 \keys_define:nn { NiceMatrix / Package }
5329 {
5330   define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
5331   define-L-C-R .default:n = true ,
5332   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
5333   renew-dots .value_forbidden:n = true ,
5334   renew-matrix .code:n = \@@_renew_matrix: ,
5335   renew-matrix .value_forbidden:n = true ,
5336   transparent .code:n =
5337   {
5338     \@@_renew_matrix:
5339     \bool_set_true:N \l_@@_renew_dots_bool
5340     \@@_error:n { Key~transparent }
5341   } ,
5342   transparent .value_forbidden:n = true,
5343   footnote .bool_set:N = \c_@@_footnote_bool ,

```



```

5344     footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
5345     unknown .code:n = \@@_error:n { Unknown-option-for-package }
5346   }
5347   \ProcessKeysOptions { NiceMatrix / Package }

5348   \@@_msg_new:nn { footnote-with-footnotehyper-package }
5349   {
5350     You~can't~use~the~option~'footnote'~because~the~package~
5351     footnotehyper~has~already~been~loaded.~
5352     If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
5353     within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
5354     of~the~package~footnotehyper.\\
5355     If~you~go~on,~the~package~footnote~won't~be~loaded.
5356   }

5357   \@@_msg_new:nn { footnotehyper-with-footnote-package }
5358   {
5359     You~can't~use~the~option~'footnotehyper'~because~the~package~
5360     footnote~has~already~been~loaded.~
5361     If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
5362     within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
5363     of~the~package~footnote.\\
5364     If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
5365   }

5366   \bool_if:NT \c_@@_footnote_bool
5367   {

```

The class `beamer` has its own system to extract footnotes and that's why we have nothing to do if `beamer` is used.

```

5368     \@ifclassloaded { beamer }
5369     { \bool_set_false:N \c_@@_footnote_bool }
5370     {
5371       \@ifpackageloaded { footnotehyper }
5372       { \@@_error:n { footnote-with-footnotehyper-package } }
5373       { \usepackage { footnote } }
5374     }
5375   }

5376   \bool_if:NT \c_@@_footnotehyper_bool
5377   {

```

The class `beamer` has its own system to extract footnotes and that's why we have nothing to do if `beamer` is used.

```

5378     \@ifclassloaded { beamer }
5379     { \bool_set_false:N \c_@@_footnote_bool }
5380     {
5381       \@ifpackageloaded { footnote }
5382       { \@@_error:n { footnotehyper-with-footnote-package } }
5383       { \usepackage { footnotehyper } }
5384     }
5385     \bool_set_true:N \c_@@_footnote_bool
5386   }

```

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

```

5387 \seq_new:N \c_@@_types_of_matrix_seq
5388 \seq_set_from_clist:Nn \c_@@_types_of_matrix_seq
5389 {
5390   NiceMatrix ,
5391   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
5392 }
5393 \seq_set_map_x:NNn \c_@@_types_of_matrix_seq \c_@@_types_of_matrix_seq
5394 { \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`.

```

5395 \cs_new_protected:Npn \@@_error_too_much_cols:
5396 {
5397   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
5398   {
5399     \int_compare:nNnTF \l_@@_last_col_int = { -2 }
5400     { \@@_fatal:n { too-much-cols-for-matrix } }
5401     {
5402       \bool_if:NF \l_@@_last_col_without_value_bool
5403       { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
5404     }
5405   }
5406   { \@@_fatal:n { too-much-cols-for-array } }
5407 }

```

The following command must *not* be protected since it's used in an error message.

```

5408 \cs_new:Npn \@@_message_hdotsfor:
5409 {
5410   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
5411   { ~Maybe-your-use-of~\token_to_str:N \Hdotsfor\ is-incorrect.}
5412 }
5413 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
5414 {
5415   You-try-to-use-more-columns-than-allowed-by-your~
5416   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of~
5417   columns-is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus-the~
5418   exterior-columns).~This-error-is-fatal.
5419 }
5420 \@@_msg_new:nn { too-much-cols-for-matrix }
5421 {
5422   You-try-to-use-more-columns-than-allowed-by-your~
5423   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall-that-the-maximal~
5424   number-of-columns-for-a-matrix-is-fixed-by-the-LaTeX-counter~
5425   'MaxMatrixCols'.~Its-actual-value-is~\int_use:N \c@MaxMatrixCols.~
5426   This-error-is-fatal.
5427 }

```

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`.

```

5428 \@@_msg_new:nn { too-much-cols-for-array }
5429 {
5430   You-try-to-use-more-columns-than-allowed-by-your~
5431   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of-columns-is~
5432   \int_use:N \g_@@_static_num_of_col_int\
5433   ~(plus-the-potential-exterior-ones).~
5434   This-error-is-fatal.
5435 }
5436 \@@_msg_new:nn { last-col-not-used }
5437 {

```

```

5438 The~key~'last-col'~is~in~force~but~you~have~not~used~that~last~column~
5439 in~your~\@@_full_name_env:~.~However,~you~can~go~on.
5440 }

5441 \@@_msg_new:nn { columns~not~used }
5442 {
5443   The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
5444   \g_@@_static_num_of_col_int\ columns~but~you~use~only~\int_use:N \c@jCol.\
5445   You~can~go~on~but~the~columns~you~did~not~used~won't~be~created.
5446 }

5447 \@@_msg_new:nn { in~first~col }
5448 {
5449   You~can't~use~the~command~#1 in~the~first~column~(number~0)~of~the~array.\
5450   If~you~go~on,~this~command~will~be~ignored.
5451 }

5452 \@@_msg_new:nn { in~last~col }
5453 {
5454   You~can't~use~the~command~#1 in~the~last~column~(exterior)~of~the~array.\
5455   If~you~go~on,~this~command~will~be~ignored.
5456 }

5457 \@@_msg_new:nn { in~first~row }
5458 {
5459   You~can't~use~the~command~#1 in~the~first~row~(number~0)~of~the~array.\
5460   If~you~go~on,~this~command~will~be~ignored.
5461 }

5462 \@@_msg_new:nn { in~last~row }
5463 {
5464   You~can't~use~the~command~#1 in~the~last~row~(exterior)~of~the~array.\
5465   If~you~go~on,~this~command~will~be~ignored.
5466 }

5467 \@@_msg_new:nn { bad~option~for~line~style }
5468 {
5469   Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line~style'~
5470   is~'standard'.~If~you~go~on,~this~key~will~be~ignored.
5471 }

5472 \@@_msg_new:nn { Unknown~option~for~xdots }
5473 {
5474   As~for~now~there~is~only~three~key~available~here:~'color',~'line~style'~
5475   and~'shorten'~(and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
5476   this~key~will~be~ignored.
5477 }

5478 \@@_msg_new:nn { Unknown~option~for~rowcolors }
5479 {
5480   As~for~now~there~is~only~two~keys~available~here:~'cols'~and~'respect~blocks'~
5481   (and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
5482   this~key~will~be~ignored.
5483 }

5484 \@@_msg_new:nn { Key~color~for~Block }
5485 {
5486   The~key~'color'~for~the~command~\token_to_str:N \Block\
5487   is~deprecated:~you~should~use~'fill'~instead.\
5488   You~can~go~on~for~this~time~but~remember~that~that~key~
5489   will~be~deleted~in~a~future~version.
5490 }

5491 \@@_msg_new:nn { ampersand~in~light~syntax }
5492 {
5493   You~can't~use~an~ampersand~(\token_to_str:N &)~to~separate~columns~because~
5494   ~you~have~used~the~key~'light~syntax'.~This~error~is~fatal.
5495 }

5496 \@@_msg_new:nn { SubMatrix~too~large }

```

```

5497 {
5498     Your~command~\token_to_str:N \SubMatrix\
5499     can't-be-drawn-because-your-matrix-is-too-small.\
5500     If-you-go-on,~this-command-will-be-ignored.
5501 }
5502 \@@_msg_new:nn { double-backslash-in-light-syntax }
5503 {
5504     You~can't-use~\token_to_str:N \~to~separate~rows~because~you~have~used~
5505     the~key~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
5506     (set-by-the-key~'end-of-row').~This~error~is~fatal.
5507 }
5508 \@@_msg_new:nn { standard-cline-in-document }
5509 {
5510     The~key~'standard-cline'~is~available~only~in~the~preamble.\
5511     If-you-go-on~this~command~will~be~ignored.
5512 }
5513 \@@_msg_new:nn { old-column-type }
5514 {
5515     The~column~type~'#1'~is~no~longer~defined~in~'nicematrix'.~
5516     Since~version~5.0,~you~have~to~use~'l',~'c'~and~'r'~instead~of~'L',~
5517     'C'~and~'R'.~You~can~also~use~the~key~'define-L-C-R'.\
5518     This~error~is~fatal.
5519 }
5520 \@@_msg_new:nn { bad-value-for-baseline }
5521 {
5522     The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
5523     valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
5524     \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\
5525     If-you-go-on,~a~value~of~1~will~be~used.
5526 }
5527 \@@_msg_new:nn { Invalid-name-format }
5528 {
5529     You~can't~give~the~name~'\l_keys_value_tl'~to~a~\token_to_str:N
5530     \SubMatrix.\
5531     A~name~must~be~accepted~by~the~regular~expression~[A-Za-z][A-Za-z0-9]*.\
5532     If-you-go-on,~this~key~will~be~ignored.
5533 }
5534 \@@_msg_new:nn { empty-environment }
5535 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }
5536 \@@_msg_new:nn { Delimiter-with-small }
5537 {
5538     You~can't~put~a~delimiter~in~the~preamble~of~your~\@@_full_name_env:\
5539     because~the~key~'small'~is~in~force.\
5540     This~error~is~fatal.
5541 }
5542 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
5543 {
5544     Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code-after'~
5545     can't~be~executed~because~a~cell~doesn't~exist.\
5546     If-you-go-on~this~command~will~be~ignored.
5547 }
5548 \@@_msg_new:nnn { Duplicate-name-for-SubMatrix }
5549 {
5550     The~name~'#1'~is~already~used~for~a~\token_to_str:N \SubMatrix\
5551     in~this~\@@_full_name_env:.\
5552     If-you-go-on,~this~key~will~be~ignored.\
5553     For~a~list~of~the~names~already~used,~type~H~<return>.
5554 }
5555 {
5556     The~names~already~defined~in~this~\@@_full_name_env:\ are:~

```

```

5557     \seq_use:Nnnn \g_@@_submatrix_names_seq { ~and~ } { ,~ } { ~and~ }.
5558 }

5559 \@@_msg_new:nn { Hdotsfor~in~col~0 }
5560 {
5561     You~can't~use~\token_to_str:N \Hdotsfor\ in~an~exterior~column~of~
5562     the~array.~This~error~is~fatal.
5563 }

5564 \@@_msg_new:nn { bad~corner }
5565 {
5566     #1~is~an~incorrect~specification~for~a~corner~(in~the~keys~
5567     'except~corners'~and~'hvlines~except~corners').~The~available~
5568     values~are:~NW,~SW,~NE~and~SE.\\
5569     If~you~go~on,~this~specification~of~corner~will~be~ignored.
5570 }

5571 \@@_msg_new:nn { last~col~non~empty~for~NiceArray }
5572 {
5573     In~the~\@@_full_name_env:,~you~must~use~the~key~
5574     'last~col'~without~value.\\
5575     However,~you~can~go~on~for~this~time~
5576     (the~value~'\l_keys_value_tl'~will~be~ignored).
5577 }

5578 \@@_msg_new:nn { last~col~non~empty~for~NiceMatrixOptions }
5579 {
5580     In~\NiceMatrixoptions,~you~must~use~the~key~
5581     'last~col'~without~value.\\
5582     However,~you~can~go~on~for~this~time~
5583     (the~value~'\l_keys_value_tl'~will~be~ignored).
5584 }

5585 \@@_msg_new:nn { Block~too~large~1 }
5586 {
5587     You~try~to~draw~a~block~in~the~cell~#1-#2~of~your~matrix~but~the~matrix~is~
5588     too~small~for~that~block. \\
5589 }

5590 \@@_msg_new:nn { Block~too~large~2 }
5591 {
5592     The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
5593     \g_@@_static_num_of_col_int\
5594     columns~but~you~use~only~\int_use:N \c@jCol\ and~that's~why~a~block~
5595     specified~in~the~cell~#1-#2~can't~be~drawn.~You~should~add~some~ampersands~
5596     (&)~at~the~end~of~the~first~row~of~your~
5597     \@@_full_name_env:.\\
5598     If~you~go~on,~this~block~and~maybe~others~will~be~ignored.
5599 }

5600 \@@_msg_new:nn { unknown~column~type }
5601 {
5602     The~column~type~'#1'~in~your~\@@_full_name_env:\
5603     is~unknown. \\
5604     This~error~is~fatal.
5605 }

5606 \@@_msg_new:nn { tabularnote~forbidden }
5607 {
5608     You~can't~use~the~command~\token_to_str:N\tabularnote\
5609     ~in~a~\@@_full_name_env:.~This~command~is~available~only~in~
5610     \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
5611     If~you~go~on,~this~command~will~be~ignored.
5612 }

5613 \@@_msg_new:nn { bottomrule~without~booktabs }
5614 {
5615     You~can't~use~the~key~'tabular/bottomrule'~because~you~haven't~
5616     loaded~'booktabs'.\\

