

# The package `nicematrix`\*

F. Pantigny  
fpantigny@wanadoo.fr

January 1, 2021

## Abstract

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

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

Product	dimensions (cm)			Price
	L	l	h	
small	3	5.5	1	30
standard	5.5	8	1.5	50.5
premium	8.5	10.5	2	80
extra	8.5	10	1.5	85.5
special	12	12	0.5	70

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

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 5.8 of `nicematrix`, at the date of 2021/01/01.

# 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`. 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-top-limit = 1pt,cell-space-bottom-limit = 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-top-limit=1pt,cell-space-bottom-limit=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.

**New 5.6** If this argument is empty, its default value is 1-1. If the number of rows is not specified, the block extends until the last row (idem for the columns).

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

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

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

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

One may wish to raise the size of the “A” placed in the block of the previous example. Since this element is composed in math mode, it's not possible to use directly a command like `\large`, `\Large` and `\LARGE`. That's why the command `\Block` provides an option between angle brackets to specify some TeX code which will be inserted before the beginning of the math mode.<sup>2</sup>

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

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

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

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

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

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

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

<sup>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			muguet
	iris	jacinthe	

**New 5.7** 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).

```

\begin{NiceTabular}{@{}>{\bfseries}lr@{}} \hline
\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}

```

<b>John</b>	12
	13
<b>Steph</b>	8
	18
<b>Sarah</b>	17
	15
<b>Ashley</b>	20
<b>Henry</b>	14
<b>Madison</b>	15
	19

## 4.3 The mono-row blocks

**New 5.6** 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
Write & \Block[1]{year of birth} \\
\midrule
Hugo & 1802 \\
Balzac & 1799 \\
\bottomrule
\end{NiceTabular}
```

Write	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. 33):

```
\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 \\
A&\cline{2-2}&C&D \\
A&B&C&D \\ \hline
\end{tabular}
```

A	B	C	D
A	<u>B</u>	C	D
A	B	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 \\
A&\cline{2-2}&C&D \\
A&B&C&D \\ \hline
\end{NiceTabular}
```

A	B	C	D
A	<u>B</u>	C	D
A	B	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 33.

## 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`.

*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

A command `\cellcolor` generates only one instruction `fill` (coded `f`) in the resulting PDF.

- 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).

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

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

A command `\rowcolor` generates only one instruction `fill` (coded `f`) in the resulting PDF.

- 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 tow 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*– 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{1-3}{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.

<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.

```


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


```

We have used the key `r` which aligns all the columns rightwards (cf. p. 27).

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{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}

```

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

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

Each instruction `\cellcolor`, `\rowcolor` or `\columncolor` will generate an instruction `fill` (coded `f`) in the resulting PDF. In cases of juxtaposed colored rectangles, one may have a thin white color line in some PDF viewers<sup>12</sup> (between the two first columns in the above example). In you want to avoid this problem, you should use the tools in the `code-before`. That's what we do with the following code.

```
\begin{NiceTabular}[colortbl-like]{ccc}%
  [code-before = \columncolor{blue!15}{1,2}]
\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

## 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).

<sup>12</sup>For example SumatraPDF, which uses MuPDF of Artifex Software, or PDF.js used by Firefox.

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

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

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

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

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

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

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

```
\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}
```

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

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

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}
```

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

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

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

The dotted lines have been drawn with the tools presented p. 17.

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. 29) 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}}
$\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col,nullify-dots]
    & C_1 & & \Cdots & & & C_4 & & \\
L_1 & & a_{11} & & a_{12} & & a_{13} & & a_{14} & & L_1 \\
\vdots & & a_{21} & & a_{22} & & a_{23} & & a_{24} & & \vdots \\
\hline
& & a_{31} & & a_{32} & & a_{33} & & a_{34} & & \\
L_4 & & a_{41} & & a_{42} & & a_{43} & & a_{44} & & L_4 \\
& & C_1 & & \Cdots & & C_4 & & & & \\
\end{pNiceArray}$

```

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



## 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 33.

- 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>15</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>16</sup> on both sides of the current cell. Of course, for `\Ldots` and `\Cdots`, it's an horizontal line; for `\Vdots`, it's a vertical line and for `\Ddots` and `\iddots` diagonal ones. It's possible to change the color of these lines with the option `color`.<sup>17</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}
```

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

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}
```

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

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}
```

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

<sup>15</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>16</sup>The precise definition of a “non-empty cell” is given below (cf. p. 34).

<sup>17</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.

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

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

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>18</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}
```

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

## 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.

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

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

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).