```

```

5617     If~you~go~on,~this~key~will~be~ignored.
5618 }
5619 \@@_msg_new:nn { enumitem~not~loaded }
5620 {
5621     You~can't~use~the~command~\token_to_str:N\tabularnote\
5622     ~because~you~haven't~loaded~'enumitem'.\\
5623     If~you~go~on,~this~command~will~be~ignored.
5624 }
5625 \@@_msg_new:nn { Wrong~last~row }
5626 {
5627     You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
5628     \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
5629     If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
5630     last~row.~You~can~avoid~this~problem~by~using~'last-row'~
5631     without~value~(more~compilations~might~be~necessary).
5632 }
5633 \@@_msg_new:nn { Yet~in~env }
5634 { Environments~of~nicematrix~can't~be~nested.\\ This~error~is~fatal. }
5635 \@@_msg_new:nn { Outside~math~mode }
5636 {
5637     The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
5638     (and~not~in~\token_to_str:N \vcenter).\\
5639     This~error~is~fatal.
5640 }
5641 \@@_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
5642 {
5643     The~value~of~key~'\l_keys_key_str'~must~be~of~length~1.\\
5644     If~you~go~on,~it~will~be~ignored.
5645 }
5646 \@@_msg_new:nnn { Unknown~key~for~Block }
5647 {
5648     The~key~'\l_keys_key_str'~is~unknown~for~the~command~\token_to_str:N
5649     \Block.\\ If~you~go~on,~it~will~be~ignored. \\
5650     For~a~list~of~the~available~keys,~type~H~<return>.
5651 }
5652 {
5653     The~available~keys~are~(in~alphabetic~order):~,~,~c,~draw,~fill,~l,~
5654     line~width~and~r.
5655 }
5656 \@@_msg_new:nnn { Unknown~key~for~CodeAfter }
5657 {
5658     The~key~'\l_keys_key_str'~is~unknown.\\
5659     If~you~go~on,~it~will~be~ignored. \\
5660     For~a~list~of~the~available~keys~in~\token_to_str:N
5661     \CodeAfter,~type~H~<return>.
5662 }
5663 {
5664     The~available~keys~are~(in~alphabetic~order):~
5665     delimiters~color,~
5666     sub~matrix~(several~subkeys)~
5667     and~xdots~(several~subkeys).~
5668     The~latter~is~for~the~command~\token_to_str:N \line.
5669 }
5670 \@@_msg_new:nn { Unknown~key~for~SubMatrix }
5671 {
5672     The~key~'\l_keys_key_str'~is~unknown.\\
5673     The~only~keys~available~here~are~'delimiters~color',~'extra~height',~
5674     'left~xshift',~'name',~'right~xshift',~'slim'~and~'xshift'.\\
5675     If~you~go~on,~this~key~will~be~ignored. \\
5676 }

```

```

5677 \@@_msg_new:nnn { Unknown~key~for~notes }
5678 {
5679   The~key~'\l_keys_key_str'~is~unknown.\\
5680   If~you~go~on,~it~will~be~ignored. \\
5681   For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
5682 }
5683 {
5684   The~available~keys~are~(in~alphabetic~order):~
5685   bottomrule,~
5686   code~after,~
5687   code~before,~
5688   enumitem~keys,~
5689   enumitem~keys~para,~
5690   para,~
5691   label~in~list,~
5692   label~in~tabular~and~
5693   style.
5694 }
5695 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
5696 {
5697   The~key~'\l_keys_key_str'~is~unknown~for~the~command~
5698   \token_to_str:N \NiceMatrixOptions. \\
5699   If~you~go~on,~it~will~be~ignored. \\
5700   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5701 }
5702 {
5703   The~available~keys~are~(in~alphabetic~order):~
5704   allow~duplicate~names,~
5705   cell~space~bottom~limit,~
5706   cell~space~limits,~
5707   cell~space~top~limit,~
5708   code~for~first~col,~
5709   code~for~first~row,~
5710   code~for~last~col,~
5711   code~for~last~row,~
5712   create~extra~nodes,~
5713   create~medium~nodes,~
5714   create~large~nodes,~
5715   delimiters~color,~
5716   end~of~row,~
5717   first~col,~
5718   first~row,~
5719   hlines,~
5720   hvlines,~
5721   hvlines~except~corners,~
5722   last~col,~
5723   last~row,~
5724   left~margin,~
5725   letter~for~dotted~lines,~
5726   light~syntax,~
5727   notes~(several~subkeys),~
5728   nullify~dots,~
5729   renew~dots,~
5730   renew~matrix,~
5731   right~margin,~
5732   small,~
5733   sub~matrix~(several~subkeys),
5734   vlines,~
5735   xdots~(several~subkeys).
5736 }
5737 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
5738 {
5739   The~key~'\l_keys_key_str'~is~unknown~for~the~environment~

```

```

5740 \{NiceArray\}. \\
5741 If-you-go-on,~it~will~be~ignored. \\
5742 For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5743 }
5744 {
5745 The~available~keys~are~(in~alphabetic~order):~
5746 b,~
5747 baseline,~
5748 c,~
5749 cell-space-bottom-limit,~
5750 cell-space-limits,~
5751 cell-space-top-limit,~
5752 code-after,~
5753 code-for-first-col,~
5754 code-for-first-row,~
5755 code-for-last-col,~
5756 code-for-last-row,~
5757 colortbl-like,~
5758 columns-width,~
5759 create-extra-nodes,~
5760 create-medium-nodes,~
5761 create-large-nodes,~
5762 delimiters-color,~
5763 extra-left-margin,~
5764 extra-right-margin,~
5765 first-col,~
5766 first-row,~
5767 hlines,~
5768 hvlines,~
5769 hvlines-except-corners,~
5770 last-col,~
5771 last-row,~
5772 left-margin,~
5773 light-syntax,~
5774 name,~
5775 notes/bottomrule,~
5776 notes/para,~
5777 nullify-dots,~
5778 renew-dots,~
5779 right-margin,~
5780 rules/color,~
5781 rules/width,~
5782 small,~
5783 t,~
5784 vlines,~
5785 xdots/color,~
5786 xdots/shorten~and~
5787 xdots/line-style.
5788 }

```

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).

```

5789 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
5790 {
5791 The~key~'\l_keys_key_str'~is~unknown~for~the~
5792 \@@_full_name_env:. \\
5793 If-you-go-on,~it~will~be~ignored. \\
5794 For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5795 }
5796 {
5797 The~available~keys~are~(in~alphabetic~order):~
5798 b,~
5799 baseline,~
5800 c,~

```



```

5801 cell-space-bottom-limit,~
5802 cell-space-limits,~
5803 cell-space-top-limit,~
5804 code-after,~
5805 code-for-first-col,~
5806 code-for-first-row,~
5807 code-for-last-col,~
5808 code-for-last-row,~
5809 colortbl-like,~
5810 columns-width,~
5811 create-extra-nodes,~
5812 create-medium-nodes,~
5813 create-large-nodes,~
5814 delimiters-color,~
5815 extra-left-margin,~
5816 extra-right-margin,~
5817 first-col,~
5818 first-row,~
5819 hlines,~
5820 hvlines,~
5821 hvlines-except-corners,~
5822 l,~
5823 last-col,~
5824 last-row,~
5825 left-margin,~
5826 light-syntax,~
5827 name,~
5828 nullify-dots,~
5829 r,~
5830 renew-dots,~
5831 right-margin,~
5832 rules/color,~
5833 rules/width,~
5834 small,~
5835 t,~
5836 vlines,~
5837 xdots/color,~
5838 xdots/shorten-and~
5839 xdots/line-style.
5840 }
5841 \@_msg_new:nnn { Unknown~option~for~NiceTabular }
5842 {
5843   The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
5844   \{NiceTabular\}. \\
5845   If~you~go~on,~it~will~be~ignored. \\
5846   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5847 }
5848 {
5849   The~available~keys~are~(in~alphabetic~order):~
5850   b,~
5851   baseline,~
5852   c,~
5853   cell-space-bottom-limit,~
5854   cell-space-limits,~
5855   cell-space-top-limit,~
5856   code-after,~
5857   code-for-first-col,~
5858   code-for-first-row,~
5859   code-for-last-col,~
5860   code-for-last-row,~
5861   colortbl-like,~
5862   columns-width,~
5863   create-extra-nodes,~

```

```

5864     create-medium-nodes,~
5865     create-large-nodes,~
5866     extra-left-margin,~
5867     extra-right-margin,~
5868     first-col,~
5869     first-row,~
5870     hlines,~
5871     hvlines,~
5872     hvlines-except-corners,~
5873     last-col,~
5874     last-row,~
5875     left-margin,~
5876     light-syntax,~
5877     name,~
5878     notes/bottomrule,~
5879     notes/para,~
5880     nullify-dots,~
5881     renew-dots,~
5882     right-margin,~
5883     rules/color,~
5884     rules/width,~
5885     t,~
5886     vlines,~
5887     xdots/color,~
5888     xdots/shorten-and~
5889     xdots/line-style.
5890 }

5891 \@@_msg_new:nnn { Duplicate-name }
5892 {
5893     The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
5894     the~same~environment~name~twice.~You~can~go~on,~but,~
5895     maybe,~you~will~have~incorrect~results~especially~
5896     if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
5897     message~again,~use~the~key~'allow-duplicate-names'~in~
5898     '\token_to_str:N \NiceMatrixOptions'.\\
5899     For~a~list~of~the~names~already~used,~type~H~<return>. \\
5900 }
5901 {
5902     The~names~already~defined~in~this~document~are:~
5903     \seq_use:Nnnn \g_@@_names_seq { ~and~ } { ,~ } { ~and~ }.
5904 }

5905 \@@_msg_new:nn { Option-auto-for-columns-width }
5906 {
5907     You~can't~give~the~value~'auto'~to~the~key~'columns-width'~here.~
5908     If~you~go~on,~the~key~will~be~ignored.
5909 }

```

## 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>49</sup>, Tikz externalization is now deactivated in the environments of the package `nicematrix`.<sup>50</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`).

$$\begin{pmatrix} & C_j & \\ 0 & \vdots & 0 \\ & a & \cdots \\ 0 & & 0 \end{pmatrix} L_i$$

---

<sup>49</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>50</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 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`.

## 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>51</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).

## 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.

---

<sup>51</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 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 customization 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](#)).

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 `hvline` 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` can take in as value 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.

# 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	
@@ commands:		
<code>\@@_Block:</code>	1099, 4438	<code>\@@_Hdotsfor:nnnn</code> 3213, 3225
<code>\@@_Block_i</code>	4440, 4443	<code>\@@_Hdotsfor_i</code> 3196, 3202, 3209
<code>\@@_Block_ii:nnnnn</code>	4443, 4444	<code>\@@_Hline:</code> 1094, 3948
<code>\@@_Block_iv:nnnnn</code>	4474, 4478	<code>\@@_Hline_i:n</code> 3948, 3949, 3955
<code>\@@_Block_iv:nnnnnn</code>	4637, 4639	<code>\@@_Hline_ii:nn</code> 3952, 3955
<code>\@@_Block_v:nnnnn</code>	4475, 4573	<code>\@@_Hline_iii:n</code> 3953, 3956
<code>\@@_Block_v:nnnnnn</code>	4665, 4667	<code>\@@_Hspace:</code> 1095, 3138
<code>\@@_Cdots</code>	1024, 1089, 3052	<code>\@@_Iddots</code> 1027, 1092, 3108
<code>\g_@@_Cdots_lines_tl</code>	1116, 2382	<code>\g_@@_Iddots_lines_tl</code> 1120, 2381
<code>\@@_Cell:</code>	199, 786, 1507, 1554, 1572, 2163, 3152, 3153, 3154, 3155, 3156, 3157, 3158, 3159, 3160, 3161, 3162, 3163	<code>\@@_Ldots</code> 1023, 1028, 1088, 3036
<code>\@@_CodeAfter:</code>	1103, 5042	<code>\g_@@_Ldots_lines_tl</code> 1117, 2383
<code>\@@_CodeAfter_i:n</code>	788, 5042, 5043, 5054	<code>\l_@@_Matrix_bool</code> 245, 1387, 1403, 2154
<code>\@@_CodeAfter_ii:n</code>	5046, 5048	<code>\l_@@_NiceArray_bool</code> 242, 345, 1164, 1277, 1332, 1433, 1445, 2130, 3825, 3826, 3944, 3945, 4147, 4154, 4966
<code>\@@_CodeAfter_keys:</code>	2321, 2343	<code>\g_@@_NiceMatrixBlock_int</code> 233, 4208, 4213, 4216, 4227
<code>\@@_Ddots</code>	1026, 1091, 3084	<code>\l_@@_NiceTabular_bool</code> 155, 243, 794, 959, 1139, 1219, 1223, 1365, 1369, 1434, 1446, 2183, 2192, 4506, 4582, 4974
<code>\g_@@_Ddots_lines_tl</code>	1119, 2380	<code>\@@_NotEmpty:</code> 1105, 2177
<code>\g_@@_HVDotsfor_lines_tl</code>	1121, 2378, 3211, 3287, 5410	<code>\@@_OnlyMainNiceMatrix:n</code> 1101, 3686
<code>\@@_Hdotsfor:</code>	1029, 1096, 3187	<code>\@@_OnlyMainNiceMatrix_i:n</code> 3689, 3696, 3699



<code>\@@_SubMatrix</code> .....	2318, 5153	<code>\@@_cellcolor</code> .....	1213, 3493, 3505, 3506
<code>\@@_Vdots</code> .....	1025, 1090, 3068	<code>\@@_cellcolor_tabular</code> .....	1033, 3663
<code>\g_@@_Vdots_lines_tl</code> .....	1118, 2379	<code>\g_@@_cells_seq</code> .....	1880, 1881, 1882, 1884
<code>\@@_Vdotsfor:</code> .....	1097, 3285	<code>\@@_chessboardcolors</code> .....	1218, 3498
<code>\@@_Vdotsfor:nnnn</code> .....	3289, 3300	<code>\@@_cline</code> .....	134, 1087
<code>\@@_W:</code> .....	1406, 1475	<code>\@@_cline_i:nn</code> .....	135, 136, 148, 151
<code>\@@_actually_color:</code> .....	1222, 3425	<code>\@@_cline_i:w</code> .....	136, 137
<code>\@@_actually_diagbox:nnnnnn</code> .....	4485, 4700, 4986, 5002	<code>\l_@@_code_before_bool</code> .....	272, 565, 592, 974, 1160, 1170,
<code>\@@_actually_draw_Cdots:</code> .....	2629, 2633		1900, 1917, 1935, 1966, 1992, 2019, 2242, 2337
<code>\@@_actually_draw_Ddots:</code> .....	2755, 2759	<code>\l_@@_code_before_tl</code> ..	271, 564, 1161, 1221
<code>\@@_actually_draw_Iddots:</code> .....	2806, 2810	<code>\l_@@_code_for_first_col_tl</code> .....	514, 2056
<code>\@@_actually_draw_Ldots:</code> ..	2587, 2591, 3276	<code>\l_@@_code_for_first_row_tl</code> ..	518, 803, 4784
<code>\@@_actually_draw_Vdots:</code> ..	2682, 2686, 3351	<code>\l_@@_code_for_last_col_tl</code> .....	516, 2099
<code>\@@_adapt_S_column:</code> .....	169, 184, 1138	<code>\l_@@_code_for_last_row_tl</code> ..	520, 810, 4787
<code>\@@_add_to_colors_seq:nn</code> .....	3412, 3424, 3446, 3461, 3476	<code>\g_@@_col_total_int</code> .....	792, 1112, 1318, 1985, 1986,
<code>\@@_adjust_pos_of_blocks_seq:</code> ..	2290, 2345		2022, 2026, 2031, 2032, 2089, 2200, 2203,
<code>\@@_adjust_pos_of_blocks_seq_i:nnnn</code> ..	2348, 2350		2208, 2215, 2259, 2719, 3183, 3184, 3346,
<code>\@@_adjust_size_box:</code> .....	865, 891, 1581, 2066, 2110		4254, 4264, 4298, 4385, 4651, 4819, 5164, 5203
<code>\@@_after_array:</code> .....	1396, 2196	<code>\l_@@_color_tl</code> .....	291, 4435, 4498, 4500
<code>\g_@@_after_col_zero_bool</code> .....	268, 1000, 2043, 3193	<code>\l_@@_colors_seq</code> ..	1220, 3415, 3419, 3420, 3429
<code>\@@_analyze_end:Nn</code> .....	1841, 1886	<code>\@@_colortbl_like:</code> .....	1031, 1106
<code>\l_@@_argspec_tl</code> .....	3034, 3035, 3036, 3052, 3068, 3084, 3108, 3207,	<code>\l_@@_colortbl_like_bool</code> ..	425, 591, 1106, 1421
	3208, 3209, 3283, 3284, 3285, 3361, 3362, 3363	<code>\c_@@_colortbl_loaded_bool</code> ...	87, 91, 1056
<code>\@@_array:</code> .....	954, 1842, 1869	<code>\l_@@_cols_tl</code> .....	3454, 3469, 3490, 3515, 3523, 3524, 3625
<code>\c_@@_arydshln_loaded_bool</code> ...	23, 30, 1494	<code>\@@_columncolor</code> .....	1217, 3457
<code>\l_@@_auto_columns_width_bool</code> .....	450, 575, 1953, 1957, 4203	<code>\@@_columncolor:n</code> .....	3463, 3466
<code>\l_@@_baseline_tl</code> ..	439, 440, 568, 569, 570, 571, 967, 1334, 1654, 1666, 1671, 1673,	<code>\@@_columncolor_preamble</code> .....	1035, 3677
	1678, 1683, 1765, 1766, 1770, 1775, 1777, 1782	<code>\c_@@_columncolor_regex</code> .....	208, 1424
<code>\@@_begin_of_NiceMatrix:nn</code> ....	2152, 2173	<code>\l_@@_columns_width_dim</code> .....	234, 576, 707, 1954, 1960, 4209, 4215
<code>\@@_begin_of_row:</code> .....	791, 815, 2044	<code>\g_@@_com_or_env_str</code> .....	256, 259
<code>\l_@@_block_auto_columns_width_bool</code> ..	1152, 1958, 4196, 4201, 4211, 4221	<code>\@@_computations_for_large_nodes:</code> ...	4325, 4338, 4343
<code>\g_@@_block_box_int</code> .....	296, 1132, 4480, 4494, 4496, 4526, 4538, 4550, 4559, 4569	<code>\@@_computations_for_medium_nodes:</code> ..	4245, 4314, 4324, 4335
<code>\g_@@_blocks_dp_dim</code> .....	239, 873, 876, 877, 4553, 4556	<code>\@@_compute_a_corner:nnnnnn</code> .....	4019, 4021, 4023, 4025, 4030
<code>\g_@@_blocks_ht_dim</code> .....	238, 879, 882, 883, 4544, 4547	<code>\@@_compute_corners:</code> .....	2289, 4011
<code>\g_@@_blocks_seq</code> .....	283, 1154, 1704, 4563, 4575, 4637	<code>\@@_construct_preamble:</code> .....	1290, 1400
<code>\g_@@_blocks_wd_dim</code> .....	237, 867, 870, 871, 4532, 4535	<code>\@@_create_col_nodes:</code> ....	1845, 1873, 1892
<code>\c_@@_booktabs_loaded_bool</code> ..	24, 33, 1039, 1736	<code>\@@_create_extra_nodes:</code> .....	1703, 4235
<code>\@@_cartesian_path:</code> .....	3455, 3470, 3491, 3576, 3588, 3623	<code>\@@_create_large_nodes:</code> .....	4243, 4319
<code>\l_@@_cell_box</code> ..	793, 839, 841, 847, 853, 856, 860, 869, 870, 875, 876, 881, 882, 892, 893,	<code>\@@_create_medium_and_large_nodes:</code> ..	4240, 4330
	894, 895, 897, 900, 904, 906, 924, 1041, 1232, 1234, 1571, 1582, 2045, 2069, 2072,	<code>\@@_create_medium_nodes:</code> .....	4241, 4309
	2074, 2090, 2113, 2117, 4708, 4812, 4846, 4981	<code>\@@_create_nodes:</code> ..	4316, 4327, 4337, 4340, 4381
<code>\l_@@_cell_space_bottom_limit_dim</code> ...	428, 496, 895	<code>\@@_create_row_node:</code> .....	970, 1003, 1040
<code>\l_@@_cell_space_top_limit_dim</code> ..	427, 494, 893	<code>\@@_cut_on_hyphen:w</code> .....	3437, 3483, 3488, 3544,
<code>\l_@@_cell_type_tl</code> .....	235, 236, 1507, 1573, 4458, 4460		3629, 3630, 3646, 3647, 4866, 4875, 5156, 5161
		<code>\g_@@_ddots_int</code> .....	2269, 2779, 2780
		<code>\@@_def_env:nnn</code> .....	2136, 2147, 2148, 2149, 2150, 2151
		<code>\@@_define_L_C_R:</code> .....	221, 1289
		<code>\c_@@_define_L_C_R_bool</code> ...	220, 1289, 5330
		<code>\@@_define_com:nnn</code> .....	4950, 4958, 4959, 4960, 4961, 4962
		<code>\@@_delimiter:nnn</code> .....	1601, 1609, 5057