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

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

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

<sup>18</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

## 9.2 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.

```


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


```

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

```


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


```

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`:

```


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


```

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

### 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>19</sup>

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>15</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 `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 `code-after` which is described p. 22) accept two optional arguments specified by the tokens `_` and `^` for labels positionned below and above the line. The arguments are composed in math mode with `\scriptstyle`.

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

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

### 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 `code-after` which is described p. 22) may be customized by three options (specified between square brackets after the command):

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

These options may also be fixed with `\NiceMatrixOptions` 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`;

<sup>19</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.

- `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>20</sup>

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

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

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

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

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

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

## 9.6 The dotted lines and the rules

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

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

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

<sup>20</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.

## 10 The code-after

The option `code-after` may be used to give some code that will be executed after the construction of the matrix.<sup>21</sup>

A special command, called `\line`, is available to draw 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.

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

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

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` (for an example, cf. p. 39). **New 5.5** Before the version 5.5, it was necessary, in some circumstances, to put the keyword `\omit` before `\CodeAfter`. Since version 5.5, one must never put `\omit`.

## 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.}
```

<sup>21</sup>There is also a key `code-before` described p. 11.

```

& 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. 24. This table has been composed with the following code.

Table 1: Use of `\tablarnote`<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 tablarnote}\tablarnote{It's possible
to put a note in the caption.}}
\label{t:tablarnote}
\begin{NiceTabular}{@{}llc@{}
[notes/bottomrule, tablarnote = Some text before the notes.]
\toprule
Last name & First name & Length of life \\
\midrule
Churchill & Wiston & 91\\
Nightingale\tablarnote{Considered as the first nurse of
history.}\tablarnote{Nicknamed ``the Lady with the Lamp''.}
& Florence & 90 \\
Schoelcher & Victor & 89\tablarnote{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. 24).

- 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. 35.

## 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`.

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

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

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}
```

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

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

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

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

## 12.4 The option small

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

```
\begin{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>22</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 `code-after` (cf. p. 22), `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).

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

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

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

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

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

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

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

---

<sup>22</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.

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

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

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>23</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.

**New 5.6** However, it's possible to impose the creation of a node with the command `\NotEmpty`.<sup>24</sup>

<sup>23</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>24</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. 17).

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 `code-after`, 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. 39).

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

## 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>25</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>26</sup>

<sup>25</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>26</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>27</sup>

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

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

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

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

Here is an array composed with the following code:

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

### 13.3 The “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 `code-after`.

<sup>27</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.

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

If we use Tikz (we remind that `nicematrix` does not load Tikz by default), we can access (in the `code-before` and the `code-after`) 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

## 14 API for the developpers

The package `nicematrix` provides two variables which are internal but public<sup>28</sup>:

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

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

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

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

<sup>28</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.



```

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 { ! 0 { 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>29</sup>:

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

The heavy vertical rule won't extend in the exterior rows.<sup>30</sup>

```

$\begin{pNiceArray}{cc?cc}[first-row,last-row=3]
C_1 & C_2 & C_3 & C_4 \\
a & b & c & d \\
e & f & g & h \\
C_1 & C_2 & C_3 & C_4
\end{pNiceArray}$

```

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

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.

<sup>29</sup>The command `\vrule` is a TeX (and not LaTeX) command.

<sup>30</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`.

## 15.2 Diagonal lines

By default, all the diagonal lines<sup>31</sup> of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That’s why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

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

## 15.3 The “empty” cells

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

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

```
\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`.