\l_@@_delimiters_color_tl .....	3383, 3385, 3404, 3407, 4041, 4060, 4078,
..... 461, 687, 741, 758, 1357, 1358,	4267, 4269, 4277, 4388, 4397, 4415, 4723,
5038, 5095, 5096, 5123, 5266, 5267, 5290, 5291	4730, 4734, 4748, 4753, 4765, 4770, 4771,
\g_@@_delta_x_one_dim .... 2271, 2782, 2792	4774, 4791, 4825, 5072, 5075, 5183, 5185,
\g_@@_delta_x_two_dim .... 2273, 2833, 2843	5190, 5192, 5206, 5208, 5213, 5215, 5241, 5252
\g_@@_delta_y_one_dim .... 2272, 2784, 2792	\g_@@_env_int .. 227, 228, 230, 1151, 1158,
\g_@@_delta_y_two_dim .... 2274, 2835, 2843	1162, 1172, 1176, 1179, 1188, 1189, 1196,
\@@_diagbox:nn .....	1197, 1242, 1245, 1260, 1263, 2207, 2228,
..... 1104, 4982	2246, 2249, 2262, 2333, 3531, 3534, 3559, 4424
\@@_dotfill: .....	\@@_error:n .....
..... 4971	11, 321, 346, 471,
\@@_dotfill_i: .....	481, 645, 690, 700, 706, 715, 723, 743, 750,
..... 4976, 4978	760, 766, 771, 782, 784, 1313, 1323, 1390,
\@@_dotfill_ii: .....	1688, 1741, 1787, 3518, 4620, 4632, 5040,
..... 4975, 4978, 4979	5137, 5148, 5151, 5165, 5340, 5345, 5372, 5382
\@@_dotfill_iii: .....	\@@_error:nn .....
..... 4979, 4980	12, 583, 1483, 1484, 1485, 3039,
\@@_double_int_eval:n .... 3357, 3371, 3372	3042, 3055, 3058, 3071, 3074, 3088, 3089,
\g_@@_dp_ante_last_row_dim .... 818, 1072	3094, 3095, 3112, 3113, 3118, 3119, 4027, 5142
\g_@@_dp_last_row_dim .....	\@@_error:nnn .....
..... 818, 819, 1075, 1076, 1233, 1234, 1351	13, 3388
\g_@@_dp_row_zero_dim .....	\@@_error_too_much_cols: .....
..... 838, 839, 1066, 1067, 1344, 1759, 1798	1456, 5395
\@@_draw_Cdots:nnn .....	\@@_everycr: .....
..... 2615	996, 1061, 1064
\@@_draw_Ddots:nnn .....	\@@_everycr_i: .....
..... 2747	996, 997
\@@_draw_Iddots:nnn .....	\l_@@_except_corners_clist .....
..... 2798	446, 551, 555, 3726, 3848, 4014
\@@_draw_Ldots:nnn .....	\l_@@_exterior_arraycolsep_bool .....
..... 2573	441, 703, 1436, 1448
\@@_draw_Vdots:nnn .....	\l_@@_extra_left_margin_dim .....
..... 2667	456, 543, 1293, 2077
\@@_draw_blocks: .....	\l_@@_extra_right_margin_dim .....
..... 1704, 4634	457, 544, 1305, 2121, 2722
\@@_draw_dotted_lines: .....	\@@_extract_coords_values: .... 4406, 4413
..... 2288, 2367	\@@_fatal:n .... 14, 250, 1142, 1597, 1607,
\@@_draw_dotted_lines_i: .....	1850, 1854, 1856, 1889, 3198, 5400, 5403, 5406
..... 2370, 2374	\@@_fatal:nn .....
\l_@@_draw_first_bool . 295, 3099, 3123, 3134	15, 1499
\@@_draw_hlines: .....	\l_@@_fill_tl ... 289, 4615, 4621, 4686, 4691
..... 2301, 3941	\l_@@_final_i_int .....
\@@_draw_line: .....	2278, 2395, 2400, 2403, 2428,
..... 2613,	2436, 2440, 2449, 2457, 2537, 2567, 2607,
2665, 2745, 2796, 2847, 2849, 3410, 4164, 4194	2704, 2771, 2822, 3230, 3258, 3326, 3336, 3338
\@@_draw_line_ii:nn .....	\l_@@_final_j_int .....
..... 3390, 3394	2279, 2396, 2401, 2408, 2413, 2419, 2429,
\@@_draw_line_iii:nn .....	2437, 2441, 2450, 2458, 2538, 2568, 2604,
..... 3397, 3401	2644, 2773, 2824, 3251, 3261, 3263, 3305, 3334
\@@_draw_non_standard_dotted_line: ..	\l_@@_final_open_bool .....
..... 2855, 2857	2281, 2402,
\@@_draw_non_standard_dotted_line:n ..	2406, 2409, 2416, 2422, 2426, 2442, 2602,
..... 2860, 2863	2642, 2651, 2662, 2689, 2702, 2710, 2731,
\@@_draw_non_standard_dotted_line:nnn	2769, 2820, 2956, 2971, 3002, 3003, 3228,
..... 2865, 2870, 2884	3252, 3264, 3303, 3327, 3339, 3380, 4130, 4169
\@@_draw_standard_dotted_line: . 2854, 2885	\@@_find_extremities_of_line:nnnn ...
\@@_draw_standard_dotted_line_i: 2948, 2952	..... 2390, 2577, 2619, 2671, 2751, 2802
\l_@@_draw_tl .....	\l_@@_first_col_int .....
..... 290, 4617,	122,
4622, 4673, 4854, 4860, 4862, 4864, 4895, 4896	135, 299, 300, 510, 764, 791, 1327, 1428,
\@@_draw_vlines: .....	1895, 1915, 2253, 3191, 3637, 3688, 4254,
..... 2302, 3822	4264, 4298, 4346, 4385, 4932, 4938, 4944, 5203
\g_@@_empty_cell_bool .... 279, 899, 908,	\l_@@_first_row_int .....
2079, 2125, 3050, 3066, 3082, 3106, 3129, 3140	297, 298, 511, 768, 1110, 1342, 1685,
\l_@@_empty_corner_cells_seq .... 2295,	1756, 1784, 1795, 2251, 4247, 4261, 4288,
3760, 3766, 3773, 3881, 3887, 3894, 4013, 4086	4345, 4383, 4727, 4745, 4930, 5069, 5180, 5523
\@@_end_Cell: .....	\c_@@_footnote_bool .....
..... 201, 886, 1509,	1127, 1398, 5307, 5343, 5366, 5369, 5379, 5385
1561, 1577, 2163, 3152, 3153, 3154, 3155,	\c_@@_footnotehyper_bool . 5306, 5344, 5376
3156, 3157, 3158, 3159, 3160, 3161, 3162, 3163	\@@_full_name_env: .... 257, 5416, 5423,
\l_@@_end_of_row_tl .....	5431, 5439, 5443, 5535, 5538, 5551, 5556,
..... 458, 459, 508, 1865, 1866, 5505	
\c_@@_endpgfortikzpicture_tl .....	
..... 42, 46, 2371, 3398, 4133	
\c_@@_enumitem_loaded_bool .....	
..... 25, 36, 318, 619, 624, 635, 640	
\@@_env: .....	
.. 228, 232, 824, 830, 925, 931, 979, 985,	
991, 1184, 1185, 1191, 1192, 1199, 1200,	
1210, 1901, 1904, 1906, 1922, 1928, 1931,	
1940, 1946, 1949, 1971, 1977, 1980, 1997,	
2003, 2009, 2022, 2026, 2032, 2313, 2448,	
2516, 2555, 2566, 3239, 3257, 3314, 3332,	

5573, 5592, 5597, 5602, 5609, 5628, 5637, 5792

\@@\_hdottedline: ..... 1093, 4115

\@@\_hdottedline:n ..... 4123, 4127

\@@\_hdottedline\_i: ..... 4118, 4120

\@@\_hdottedline\_i:n ..... 4132, 4136

\@@\_hline:nn ..... 3829, 3946, 3964

\@@\_hline\_i:nn ..... 2299, 3832, 3835

\@@\_hline\_i\_complete:nn ..... 2299, 3939

\@@\_hline\_ii:nnnn ... 3857, 3868, 3901, 3940

\l\_@@\_hlines\_bool .....  
..... 445, 522, 527, 557, 1004, 2301

\g\_@@\_ht\_last\_row\_dim .....  
..... 820, 1073, 1074, 1231, 1232, 1350

\g\_@@\_ht\_row\_one\_dim .. 846, 847, 1070, 1071

\g\_@@\_ht\_row\_zero\_dim .....  
.... 840, 841, 1068, 1069, 1345, 1758, 1797

\@@\_i: ..... 4247, 4249,  
4250, 4251, 4252, 4261, 4267, 4269, 4270,  
4271, 4272, 4277, 4278, 4279, 4280, 4288,  
4291, 4293, 4294, 4295, 4347, 4349, 4352,  
4353, 4357, 4358, 4383, 4388, 4390, 4392,  
4396, 4397, 4408, 4415, 4417, 4419, 4423, 4424

\g\_@@\_iddots\_int ..... 2270, 2830, 2831

\l\_@@\_in\_env\_bool .... 241, 345, 1142, 1143

\c\_@@\_in\_preamble\_bool . 20, 21, 22, 615, 631

\l\_@@\_initial\_i\_int ..... 2276,  
2393, 2468, 2471, 2496, 2504, 2508, 2517,  
2525, 2535, 2556, 2598, 2653, 2655, 2698,  
2763, 2814, 3229, 3230, 3240, 3308, 3318, 3320

\l\_@@\_initial\_j\_int .....  
..... 2277, 2394, 2469, 2476,  
2481, 2487, 2497, 2505, 2509, 2518, 2526,  
2536, 2557, 2595, 2637, 2712, 2714, 2719,  
2765, 2816, 3233, 3243, 3245, 3304, 3305, 3316

\l\_@@\_initial\_open\_bool .....  
.... 2280, 2470, 2474, 2477, 2484, 2490,  
2494, 2510, 2593, 2635, 2650, 2660, 2689,  
2696, 2708, 2761, 2812, 2954, 3001, 3227,  
3234, 3246, 3302, 3309, 3321, 3379, 4129, 4168

\@@\_insert\_tabularnotes: ..... 1708, 1711

\@@\_instruction\_of\_type:nnn .....  
..... 935, 3044, 3060, 3076, 3099, 3123

\l\_@@\_inter\_dots\_dim .....  
.. 429, 430, 2285, 2959, 2966, 2977, 2985,  
2992, 2997, 3009, 3017, 4159, 4162, 4190, 4192

\g\_@@\_internal\_code\_after\_tl .....  
..... 264, 1543, 1600, 1608,  
1628, 2303, 2304, 3963, 4122, 4483, 4698, 4984

\@@\_intersect\_our\_row:nnnn ..... 3612

\@@\_intersect\_our\_row\_p:nnnn ..... 3563

\@@\_j: ..... 4254, 4256,  
4257, 4258, 4259, 4264, 4267, 4269, 4272,  
4274, 4275, 4277, 4280, 4282, 4283, 4298,  
4301, 4303, 4304, 4305, 4360, 4362, 4365,  
4367, 4371, 4372, 4385, 4388, 4389, 4391,  
4396, 4397, 4409, 4415, 4416, 4418, 4423, 4424

\l\_@@\_l\_dim .....  
.... 2932, 2933, 2946, 2947, 2959, 2965,  
2976, 2984, 2992, 2997, 3009, 3010, 3017, 3018

\l\_@@\_large\_nodes\_bool 453, 534, 4239, 4243

\g\_@@\_last\_col\_found\_bool .. 307, 1115,  
1319, 1382, 1984, 2013, 2087, 2199, 2256, 4817

\l\_@@\_last\_col\_int .....  
..... 305, 306, 691, 734, 736, 751, 767,  
783, 1182, 1256, 1262, 1269, 1322, 1440,  
2159, 2161, 2200, 2203, 2255, 2676, 2717,  
3041, 3057, 3095, 3119, 4642, 4647, 4648,  
4651, 4654, 4681, 4693, 4704, 4719, 4748,  
4753, 4761, 4776, 4934, 4940, 4946, 5399, 5417

\l\_@@\_last\_col\_without\_value\_bool ...  
..... 304, 733, 2201, 5402

\l\_@@\_last\_empty\_column\_int .....  
..... 4051, 4052, 4065, 4071, 4084

\l\_@@\_last\_empty\_row\_int .....  
..... 4033, 4034, 4047, 4068

\l\_@@\_last\_row\_int .....  
..... 301, 302, 512, 808, 854, 1008,  
1180, 1227, 1237, 1244, 1251, 1307, 1311,  
1314, 1326, 1348, 1867, 1868, 2052, 2053,  
2096, 2097, 2222, 2582, 2624, 3073, 3089,  
3113, 3694, 3702, 4641, 4644, 4645, 4650,  
4681, 4693, 4703, 4717, 4775, 4786, 4942, 5627

\l\_@@\_last\_row\_without\_value\_bool ...  
..... 303, 1239, 1309, 2220

\l\_@@\_left\_delim\_dim .....  
..... 1275, 1279, 1284, 1833, 2075

\l\_@@\_left\_delim\_tl .. 1129, 1283, 1359, 4158

\l\_@@\_left\_margin\_dim .....  
..... 454, 537, 1292, 2076, 4148, 4376

\l\_@@\_letter\_for\_dotted\_lines\_str ...  
..... 714, 725, 726, 1489

\l\_@@\_light\_syntax\_bool .....  
..... 438, 506, 1295, 1300, 2223

\@@\_light\_syntax\_i ..... 1858, 1861

\@@\_line ..... 2317, 3363

\@@\_line\_i:nn ..... 3370, 3377

\l\_@@\_line\_width\_dim .....  
..... 292, 4624, 4674, 4855, 4887, 4898

\@@\_line\_with\_light\_syntax:n ... 1872, 1876

\@@\_line\_with\_light\_syntax\_i:n .....  
..... 1871, 1877, 1878

\@@\_math\_toggle\_token: .....  
154, 888, 2046, 2063, 2091, 2107, 5026, 5030

\g\_@@\_max\_cell\_width\_dim .....  
..... 896, 897, 1153, 1959, 4202, 4228

\l\_@@\_max\_delimiter\_width\_bool .....  
..... 462, 505, 1378

\c\_@@\_max\_l\_dim ..... 2946, 2951

\l\_@@\_medium\_nodes\_bool 452, 533, 4237, 4767

\@@\_message\_hdotsfor: 5408, 5416, 5423, 5431

\@@\_msg\_new:nn .....  
.. 16, 5322, 5348, 5357, 5413, 5420, 5428,  
5436, 5441, 5447, 5452, 5457, 5462, 5467,  
5472, 5478, 5484, 5491, 5496, 5502, 5508,  
5513, 5520, 5527, 5534, 5536, 5542, 5559,  
5564, 5571, 5578, 5585, 5590, 5600, 5606,  
5613, 5619, 5625, 5633, 5635, 5641, 5670, 5905

\@@\_msg\_new:nnn ..... 17, 5308, 5548,  
5646, 5656, 5677, 5695, 5737, 5789, 5841, 5891

\@@\_msg\_redirect\_name:nn .....  
..... 18, 709, 1394, 4657, 4660

\@@\_multicolumn:nnn ..... 1098, 3144

\g\_@@\_multicolumn\_cells\_seq .....  
... 1108, 3171, 4272, 4280, 4402, 4732, 4750

\g\_@@\_multicolumn\_sizes\_seq 1109, 3173, 4403

<code>\g_@@_name_env_str</code> .....	255, 260, 261, 1136, 1137, 1888, 2131, 2132, 2140, 2141, 2170, 2181, 2189, 2339, 4954, 5397	<code>\@@_patch_preamble_vii:n</code> .....	1477, 1478, 1479, 1594
<code>\l_@@_name_str</code> .....	451, 585, 826, 829, 927, 930, 987, 990, 1240, 1249, 1252, 1258, 1267, 1270, 1905, 1906, 1930, 1931, 1948, 1949, 1979, 1980, 2005, 2008, 2028, 2031, 2210, 2214, 2231, 2235, 4393, 4396, 4420, 4423	<code>\@@_patch_preamble_viii:n</code> .....	1480, 1481, 1482, 1604
<code>\g_@@_names_seq</code> .....	240, 582, 584, 5903	<code>\@@_patch_preamble_viii_i:n</code> ....	1610, 1612
<code>\l_@@_nb_cols_int</code> .....	4921, 4926, 4929, 4933, 4939, 4945	<code>\@@_patch_preamble_x:n</code> .....	1512, 1564, 1586, 1592, 1632, 1648
<code>\l_@@_nb_rows_int</code> .....	4920, 4925, 4936	<code>\@@_patch_preamble_xi:n</code> .....	1490, 1624
<code>\@@_newcolumnntype</code> .....	1014, 1405, 1406	<code>\@@_pgf_rect_node:nnn</code> .....	401, 4769
<code>\@@_node_for_multicolumn:nn</code> ....	4404, 4411	<code>\@@_pgf_rect_node:nnnnn</code> .....	376, 4387, 4414, 4722, 4764, 5228
<code>\@@_node_for_the_cell:</code> .....	905, 911, 2073, 2122	<code>\c_@@_pgfortikzpicture_tl</code> .....	41, 45, 2369, 3396, 4131
<code>\@@_node_left:nn</code> .....	5240, 5241, 5258	<code>\@@_picture_position:</code> ....	1185, 1193, 1201
<code>\@@_node_position:</code> ..	1191, 1193, 1199, 1201	<code>\l_@@_pos_of_block_tl</code> ...	293, 294, 4429, 4431, 4433, 4459, 4460, 4462, 4505, 4509, 4516, 4566, 4589, 4591, 4602, 4605, 4626, 4628, 4630, 4793, 4805, 4816, 4820, 4827, 4839
<code>\@@_node_right:nn</code> .....	5250, 5252, 5282	<code>\g_@@_pos_of_blocks_seq</code> .....	284, 1155, 2263, 2293, 2347, 3174, 3720, 3842, 4098, 4669, 4994
<code>\g_@@_not_empty_cell_bool</code> .....	270, 903, 909, 2178	<code>\g_@@_pos_of_stroken_blocks_seq</code> .....	286, 1156, 3724, 3846, 4683
<code>\@@_not_in_exterior:nnnn</code> .....	3604	<code>\g_@@_pos_of_xdots_seq</code> .....	285, 1157, 2294, 2533, 3722, 3844
<code>\@@_not_in_exterior_p:nnnn</code> .....	3536	<code>\@@_pre_array:</code> .....	1037, 1274
<code>\l_@@_notes_above_space_dim</code> ....	447, 448	<code>\c_@@_preamble_first_col_tl</code> ....	1429, 2039
<code>\l_@@_notes_bottomrule_bool</code> .....	603, 754, 777, 1734	<code>\c_@@_preamble_last_col_tl</code> ....	1441, 2083
<code>\l_@@_notes_code_after_tl</code> .....	601, 1743	<code>\g_@@_preamble_tl</code> .....	1131, 1407, 1413, 1416, 1426, 1429, 1438, 1441, 1450, 1455, 1496, 1505, 1516, 1529, 1551, 1568, 1590, 1599, 1621, 1626, 1639, 1647, 1842, 1869
<code>\l_@@_notes_code_before_tl</code> ....	599, 1715	<code>\@@_pred:n</code> .....	123, 153, 2161, 3761, 3774, 3882, 3895
<code>\@@_notes_label_in_list:n</code> .....	314, 333, 341, 611	<code>\@@_provide_pgfsyspdfmark:</code> ..	209, 218, 1126
<code>\@@_notes_label_in_tabular:n</code> ..	313, 354, 608	<code>\@@_put_box_in_flow:</code> .....	1380, 1650, 1835
<code>\l_@@_notes_para_bool</code> ..	597, 752, 775, 1719	<code>\@@_put_box_in_flow_bis:nn</code> ....	1379, 1802
<code>\@@_notes_style:n</code> .....	312, 315, 333, 341, 357, 362, 605	<code>\@@_put_box_in_flow_i:</code> .....	1656, 1658
<code>\l_@@_nullify_dots_bool</code> .....	449, 532, 3048, 3064, 3080, 3104, 3127	<code>\@@_qpoint:n</code> ..	231, 1661, 1663, 1675, 1691, 1750, 1752, 1768, 1779, 1790, 2595, 2598, 2604, 2607, 2637, 2644, 2653, 2655, 2698, 2704, 2712, 2714, 2763, 2765, 2771, 2773, 2814, 2816, 2822, 2824, 3404, 3407, 3636, 3640, 3653, 3655, 3784, 3786, 3788, 3905, 3907, 3909, 4139, 4143, 4150, 4186, 4189, 4191, 4293, 4303, 4713, 4715, 4717, 4719, 4741, 4761, 4789, 4871, 4873, 4880, 4882, 5007, 5009, 5012, 5014, 5062, 5064, 5197, 5200
<code>\l_@@_number_of_notes_int</code> .....	311, 348, 358, 368	<code>\l_@@_radius_dim</code> .....	433, 434, 1627, 2284, 2611, 2612, 3026, 4117, 4141, 4187, 4188
<code>\@@_old_CT@arc@</code> .....	1144, 2341	<code>\l_@@_real_left_delim_dim</code> .....	1804, 1819, 1834
<code>\@@_old_arraycolsep_dim</code> .....	280	<code>\l_@@_real_right_delim_dim</code> .....	1805, 1831, 1837
<code>\@@_old_cdots</code> .....	1081, 3065	<code>\@@_rectanglecolor</code> .....	1214, 3472, 3496
<code>\@@_old_ddots</code> .....	1083, 3105	<code>\@@_rectanglecolor:nn</code> .....	3478, 3481
<code>\@@_old_dotfill</code> .....	4970, 4973, 4981	<code>\@@_renew_NC@rewrite@S:</code> ....	190, 192, 1114
<code>\@@_old_dotfill:</code> .....	1102	<code>\@@_renew_dots:</code> .....	1021, 1107
<code>\l_@@_old_iRow_int</code> .....	265, 1043, 2387	<code>\l_@@_renew_dots_bool</code> .....	530, 699, 1107, 5332, 5339
<code>\@@_old_ialign:</code> .....	969, 1077, 4636	<code>\@@_renew_matrix:</code> .....	694, 698, 4900, 5334, 5338
<code>\@@_old_iddots</code> .....	1084, 3128	<code>\l_@@_respect_blocks_bool</code> .....	3513, 3528, 3556
<code>\l_@@_old_jCol_int</code> .....	266, 1046, 2388	<code>\@@_restore_iRow_jCol:</code> .....	2340, 2385
<code>\@@_old_ldots</code> .....	1080, 3049	<code>\@@_revtex_array:</code> .....	946, 957
<code>\@@_old_multicolumn</code> .....	3143, 3147	<code>\c_@@_revtex_bool</code> .....	49, 51, 54, 56, 956
<code>\@@_old_pgful@check@rerun</code> .....	80, 84		
<code>\@@_old_vdots</code> .....	1082, 3081		
<code>\l_@@_parallelize_diags_bool</code> .....	442, 443, 529, 2267, 2777, 2828		
<code>\@@_patch_preamble:n</code> .....	1418, 1460, 1497, 1517, 1546, 1602, 1622, 1630, 1642		
<code>\@@_patch_preamble_i:n</code> .....	1464, 1465, 1466, 1503		
<code>\@@_patch_preamble_ii:nn</code> .....	1467, 1468, 1469, 1514		
<code>\@@_patch_preamble_iii:n</code> ..	1470, 1519, 1527		
<code>\@@_patch_preamble_iii_i:n</code> ....	1522, 1524		
<code>\@@_patch_preamble_iv:nnn</code> .....	1471, 1472, 1473, 1549		
<code>\@@_patch_preamble_ix:n</code> .....	1635, 1645		
<code>\@@_patch_preamble_v:nnnn</code> .....	1474, 1475, 1566		
<code>\@@_patch_preamble_vi:n</code> .....	1476, 1588		

<code>\l_@@_right_delim_dim</code> .....	1276, 1280, 1286, 1836, 2119
<code>\l_@@_right_delim_tl</code> ..	1130, 1285, 1375, 4161
<code>\l_@@_right_margin_dim</code> .....	455, 539, 1304, 2120, 2721, 4155, 4379
<code>\@@_rotate:</code> .....	1100, 3356
<code>\g_@@_rotate_bool</code> .....	246, 863, 890, 1580, 2065, 2109, 3356, 4505, 4523, 4588, 4601, 4709
<code>\@@_rotate_cell_box:</code> .....	851, 890, 1580, 2065, 2109, 4709
<code>\g_@@_row_of_col_done_bool</code> .....	269, 1001, 1135, 1914
<code>\g_@@_row_total_int</code> .....	1111, 1325, 1686, 1785, 2222, 2229, 2236, 2252, 3271, 4247, 4261, 4288, 4383, 4650, 4727, 4745, 5069, 5163, 5180, 5524
<code>\@@_rowcolor</code> .....	1215, 3442
<code>\@@_rowcolor:n</code> .....	3448, 3451
<code>\@@_rowcolor_tabular</code> .....	1034, 3668
<code>\@@_rowcolors</code> .....	1216, 3520
<code>\@@_rowcolors_i:nnnn</code> .....	3564, 3599
<code>\l_@@_rowcolors_restart_bool</code> ..	3516, 3547
<code>\g_@@_rows_seq</code> ..	1864, 1866, 1868, 1870, 1872
<code>\l_@@_rows_tl</code> ..	3453, 3468, 3489, 3566, 3642
<code>\l_@@_rules_color_tl</code> ..	267, 485, 1168, 1169
<code>\@@_set_CT@arc@:</code> .....	156, 1169
<code>\@@_set_CT@arc@_i:</code> .....	157, 158
<code>\@@_set_CT@arc@_ii:</code> .....	157, 160
<code>\@@_set_final_coords:</code> .....	2546, 2571
<code>\@@_set_final_coords_from_anchor:n</code> ..	2562, 2610, 2648, 2692, 2707, 2776, 2827
<code>\@@_set_initial_coords:</code> .....	2541, 2560
<code>\@@_set_initial_coords_from_anchor:n</code> ..	2551, 2601, 2641, 2691, 2701, 2768, 2819
<code>\@@_set_size:n</code> .....	4918, 4927
<code>\c_@@_siunitx_loaded_bool</code> ..	162, 166, 171, 189
<code>\l_@@_small_bool</code> .....	692, 739, 747, 769, 797, 1049, 1596, 1606, 2047, 2092, 2282
<code>\@@_standard_cline</code> .....	119, 1086
<code>\@@_standard_cline:w</code> .....	119, 120
<code>\l_@@_standard_cline_bool</code> ..	426, 492, 1085
<code>\c_@@_standard_tl</code> ..	436, 437, 2853, 4163, 4193
<code>\g_@@_static_num_of_col_int</code> .....	288, 1389, 1419, 4654, 5432, 5444, 5593
<code>\l_@@_stop_loop_bool</code> .....	2397, 2398, 2430, 2443, 2452, 2465, 2466, 2498, 2511, 2520
<code>\@@_stroke_block:nnn</code> .....	4678, 4851
<code>\l_@@_submatrix_extra_height_dim</code> ....	5115, 5167, 5223
<code>\l_@@_submatrix_left_xshift_dim</code> .....	5117, 5168, 5236
<code>\l_@@_submatrix_name_str</code> .....	5144, 5170, 5226, 5228, 5239, 5241, 5249, 5252
<code>\g_@@_submatrix_names_seq</code> .....	287, 2320, 5141, 5145, 5557
<code>\l_@@_submatrix_right_xshift_dim</code> ....	5119, 5169, 5246
<code>\l_@@_submatrix_slim_bool</code> .....	5125, 5178
<code>\@@_succ:n</code> .....	148, 152, 979, 985, 1544, 1601, 1663, 1997, 2003, 2008, 2009, 2022, 2026, 2031, 2032, 2257, 2604, 2644, 2655, 2704, 2714, 2771, 2773, 2816, 2822, 3608, 3640, 3653, 3757, 3788, 3826, 3878, 3909, 3945, 3964, 4103, 4105, 4107, 4109, 4150, 4191, 4353, 4357, 4367, 4371, 4717, 4719, 4761, 4880, 4882, 5012, 5014, 5064
<code>\l_@@_suffix_tl</code> .....	4315, 4326, 4336, 4339, 4388, 4396, 4397, 4415, 4423, 4424
<code>\c_@@_table_collect_begin_tl</code> ..	179, 181, 199
<code>\c_@@_table_print_tl</code> .....	182, 183, 201
<code>\l_@@_tabular_width_dim</code> .....	244, 962, 964, 1452, 2190
<code>\l_@@_tabularnote_tl</code> ..	310, 756, 779, 1707, 1716
<code>\g_@@_tabularnotes_seq</code> .....	309, 349, 1722, 1728, 1744
<code>\@@_test_if_cell_in_a_block:nn</code> .....	4037, 4055, 4073, 4093
<code>\@@_test_if_cell_in_block:nnnnnnn</code> ...	4099, 4101
<code>\@@_test_if_hline_in_block:nnnn</code> .....	3843, 3845, 3967
<code>\@@_test_if_hline_in_stroken_block:nnnn</code> .....	3847, 3989
<code>\@@_test_if_math_mode:</code> ....	247, 1141, 2142
<code>\@@_test_if_vline_in_block:nnnn</code> .....	3721, 3723, 3978
<code>\@@_test_if_vline_in_stroken_block:nnnn</code> .....	3725, 4000
<code>\@@_test_in_corner_h:</code> .....	3848, 3876
<code>\@@_test_in_corner_v:</code> .....	3727, 3755
<code>\l_@@_the_array_box</code> ..	1288, 1291, 1701, 1702
<code>\c_@@_tikz_loaded_bool</code> .....	26, 40, 1205, 2305, 4170
<code>\@@_true_c:</code> .....	200, 1476
<code>\l_@@_type_of_col_tl</code> ..	737, 738, 2171, 2173
<code>\c_@@_types_of_matrix_seq</code> .....	5387, 5388, 5393, 5397
<code>\@@_update_for_first_and_last_row:</code> ..	834, 898, 1229, 2067, 2111
<code>\@@_use_arraybox_with_notes:</code> ...	1339, 1763
<code>\@@_use_arraybox_with_notes_b:</code> ..	1336, 1747
<code>\@@_use_arraybox_with_notes_c:</code> .....	1337, 1368, 1699, 1761, 1800
<code>\@@_vdottedline:n</code> .....	1629, 4166
<code>\@@_vdottedline_i:n</code> .....	4173, 4178, 4182
<code>\@@_vline:nn</code> .....	1544, 3704, 3827
<code>\@@_vline_i:nn</code> .....	2298, 3709, 3713
<code>\@@_vline_i_complete:nn</code> .....	2298, 3820
<code>\@@_vline_ii:nnnn</code> ...	3736, 3747, 3780, 3821
<code>\l_@@_vlines_bool</code> .....	444, 523, 526, 556, 1411, 1435, 1447, 1637, 2302
<code>\@@_w:</code> .....	1405, 1474
<code>\g_@@_width_first_col_dim</code> .....	282, 1134, 1330, 1909, 2068, 2069
<code>\g_@@_width_last_col_dim</code> .....	281, 1133, 1384, 2018, 2112, 2113
<code>\l_@@_x_final_dim</code> .....	275, 2548, 2605, 2606, 2645, 2646, 2694, 2716, 2724, 2728, 2732, 2734, 2739, 2741, 2774, 2783, 2791, 2825, 2834, 2842, 2881, 2896, 2905, 2939, 2991, 3007, 3408, 4151, 4162, 4188, 5177, 5193, 5194, 5230, 5246
<code>\l_@@_x_initial_dim</code> ..	273, 2543, 2596, 2597, 2638, 2639, 2694, 2715, 2716, 2723, 2728, 2732, 2734, 2736, 2739, 2741, 2766, 2783,