<sup>31</sup>We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

## 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>32</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>33</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}` of `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`).

# 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. 22.

Let's consider that we wish to number the notes of a tabular with stars.<sup>34</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>35</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 =
```

---

<sup>32</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>33</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

<sup>34</sup>Of course, it's realistic only when there is very few notes in the tabular.

<sup>35</sup>In fact: the value of its argument.

```

    {
      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.

## 16.2 Dotted lines

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

```

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

```

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

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}
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \dots\dots\dots 10 \text{ other rows } \dots\dots\dots \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{pmatrix}$$

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}
```

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

An example for the resultant of two polynomials:

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

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

An example for a linear system:

```

 $\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 0 & \\ 0 & 1 & 0 & \cdots & 0 & & L_2 - L_1 \\ 0 & 0 & 1 & \ddots & \vdots & & L_3 - L_1 \\ & & & \ddots & \vdots & \vdots & \\ \vdots & & & \ddots & 0 & & \\ 0 & & & \cdots & 0 & 1 & 0 & L_n - L_1 \end{pmatrix}$ 

```

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

## 16.3 Dotted lines which are no longer dotted

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

```

\NiceMatrixOptions
{nullify-dots,code-for-first-col = \color{blue},code-for-first-col=\color{blue}}
 $\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}$ 

```

$$\begin{array}{c} \xrightarrow{n \text{ columns}} \\ \uparrow n \text{ rows} \end{array} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix}$$

## 16.4 Width of the columns

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

```

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

```

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

## 16.5 How to highlight cells of a matrix

In order to highlight a cell of a matrix, it's possible to “draw” that cell with the key `draw` of the command `\Block` (this is one of the uses of a mono-cell block<sup>36</sup>).

```

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

```

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

<sup>36</sup>We recall that, if the first mandatory argument of the command `\Block` is left empty, that means that the block is a mono-cell block

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the command `\hline`, the specifier “|” and the options `hlines`, `vlines` and `hvlines` spread the cells.<sup>37</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 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`.

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

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

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

---

<sup>37</sup>For the command `\cline`, see the remark p. 7.



```

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

```

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

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.

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

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} \backslash \backslash
A_{21} \& A_{22} \& A_{23} \& A_{24} \backslash \backslash
A_{31} \& A_{32} \& A_{33} \& A_{34} \backslash \backslash
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}
&

```

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} & \\
\& Cdots & & \& Cdots & & \& Cdots & \\
& & b_{kj} & & \\
& & \& Cdots & \\
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} & \\
& \& Cdots & & & & \& Cdots & \\
L_i & a_{i1} & \& Cdots & a_{ik} & \& Cdots & a_{in} & \\
& \& Cdots & & & & \& Cdots & \\
& a_{n1} & \& Cdots & & & a_{nn} & \\
\end{bNiceArray}
&

```

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

```

\begin{bNiceArray}{cc>{\strut}ccc}
& & & & \\
& & \& Cdots & \\
\& Cdots & & c_{ij} & \\
& & & & \\
& & & & \\
\end{bNiceArray}

```

```

\end{array}$

\end{NiceMatrixBlock}

\begin{tikzpicture}[remember picture, overlay]
\node [highlight = (A-3-1) (A-3-5) ] {} ;
\node [highlight = (B-1-3) (B-5-3) ] {} ;
\draw [color = gray] (A-3-3) to [bend left] (B-3-3) ;
\end{tikzpicture}

```

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

## 17 Implementation

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

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

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

Of course, the package `nicematrix` uses the features of the package `array`. It tries to be independent of its implementation. Unfortunately, it was not possible to be strictly independent: 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 \RequirePackage { xparse }

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

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

```

## Technical definitions

```

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

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

```

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.

```

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