2791, 2817, 2834, 2842, 2878, 2895, 2905, 2939, 2991, 3005, 3007, 3025, 3027, 3405, 4144, 4159, 4187, 5176, 5186, 5187, 5229, 5236	
\l_@@_xdots_color_tl	460, 474, 2586, 2628, 2680, 2681, 2754, 2805, 2861, 3275, 3350, 3367
\l_@@_xdots_down_tl	478, 2868, 2889, 2924
\l_@@_xdots_line_style_tl	435, 437, 470, 2853, 2861, 4163, 4193
\l_@@_xdots_shorten_dim	431, 432, 476, 2286, 2875, 2876, 2965, 2976, 2984
\l_@@_xdots_up_tl	479, 2867, 2888, 2914
\l_@@_y_final_dim	276, 2549, 2608, 2612, 2657, 2661, 2663, 2705, 2772, 2785, 2788, 2823, 2836, 2839, 2881, 2896, 2904, 2941, 2996, 3015, 3409, 4142, 4192, 5065, 5087, 5102, 5201, 5216, 5217, 5222, 5230, 5237, 5247
\l_@@_y_initial_dim	274, 2544, 2599, 2611, 2656, 2657, 2661, 2663, 2699, 2764, 2785, 2790, 2815, 2836, 2841, 2878, 2895, 2904, 2941, 2996, 3013, 3015, 3025, 3028, 3406, 4140, 4141, 4142, 4190, 5063, 5087, 5102, 5198, 5209, 5210, 5222, 5229, 5237, 5247
\	1855, 1877, 4934, 4940, 4946, 5310, 5311, 5354, 5363, 5444, 5449, 5454, 5459, 5464, 5487, 5499, 5504, 5510, 5517, 5524, 5530, 5531, 5539, 5545, 5551, 5552, 5568, 5574, 5581, 5588, 5597, 5603, 5610, 5616, 5622, 5634, 5638, 5643, 5649, 5658, 5659, 5672, 5674, 5675, 5679, 5680, 5698, 5699, 5740, 5741, 5792, 5793, 5844, 5845, 5898, 5899
\{	261, 1479, 1618, 2149, 4962, 5544, 5610, 5740, 5844
\}	261, 1482, 2149, 4962, 5544, 5610, 5740, 5844
\	2151, 4961
\sqcup	5411, 5416, 5423, 5431, 5432, 5443, 5444, 5486, 5498, 5523, 5524, 5535, 5538, 5550, 5556, 5561, 5592, 5593, 5594, 5602, 5608, 5621, 5628, 5629, 5637

## A

\A	5139
\aboverulesep	1738
\addtocounter	366
\alph	312
\arraybackslash	1557
\arraycolsep	538, 540, 542, 961, 1052, 1279, 1280, 1367, 1371, 4147, 4154
\arrayrulecolor	94
\arrayrulewidth	127, 132, 144, 487, 825, 978, 980, 986, 1009, 1414, 1537, 1640, 1755, 1794, 1921, 1923, 1929, 1939, 1941, 1947, 1970, 1972, 1978, 1996, 1998, 2004, 3638, 3639, 3641, 3654, 3656, 3796, 3797, 3799, 3810, 3816, 3918, 3929, 3935, 3960, 4228, 4855, 5087, 5223
\arraystretch	1051, 4502, 4585, 4598, 5199, 5202
\AtBeginDocument	22, 27, 72, 88, 163, 187, 316, 430, 432, 434, 448, 617, 633, 2365, 3032, 3205, 3281, 3359, 3392, 4125
\AutoNiceMatrix	4963
\AutoNiceMatrixWithDelims	4923, 4955, 4967

## B

\baselineskip	97, 104
\bgroup	1128
\Block	1099, 5486, 5649
\BNiceMatrix	4915
\bNiceMatrix	4912
bool commands:	
\bool_do_until:Nn	2398, 2466
\bool_gset_false:N	863, 908, 909, 1000, 1115, 1135, 2079, 2125, 3976, 3987, 3998, 4009, 4528
\bool_gset_true:N	1914, 2043, 2087, 2178, 3050, 3066, 3082, 3106, 3129, 3140, 3356, 3719, 3841
\bool_if:NTF	155, 171, 619, 624, 635, 640, 794, 797, 890, 974, 1001, 1004, 1039, 1049, 1106, 1107, 1127, 1142, 1152, 1170, 1205, 1219, 1223, 1289, 1309, 1382, 1387, 1398, 1403, 1421, 1580, 1596, 1606, 1637, 1734, 1900, 1917, 1935, 1953, 1966, 1992, 2013, 2019, 2047, 2065, 2092, 2109, 2199, 2201, 2220, 2223, 2242, 2267, 2282, 2301, 2302, 2305, 2660, 2662, 2777, 2828, 3048, 3064, 3080, 3104, 3127, 4046, 4064, 4082, 4211, 4221, 4243, 4505, 4523, 4588, 4601, 4709, 4767, 4974, 5366, 5376, 5402
\bool_if:nTF	189, 318, 345, 937, 1319, 3381, 3614, 4237, 4817, 5066, 5076, 5078, 5091, 5097, 5106
\bool_lazy_all:nTF	1431, 1443, 2291, 3969, 3980, 3991, 4002
\bool_lazy_and:nnTF	1492, 1956, 2048, 2254, 2649, 2887, 3189, 3527, 3555, 3790, 3911, 4867
\bool_lazy_any:nTF	1614
\bool_lazy_or:nnTF	467, 902, 1684, 1705, 1783, 2095, 2689, 2945, 3606, 4038, 4056, 4074, 4446, 4451, 4471, 4649, 4672, 5162
\bool_lazy_or_p:nn	2051
\bool_not_p:n	1434, 1435, 1436, 1446, 1447, 1448, 1958, 2256
\bool_set:Nn	2693, 3549
\c_false_bool	1609, 3044, 3060, 3076
\g_tmpa_bool	3719, 3728, 3762, 3770, 3775, 3841, 3849, 3883, 3891, 3896, 3976, 3987, 3998, 4009
\l_tmpb_bool	4043, 4057, 4075, 4097, 4110
\c_true_bool	1601
box commands:	
\box_clear_new:N	1041, 1288
\box_dp:N	819, 839, 876, 895, 1067, 1076, 1234, 1653, 1813, 1826, 4558, 5202
\box_gclear_new:N	4493
\box_grotate:Nn	4525
\box_ht:N	820, 841, 847, 859, 882, 893, 1069, 1071, 1074, 1232, 1652, 1813, 1826, 4549, 5199
\box_move_up:nn	63, 65, 67, 1696, 1761, 1800
\box_rotate:Nn	853
\box_set_dp:Nn	875, 894, 1653
\box_set_ht:Nn	881, 892, 1652
\box_set_wd:Nn	869
\box_use:N	369, 860, 1558, 1561

<code>\box_use_drop:N</code> . . . . .	900, 906, 924, 1582, 1655, 1696, 1697, 1702, 2074, 4568, 4812, 4846
<code>\box_wd:N</code> . . . . .	370, 870, 897, 904, 1284, 1286, 1701, 1820, 1832, 2069, 2072, 2113, 2117, 4537, 4981
<code>\l_tmpa_box</code> . . . . .	352, 369, 370, 1283, 1284, 1285, 1286, 1354, 1652, 1653, 1655, 1696, 1697, 1813, 1826
<code>\l_tmpb_box</code> . . . . .	1806, 1820, 1821, 1832
<b>C</b>	
<code>\c</code> . . . . .	208, 1425
<code>\Cdots</code> . . . . .	1089, 3055, 3058
<code>\cdots</code> . . . . .	1024, 1081
<code>\cellcolor</code> . . . . .	1033, 1213, 3666
<code>\chessboardcolors</code> . . . . .	1218
<code>\cline</code> . . . . .	147, 1086, 1087
clist commands:	
<code>\clist_if_empty:N</code> . . . . .	3726, 3848
<code>\clist_map_inline:Nn</code> . . . . .	3625, 3642, 4014
<code>\clist_map_inline:nn</code> . . . . .	2166, 3495, 3540
<code>\clist_new:N</code> . . . . .	446
<code>\clist_set:Nn</code> . . . . .	555
<code>\CodeAfter</code> . . . . .	788, 1103, 1858, 1861, 2319, 5661
<code>\color</code> . . . . .	98, 105, 159, 161, 1358, 2580, 2583, 2586, 2622, 2625, 2628, 2674, 2677, 2681, 2754, 2805, 3269, 3272, 3275, 3344, 3347, 3350, 3367, 3431, 3573, 3574, 3585, 3586, 4500, 4622, 5096, 5267, 5291
<code>\colorlet</code> . . . . .	253, 254, 804, 811, 2057, 2100
<code>\columncolor</code> . . . . .	1035, 1217, 2328, 3682
<code>\cr</code> . . . . .	131, 149, 2037
<code>\crr</code> . . . . .	1894
cs commands:	
<code>\cs_generate_variant:Nn</code> . . . . .	57, 151, 2884, 3424, 4233, 4234
<code>\cs_gset:Npn</code> . . . . .	98, 105, 2207, 2214, 2228, 2235, 4226
<code>\cs_gset_eq:NN</code> . . . . .	184, 218, 1060, 1144, 2341
<code>\cs_if_exist:N</code> . . . . .	56, 1042, 1045, 1145, 1148, 1242, 1249, 1260, 1267, 2387, 2388, 2433, 2446, 2501, 2514, 3237, 3255, 3312, 3330, 4213, 4266, 4729, 4747, 5071, 5182, 5189, 5205, 5212
<code>\cs_if_exist_p:N</code> . . . . .	468, 3530, 3558, 4040, 4059, 4077
<code>\cs_if_free:N</code> . . . . .	214, 2575, 2617, 2669, 2749, 2800
<code>\cs_if_free_p:N</code> . . . . .	3383, 3385
<code>\cs_new_protected:Npx</code> . . . . .	2367, 3394, 4127
<code>\cs_set:Nn</code> . . . . .	605, 608, 611
<code>\cs_set:Npn</code> . . . . .	94, 95, 101, 102, 107, 119, 120, 134, 136, 137, 159, 161, 315, 1016, 2392, 2454, 2522, 3279, 3354, 3948, 3949, 3955, 3956, 4502, 4585, 4598
<code>\cs_set_nopar:Npn</code> . . . . .	951, 963, 1051, 1054, 4408, 4409
<code>\cs_set_nopar:Npx</code> . . . . .	964
<code>\cs_set_protected:Npn</code> . . . . .	4952
<code>\cs_set_protected_nopar:Npn</code> . . . . .	4481, 4696
<b>D</b>	
<code>\Ddots</code> . . . . .	1091, 3088, 3089, 3094, 3095
<code>\ddots</code> . . . . .	1026, 1083
<code>\diagbox</code> . . . . .	1104, 4481, 4696
dim commands:	
<code>\dim_add:Nn</code> . . . . .	4377
<code>\dim_compare:nNnTF</code> . . . . .	97, 104, 867, 873, 879, 1452, 1954, 2117, 2734, 4290, 4300, 4739, 4759, 4981
<code>\dim_gzero:N</code> . . . . .	871, 877, 883
<code>\dim_max:nn</code> . . . . .	4279, 4283
<code>\dim_min:nn</code> . . . . .	4271, 4275
<code>\dim_ratio:nn</code> . . . . .	2792, 2843, 2959, 2964, 2975, 2983, 2992, 2997, 3008, 3016
<code>\dim_set:Nn</code> . . . . .	4252, 4259, 4270, 4274, 4278, 4282, 4294, 4295, 4304, 4305, 4349, 4362
<code>\dim_set_eq:NN</code> . . . . .	4250, 4257, 4357, 4371
<code>\dim_sub:Nn</code> . . . . .	4374
<code>\dim_use:N</code> . . . . .	4291, 4301, 4352, 4353, 4365, 4366, 4389, 4390, 4391, 4392, 4416, 4417, 4418, 4419
<code>\dim_zero_new:N</code> . . . . .	4249, 4251, 4256, 4258
<code>\c_max_dim</code> . . . . .	4250, 4252, 4257, 4259, 4291, 4301, 4726, 4739, 4744, 4759, 5067, 5068, 5176, 5177
<code>\l_tmpc_dim</code> . . . . .	277, 3638, 3639, 3658, 3789, 3797, 3802, 3807, 3813, 3910, 3921, 3926, 3932, 4718, 4724, 4766, 4874, 4885, 5013, 5016, 5024
<code>\l_tmpd_dim</code> . . . . .	278, 3656, 3658, 3798, 3802, 3917, 3921, 4720, 4724, 4744, 4755, 4759, 4762, 4766, 4883, 4886, 5015, 5016, 5028
<code>\dotfill</code> . . . . .	1102, 4970
<code>\dots</code> . . . . .	1028
<code>\doublerulesep</code> . . . . .	1538, 3799, 3811, 3918, 3930, 3961
<code>\doublerulesepcolor</code> . . . . .	101
<code>\draw</code> . . . . .	2872
<b>E</b>	
<code>\egroup</code> . . . . .	1397
else commands:	
<code>\else:</code> . . . . .	249
<code>\endarray</code> . . . . .	1846, 1874
<code>\endBNiceMatrix</code> . . . . .	4916
<code>\endbNiceMatrix</code> . . . . .	4913
<code>\endNiceArray</code> . . . . .	2186, 2195
<code>\endNiceArrayWithDelims</code> . . . . .	2135, 2145
<code>\endpgfscope</code> . . . . .	2929, 5027
<code>\endpNiceMatrix</code> . . . . .	4904
<code>\endsavenotes</code> . . . . .	1398
<code>\endtabularnotes</code> . . . . .	1729
<code>\endVNiceMatrix</code> . . . . .	4910
<code>\endvNiceMatrix</code> . . . . .	4907
<code>\enskip</code> . . . . .	1599, 1621
<code>\ensuremath</code> . . . . .	3049, 3065, 3081, 3105, 3128
<code>\everycr</code> . . . . .	130, 149, 1064
exp commands:	
<code>\exp_after:wN</code> . . . . .	196, 1169, 1359, 1375, 1418
<code>\exp_args:Ne</code> . . . . .	3150
<code>\exp_args:Nnc</code> . . . . .	4215
<code>\exp_args:Nne</code> . . . . .	3146
<code>\exp_args:Nne</code> . . . . .	2173
<code>\exp_args:NNV</code> . . . . .	1866, 3036, 3052, 3068, 3084, 3108, 3209, 3285, 3363
<code>\exp_args:NNv</code> . . . . .	1161
<code>\exp_args:Nnx</code> . . . . .	4508, 4515, 4590, 4604
<code>\exp_args:No</code> . . . . .	2860

\exp_args:NV . . . . .	1407, 1842, 1869, 1871	int commands:	
\exp_args:Nxx . . . . .	4474, 4475	\int_case:nnTF . . . . .	3086, 3092, 3110, 3116
\exp_last_unbraced:NV . . . . .	2321	\int_compare:nNnTF . . . . .	122, 123, 139,
\exp_not:N . . . . .	41,	790, 791, 801, 808, 844, 854, 1006, 1008,	
42, 45, 46, 1531, 3672, 3682, 4129, 4130, 4690		1180, 1182, 1227, 1237, 1256, 1307, 1311,	
\exp_not:n . . . . .		1322, 1326, 1327, 1389, 1598, 1717, 1756,	
943, 2334, 3219, 3295, 3674, 4490,		1795, 1867, 1895, 2093, 2408, 2415, 2419,	
4566, 4578, 4579, 4679, 4691, 4705, 4991, 4992		2421, 2476, 2483, 2487, 2489, 2582, 2624,	
\ExplSyntaxOff . . . . .	216, 2218, 2239, 2265, 2336, 4230	2676, 2717, 2719, 3169, 3183, 3271, 3346,	
\ExplSyntaxOn . . . . .	213, 2204, 2225, 2244, 2329, 4223	3601, 3634, 3651, 3679, 3693, 3694, 3701,	
\extrarowheight . . . . .	4503, 4586, 4599, 5199	3702, 3706, 4103, 4105, 4107, 4109, 4499,	
		4530, 4542, 4786, 4815, 4819, 4876, 4878,	
		4930, 4932, 4934, 4938, 4940, 4942, 4944, 4946	
<b>F</b>		\int_compare_p:n . . . . .	3616, 3618, 4868, 4869
\fi . . . . .	109, 1409, 3948, 3965	\int_do_until:nNnn . . . . .	3552
fi commands:		\int_gadd:Nn . . . . .	3182
\fi: . . . . .	251	\int_gincr:N . . . . .	789, 817, 1151, 1511, 1563,
\fontdimen . . . . .	1692	1585, 1591, 1990, 2088, 2779, 2830, 4208, 4480	
fp commands:		\int_if_even:nTF . . . . .	3504
\fp_eval:n . . . . .	2900	\int_if_odd_p:n . . . . .	3549
\fp_to_dim:n . . . . .	2935	\int_incr:N . . . . .	348, 1521
\futurelet . . . . .	114	\int_step_inline:nn . . . . .	3500, 3502
		\int_step_inline:nnnn . . . . .	4035, 4053, 4068, 4071
<b>G</b>		\l_tmpc_int . . . . .	3550, 3551, 3552
\globaldefs . . . . .	1393, 4659	\c_zero_int . . . . .	1598, 3417, 3607
group commands:		iow commands:	
\group_insert_after:N . . . . .	4975, 4976, 4978, 4979	\iow_now:Nn . . . . .	75,
		211, 2244, 2245, 2247, 2265, 2329, 2330, 2336	
<b>H</b>		\iow_shipout:Nn . . . . .	2204, 2205, 2212,
\halign . . . . .	1078	2218, 2225, 2226, 2233, 2239, 4223, 4224, 4230	
\hbox . . . . .	972,	\item . . . . .	1722, 1728
1363, 1761, 1800, 1919, 1937, 1964, 1968, 1994			
hbox commands:		<b>K</b>	
\hbox:n . . . . .	63, 65, 68	\kern . . . . .	68
\hbox_gset:Nn . . . . .	4495	keys commands:	
\hbox_overlap_left:n . . . . .	1898, 2070	\keys_define:nn . . . . .	
\hbox_overlap_right:n . . . . .	369, 2015, 2115	463, 483, 490, 549, 595, 647,	
\hbox_set:Nn . . . . .		685, 729, 745, 762, 773, 3132, 3511, 4197,	
352, 1283, 1285, 1354, 1806, 1821, 4708		4427, 4613, 4893, 5033, 5113, 5128, 5133, 5328	
\hbox_set:Nw . . . . .	793, 1291, 1571, 2045, 2090	\l_keys_key_str . . . . .	5310, 5475, 5481, 5643,
\hbox_set_end: . . . . .	889, 1306, 1579, 2064, 2108	5648, 5658, 5672, 5679, 5697, 5739, 5791, 5843	
\hbox_to_wd:nn . . . . .	394, 419	\keys_set:nn . . . . .	504,
\Hdotsfor . . . . .	1096, 5411, 5561	718, 721, 728, 1165, 1166, 2172, 2182,	
\hdotsfor . . . . .	1029	2191, 2344, 2585, 2627, 2679, 2753, 2804,	
\hdottedline . . . . .	1093	3274, 3349, 3366, 3525, 4210, 4671, 5036, 5171	
\heavyrulewidth . . . . .	1739	\keys_set_known:nn . . . . .	3098, 3122, 4463, 4856
\hfil . . . . .	1475	\l_keys_value_tl . . . . .	5529, 5576, 5583, 5893
\hfill . . . . .	127, 144		
\Hline . . . . .	1094, 3951	<b>L</b>	
\hline . . . . .	107	\ldots . . . . .	1088, 3039, 3042
\hrule . . . . .	111, 127, 144, 1009, 1739, 5101, 5272, 5296	\ldots . . . . .	1023, 1080
\hskip . . . . .	110	\leaders . . . . .	127, 144
\Hspace . . . . .	1095	\left . . . . .	1359, 1809, 1824, 5097, 5268, 5292
\hspace . . . . .	3141	legacy commands:	
\hss . . . . .	1475	\legacy_if:nTF . . . . .	579
		\line . . . . .	2317, 5544, 5668
<b>I</b>			
\ialign . . . . .	969, 1054, 1077, 4636	<b>M</b>	
\iddots . . . . .	1092, 3112, 3113, 3118, 3119	\makebox . . . . .	1582
\iddots . . . . .	58, 1027, 1084	\mathinner . . . . .	60
if commands:		mode commands:	
\if_mode_math: . . . . .	249	\mode_leave_vertical: . . . . .	1140, 1556
\ifnum . . . . .	109, 3948, 3965	msg commands:	
\ifstandalone . . . . .	1148	\msg_error:nn . . . . .	11