```

We test whether the current class is `revtex4-1` or `revtex4-2` because these classes redefines `\array` (of `array`) in a way incompatible with our programming.

```

46 \bool_new:N \c_@@_revtex_bool
47 \ifclassloaded { revtex4-1 }
48 { \bool_set_true:N \c_@@_revtex_bool }
49 { }

```

```

50 \ifclassloaded { revtex4-2 }
51 { \bool_set_true:N \c_@@_revtex_bool }
52 { }

```

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.

```

53 \ProvideDocumentCommand \iddots { }
54 {
55   \mathinner
56   {
57     \tex_mkern:D 1 mu
58     \box_move_up:nn { 1 pt } { \hbox:n { . } }
59     \tex_mkern:D 2 mu
60     \box_move_up:nn { 4 pt } { \hbox:n { . } }
61     \tex_mkern:D 2 mu
62     \box_move_up:nn { 7 pt }
63     { \vbox:n { \kern 7 pt \hbox:n { . } } }
64     \tex_mkern:D 1 mu
65   }
66 }

```

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.

```

67 \AtBeginDocument
68 {
69   \ifpackageloaded { booktabs }
70   { \iow_now:Nn \mainaux \nicematrix@redefine@check@rerun }
71   { }
72 }
73 \cs_set_protected:Npn \nicematrix@redefine@check@rerun
74 {
75   \cs_set_eq:NN \@@_old_pgful@check@rerun \pgfutil@check@rerun

```

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

```

76   \cs_set_protected:Npn \pgfutil@check@rerun ##1 ##2
77   {
78     \str_if_eq:eeF { nm- } { \tl_range:nnn { ##1 } 1 3 }
79     { \@@_old_pgful@check@rerun { ##1 } { ##2 } }
80   }
81 }

```

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

```

82 \bool_new:N \c_@@_colortbl_loaded_bool
83 \AtBeginDocument
84 {
85   \ifpackageloaded { colortbl }
86   { \bool_set_true:N \c_@@_colortbl_loaded_bool }
87   { }

```

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

```

88   \cs_set_protected:Npn \CT@arc@ { }
89   \cs_set:Npn \arrayrulecolor #1 # { \CT@arc@ { #1 } }
90   \cs_set:Npn \CT@arc@ #1 #2
91   {
92     \dim_compare:nNt \baselineskip = \c_zero_dim \noalign
93     { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }

```

```

94     }
Idem for \CT@drs@.
95     \cs_set_protected:Npn \CT@drsc@ { }
96     \cs_set:Npn \doublerulesepcolor #1 # { \CT@drs { #1 } }
97     \cs_set:Npn \CT@drs #1 #2
98     {
99         \dim_compare:nNnT \baselineskip = \c_zero_dim \noalign
100         { \cs_gset:Npn \CT@drsc@ { \color #1 { #2 } } }
101     }
102     \cs_set:Npn \hline
103     {
104         \noalign { \ifnum 0 = ` } \fi
105         \cs_set_eq:NN \hskip \vskip
106         \cs_set_eq:NN \vrule \hrule
107         \cs_set_eq:NN \@width \@height
108         { \CT@arc@ \vline }
109         \futurelet \reserved@a
110         \@xhline
111     }
112 }
113 }

```

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.

```

114 \cs_set:Npn \@@_standard_cline #1 { \@@_standard_cline:w #1 \q_stop }
115 \cs_set:Npn \@@_standard_cline:w #1-#2 \q_stop
116 {
117     \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
118     \int_compare:nNnT { #1 } > 1 { \multispan { \@@_pred:n { #1 } } & }
119     \multispan { \int_eval:n { #2 - #1 + 1 } }
120 {
121     \CT@arc@
122     \leaders \hrule \@height \arrayrulewidth \hfill

```

The following `\skip_horizontal:N \c_zero_dim` is to prevent a potential `\unskip` to delete the `\leaders`<sup>38</sup>

```

123 \skip_horizontal:N \c_zero_dim
124 }

```

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`.

```

125 \everycr { }
126 \cr
127 \noalign { \skip_vertical:N -\arrayrulewidth }
128 }

```

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.

```

129 \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`.

```

130 { \@@_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*.

```

131 \cs_set:Npn \@@_cline_i:nn #1 #2 { \@@_cline_i:w #1-#2 \q_stop }
132 \cs_set:Npn \@@_cline_i:w #1-#2-#3 \q_stop
133 {

```

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

---

<sup>38</sup>See question 99041 on TeX StackExchange.

```

134 \int_compare:nNnT { #1 } < { #2 }
135 { \multispan { \int_eval:n { #2 - #1 } } & }
136 \multispan { \int_eval:n { #3 - #2 + 1 } }
137 {
138   \CT@arc@
139   \leaders \hrule \@height \arrayrulewidth \hfill
140   \skip_horizontal:N \c_zero_dim
141 }

```

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

```

142 \peek_meaning_remove_ignore_spaces:NTF \cline
143 { & \@@_cline_i:en { \@@_succ:n { #3 } } }
144 { \everycr { } \cr }
145 }
146 \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.

```

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

```

The following command is a small shortcut.

```

149 \cs_new:Npn \@@_math_toggle_token:
150 { \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token }

151 \cs_new_protected:Npn \@@_set_CT@arc@:
152 { \peek_meaning:NTF [ \@@_set_CT@arc@_i: \@@_set_CT@arc@_ii: }
153 \cs_new_protected:Npn \@@_set_CT@arc@_i: [ #1 ] #2 \q_stop
154 { \cs_set:Npn \CT@arc@ { \color [ #1 ] { #2 } } }
155 \cs_new_protected:Npn \@@_set_CT@arc@_ii: #1 \q_stop
156 { \cs_set:Npn \CT@arc@ { \color { #1 } } }

```

## The column S of siunitx

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

```

157 \bool_new:N \c_@@_siunitx_loaded_bool
158 \AtBeginDocument
159 {
160   \@ifpackageloaded { siunitx }
161   { \bool_set_true:N \c_@@_siunitx_loaded_bool }
162   { }
163 }

```

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:`.

```

164 \cs_set_protected:Npn \@@_adapt_S_column:
165 {
166   \bool_if:NT \c_@@_siunitx_loaded_bool
167   {
168     \group_begin:
169     \@temptokena = { }

```

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

```

170   \cs_set_eq:NN \NC@find \prg_do_nothing:
171   \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`).

```

172   \tl_gset:NV \g_tmpa_tl \@temptokena
173   \group_end:
174   \tl_new:N \c_@@_table_collect_begin_tl
175   \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
176   \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
177   \tl_new:N \c_@@_table_print_tl
178   \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).

```

179   \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
180 }
181 }

```

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

```

182 \AtBeginDocument
183 {
184   \bool_if:nTF { ! \c_@@_siunitx_loaded_bool }
185   { \cs_set_eq:NN \@@_renew_NC@rewrite@S: \prg_do_nothing: }
186   {
187     \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
188     {
189       \renewcommand*{\NC@rewrite@S}[1] []
190       {
191         \@temptokena \exp_after:wN
192         {
193           \tex_the:D \@temptokena
194           > { \@@_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.

```

195         \@@_true_c:
196         < { \c_@@_table_print_tl \@@_end_Cell: }
197     }
198     \NC@find
199 }
200 }
201 }
202 }
```

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

```

203 \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.

```

204 \cs_new_protected:Npn \@@_provide_pgfsyspdfmark:
205 {
206     \iow_now:Nn \@mainaux
207     {
208         \ExplSyntaxOn
209         \cs_if_free:NT \pgfsyspdfmark
210         { \cs_set_eq:NN \pgfsyspdfmark \@gobblethree }
211         \ExplSyntaxOff
212     }
213     \cs_gset_eq:NN \@@_provide_pgfsyspdfmark: \prg_do_nothing:
214 }
```

## 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.