\msg_error:nnn	12	\pgfpointorigin	1904, 2027
\msg_error:nnnn	13, 4656, 4663	\pgfpointscale	410
\msg_fatal:nn	14	\pgfpointshapeborder	3404, 3407
\msg_fatal:nnn	15	\pgfrememberpicturerepositiononpagetrue	822, 915,
\msg_new:nnn	16		984, 1903, 1927, 1945, 1976, 2002, 2025,
\msg_new:nnnn	17		2376, 2851, 3403, 3782, 3903, 4138, 4185,
\msg_redirect_name:nnn	19		4312, 4322, 4333, 4711, 4858, 5006, 5060, 5173
\multicolumn	1098, 3143, 3195, 3201, 3222	\pgfscope	2891, 5023
\multispan	123, 124, 140, 141	\pgfset	379, 404, 916, 4801, 4835, 5022, 5081, 5175
\myfiledate	6	\pgfsetbaseline	914
\myfileversion	7	\pgfsetlinewidth	3816, 3935, 4887
<b>N</b>			
\newcolumnntype	223, 224, 225	\pgfsetrectcap	3817, 3936
\newcounter	308	\pgfsetroundcap	5019
\NewDocumentCommand	320, 343,	\pgfsetstrokecolor	4864
	727, 2343, 3036, 3052, 3068, 3084, 3108,	\pgfsyspdfmark	214, 215
	3209, 3285, 3363, 3442, 3457, 3472, 3493,	\pgftransformrotate	2898
	3498, 3520, 3663, 3668, 3677, 4923, 4963, 5153	\pgftransformshift	385, 410,
\NewDocumentEnvironment	1124,		2892, 4800, 4822, 5024, 5028, 5083, 5233, 5243
	1839, 1848, 2128, 2138, 2168, 2179, 2187, 4206	\pgfusepath	2918, 2928
\NewExpandableDocumentCommand	229, 4438	\pgfusepathqfill	3030, 3433, 3577, 3589, 3803, 3922
\newlist	324, 335	\pgfusepathqstroke	3818, 3937, 4888, 5020
\NiceArray	2184, 2193	\phantom	3049, 3065, 3081, 3105, 3128
\NiceArrayWithDelims	2133, 2143	\pNiceMatrix	4903
nicematrix commands:		prg commands:	
\g_nicematrix_code_after_tl	263, 588, 1863, 2321, 2323, 4676, 5045, 5053	\prg_do_nothing:	175, 184, 190, 218, 480, 1060, 2319
\g_nicematrix_code_before_tl	1122, 2325, 2334, 3665, 3670, 3681, 4688	\prg_new_conditional:Nnn	3604, 3612
\NiceMatrixLastEnv	229	\prg_replicate:nn	358, 1985,
\NiceMatrixOptions	727, 5698, 5898		1986, 3222, 3808, 3927, 4933, 4936, 4939, 4945
\NiceMatrixoptions	5580	\prg_return_false:	3609, 3621
\noalign	97, 104, 109, 132, 996, 1060, 3948, 4117	\prg_return_true:	3610, 3620
\nobreak	338	\ProcessKeysOptions	5347
\normalbaselines	1048	\ProvideDocumentCommand	58
\NotEmpty	1105	\ProvidesExplPackage	4
\nulldelimiterspace	1820, 1832, 5082, 5225	<b>Q</b>	
\nullfont	5093, 5100, 5264, 5271, 5288, 5295	\quad	339
\numexpr	152, 153	quark commands:	
<b>O</b>			
\omit	122, 1897, 1913, 1989, 5042	\q_stop	119, 120, 136, 137,
\OnlyMainNiceMatrix	1101, 3685		158, 160, 1169, 1418, 1486, 1619, 1858,
<b>P</b>			
\par	1716, 1724		1861, 3357, 3371, 3372, 3437, 3483, 3488,
peek commands:			3544, 3629, 3630, 3646, 3647, 4406, 4413,
\peek_meaning:Ntf	157, 350, 1017		4440, 4443, 4866, 4875, 4918, 4927, 5156, 5161
\peek_meaning_ignore_spaces:Ntf	1841, 3951	<b>R</b>	
\peek_meaning_remove_ignore_spaces:Ntf	147	\rectanglecolor	1214, 2327, 3672, 4690
\peek_remove_spaces:n	3167	\refstepcounter	367
\pgfextracty	4789	regex commands:	
\pgfgetlastxy	412	\regex_const:Nn	208
\pgfpathcircle	3024	\regex_match:nnTF	5139
\pgfpathlineto	3807, 3813, 3926, 3932, 5016	\regex_replace_all:NnN	1423
\pgfpathmoveto	3806, 3812, 3925, 3931, 5011	\relax	152, 153
\pgfpathrectanglecorners	3657, 3800, 3919, 4884	\renewcommand	194
\pgfpointadd	410	\RenewDocumentEnvironment	4902, 4905, 4908, 4911, 4914
\pgfpointanchor	232,	\RequirePackage	1, 3, 9, 10
	2553, 2564, 4269, 4277, 4734, 4752, 4771,	\right	1375, 1816, 1828, 5106, 5276, 5300
	4773, 4790, 4824, 5074, 5185, 5192, 5208, 5215	\rotate	1100
\pgfpointdiff	411, 1193, 1201	\rowcolor	1034, 1215
\pgfpointlineattime	2894	\rowcolors	1216
<b>S</b>			
\savenotes	1127		

scan commands:	
\scan_stop: . . . . .	2322
\scriptstyle . . . . .	
. . . . .	797, 2047, 2092, 2879, 2880, 2914, 2924
seq commands:	
\seq_clear_new:N . . . . .	1220, 2246, 4013
\seq_count:N . . . . .	1868, 3420
\seq_gclear:N . . . . .	
. . . . .	1154, 1155, 1156, 1157, 1744, 2320
\seq_gclear_new:N . . . . .	1108, 1109, 1864, 1880
\seq_gpop_left:NN . . . . .	1870, 1882
\seq_gput_left:Nn . . . . .	584, 3171, 3173, 4669
\seq_gput_right:Nn . . . . .	
. . . . .	349, 2533, 3174, 4563, 4575, 4683, 4994, 5145
\seq_gset_from_clist:Nn . . . . .	2249, 2261
\seq_gset_map_x:NNn . . . . .	2347
\seq_gset_split:Nnn . . . . .	1866, 1881
\seq_if_empty:NTF . . . . .	1704
\seq_if_empty_p:N . . . . .	2293, 2294, 2295
\seq_if_exist:NTF . . . . .	1172
\seq_if_in:NnTF . . . . .	
. . . . .	582, 3759, 3765, 3772, 3880,
. . . . .	3886, 3893, 4272, 4280, 4732, 4750, 5141, 5397
\seq_item:Nn . . . . .	1176, 1179, 1188, 1189, 1196, 1197
\seq_map_function:NN . . . . .	1872
\seq_map_indexed_inline:Nn . . . . .	3415, 3429
\seq_map_inline:Nn . . . . .	1722, 1728, 1884, 3564,
. . . . .	3720, 3722, 3724, 3842, 3844, 3846, 4098, 4637
\seq_mapthread_function:NNN . . . . .	4401
\seq_new:N . . . . .	
. . . . .	240, 283, 284, 285, 286, 287, 309, 5387
\seq_put_right:Nn . . . . .	3419, 4085
\seq_set_eq:NN . . . . .	3533
\seq_set_filter:NNn . . . . .	3535, 3562
\seq_set_from_clist:Nn . . . . .	5388
\seq_set_map_x:NNn . . . . .	5393
\seq_use:Nnnn . . . . .	2263, 5557, 5903
\l_tmpa_seq . . . . .	3535, 3562
\l_tmpb_seq . . . . .	3533, 3535, 3562, 3564
\setlist . . . . .	325, 336, 620, 625, 636, 641
skip commands:	
\skip_gadd:Nn . . . . .	1961
\skip_gset:Nn . . . . .	1952
\skip_gset_eq:NN . . . . .	1959, 1960
\skip_horizontal:N . . . . .	
. . . . .	128, 145, 1292, 1293, 1304, 1305, 1329,
. . . . .	1330, 1366, 1367, 1370, 1371, 1384, 1385,
. . . . .	1414, 1627, 1640, 1833, 1834, 1836, 1837,
. . . . .	1908, 1909, 1921, 1923, 1939, 1941, 1963,
. . . . .	1970, 1972, 1991, 1996, 1998, 2017, 2018,
. . . . .	2075, 2076, 2077, 2080, 2114, 2119, 2120, 2121
\skip_horizontal:n . . . . .	370, 1533
\skip_vertical:N . . . . .	
. . . . .	132, 978, 980, 1362, 1373, 1713, 1738, 4117
\skip_vertical:n . . . . .	859, 3958
\g_tmpa_skip . . . . .	1952, 1959, 1960, 1961, 1963, 1991
\c_zero_skip . . . . .	1065
\space . . . . .	260, 261
\stepcounter . . . . .	356, 361
str commands:	
\c_backslash_str . . . . .	260
\c_colon_str . . . . .	726
\str_case:nn . . . . .	3150, 4793, 4805, 4827, 4839
\str_case:nnTF . . . . .	1334, 1462, 1678, 4016
\str_clear_new:N . . . . .	5170
\str_foldcase:n . . . . .	3150
\str_gclear:N . . . . .	2339
\str_gset:Nn . . . . .	
. . . . .	1137, 2132, 2141, 2170, 2181, 2189, 4954
\str_if_empty:NTF . . . . .	
. . . . .	826, 927, 987, 1136, 1240, 1258,
. . . . .	1905, 1930, 1948, 1979, 2005, 2028, 2131,
. . . . .	2140, 2210, 2231, 4393, 4420, 5226, 5239, 5249
\str_if_eq:nnTF . . . . .	83, 259, 967,
. . . . .	1489, 1526, 1634, 1853, 1855, 1888, 4862, 5050
\str_if_eq_p:nn . . . . .	
. . . . .	469, 1493, 1616, 1617, 1618, 1619, 4448, 4453
\str_if_in:NnTF . . . . .	1666, 1770
\str_new:N . . . . .	255, 451, 725
\str_range:Nnn . . . . .	1670, 1774
\str_set:Nn . . . . .	581, 714, 5144
\str_set_eq:NN . . . . .	585, 726
\l_tmpa_str . . . . .	581, 582, 584, 585
\strut . . . . .	1722, 1728
\strutbox . . . . .	5199, 5202
\SubMatrix . . . . .	2318, 5498, 5530, 5550
T	
\tabcolsep . . . . .	960, 1366, 1370
\tabskip . . . . .	1065
\tablarnote . . . . .	320, 343, 350, 5608, 5621
\tablarnotes . . . . .	1727
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\@BTnormal . . . . .	1040
\@acol . . . . .	950
\@acoll . . . . .	948
\@acolr . . . . .	949
\@array@array . . . . .	952
\@arrayacol . . . . .	948, 949, 950
\@arstrutbox . . . . .	819, 820,
. . . . .	859, 1067, 1069, 1071, 1074, 1076, 1558, 1561
\@currenvir . . . . .	5050
\@depth . . . . .	5103, 5273, 5297
\@gobblethree . . . . .	215
\@halignto . . . . .	951, 963, 964
\@height . . . . .	112, 127, 144, 5101, 5272, 5296
\@ifclassloaded . . . . .	50, 53, 5368, 5378
\@ifnextchar . . . . .	1113
\@ifpackageloaded . . . . .	
. . . . .	29, 32, 35, 38, 74, 90, 165, 5371, 5381
\@mainaux . . . . .	75, 211, 2204, 2205, 2212,
. . . . .	2218, 2225, 2226, 2233, 2239, 2244, 2245,
. . . . .	2247, 2265, 2329, 2330, 2336, 4223, 4224, 4230
\@tabarray . . . . .	965
\@tempswafalse . . . . .	1409
\@tempswattrue . . . . .	1408
\@temptokena . . . . .	174, 177, 196, 198, 1407, 1418
\@whiles . . . . .	1409
\@width . . . . .	112, 5104, 5274, 5298
\@xhline . . . . .	115
\bBigg@ . . . . .	1283, 1285
\c@MaxMatrixCols . . . . .	2160, 5425
\c@tablarnote . . . . .	1706, 1717, 1745
\col@sep . . . . .	960, 961, 1329, 1385, 1908,
. . . . .	1961, 2017, 2080, 2114, 2597, 2606, 2639, 2646
\CT@arc . . . . .	94, 95



<code>\VNiceMatrix</code> .....	4909		<b>X</b>
<code>\vNiceMatrix</code> .....	4906	<code>\xglobal</code> .....	804, 811, 2057, 2100
<code>\vrule</code> .....	111		<b>Z</b>
<code>\vskip</code> .....	110	<code>\Z</code> .....	5139
<code>\vtop</code> .....	976		

## Contents

<b>1</b>	<b>The environments of this package</b>	<b>2</b>
<b>2</b>	<b>The vertical space between the rows</b>	<b>2</b>
<b>3</b>	<b>The vertical position of the arrays</b>	<b>3</b>
<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>hvlines</code> .....	8
5.5	The key <code>hvlines-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	Customization 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> .....	21
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 Tikz 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 “row-nodes” and the “col-nodes” . . . . .	32
13.4	The nodes corresponding to the command <code>\SubMatrix</code> . . . . .	33
<b>14</b>	<b>API for the developpers</b>	<b>33</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 <code>exterior-arraycolsep</code> . . . . .	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 . . . . .	39
16.4	Width of the columns . . . . .	40
16.5	How to highlight cells of a matrix . . . . .	41
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>178</b>
	<b>Index</b>	<b>184</b>