```

215 \bool_new:N \c_@@_define_L_C_R_bool
216 \cs_new_protected:Npn \@@_define_L_C_R:
217 {
218     \newcolumntype L l
219     \newcolumntype C c
220     \newcolumntype R r
221 }
```

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.

```

222 \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).

```

223 \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.

```

224 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
225 { \int_use:N \g_@@_env_int }
```

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

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

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

```
228 \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).

```
229 \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.

```
230 \tl_new:N \l_@@_cell_type_tl
231 \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`.

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

Idem pour the blocks mono-row.

```
233 \dim_new:N \g_@@_blocks_ht_dim
234 \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`.

```
235 \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.

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

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

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

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

```
238 \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.

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

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

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

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

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

```

242 \cs_new_protected:Npn \@@_test_if_math_mode:
243 {
244   \if_mode_math: \else:
245     \@@_fatal:n { Outside-math-mode }
246   \fi:
247 }

```

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

```

248 \colorlet { nicematrix-last-col } { . }
249 \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*).

```

250 \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*.

```

251 \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).

```

252 \cs_new:Npn \@@_full_name_env:
253 {
254   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
255     { command \space \c_backslash_str \g_@@_name_env_str }
256     { environment \space \{ \g_@@_name_env_str \} }
257 }

```

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`).

```

258 \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.

```

259 \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.

```

260 \int_new:N \l_@@_old_iRow_int
261 \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.

```

262 \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.

```

263 \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).

```
264 \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).

```
265 \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`.

```
266 \tl_new:N \l_@@_code_before_tl
267 \bool_new:N \l_@@_code_before_bool
```

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

```
268 \dim_new:N \l_@@_x_initial_dim
269 \dim_new:N \l_@@_y_initial_dim
270 \dim_new:N \l_@@_x_final_dim
271 \dim_new:N \l_@@_y_final_dim
```

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

```
272 \dim_zero_new:N \l_tmpc_dim
273 \dim_zero_new:N \l_tmpd_dim
```

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

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

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

```
275 \dim_new:N \@@_old_arraycolsep_dim
```

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

```
276 \dim_new:N \g_@@_width_last_col_dim
277 \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.

```
278 \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}`.

```
279 \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}`.

```
280 \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.

```
281 \seq_new:N \g_@@_pos_of_stroken_blocks_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).

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

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

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

```
284 \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).

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

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

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

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

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

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

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

```
289 \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.

```
290 \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.

```
291 \int_new:N \l_@@_first_row_int
```

```
292 \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.

```
293 \int_new:N \l_@@_first_col_int
294 \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).

```
295 \int_new:N \l_@@_last_row_int
296 \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>39</sup>

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

Idem for `\l_@@_last_col_without_value_bool`

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

- **Last column**

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

```
299 \int_new:N \l_@@_last_col_int
300 \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.

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

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

---

<sup>39</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.

## The command `\tabularnote`

The LaTeX counter `tabularnote` 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.

```
302 \newcounter { tabularnote }
```

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

```
303 \seq_new:N \g_@@_tabularnotes_seq
```

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

```
304 \tl_new:N \l_@@_tabularnote_tl
```

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

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

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

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

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

```
307 \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`.

```
308 \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`.

```
309 \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`).

```
310 \AtBeginDocument
311 {
312   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
313   {
314     \NewDocumentCommand \tabularnote { m }
315     { \@@_error:n { enumitem-not-loaded } }
316   }
317   {
```

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.

```
318     \newlist { tabularnotes } { enumerate } { 1 }
319     \setlist [ tabularnotes ]
320     {
321       topsep = Opt ,
322       noitemsep ,
323       leftmargin = * ,
```

```

324         align = left ,
325         labelsep = Opt ,
326         label =
327             \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
328     }
329     \newlist { tabularnotes* } { enumerate* } { 1 }
330     \setlist [ tabularnotes* ]
331     {
332         afterlabel = \nobreak ,
333         itemjoin = \quad ,
334         label =
335             \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
336     }

```

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>40</sup>

```

337     \NewDocumentCommand \tabularnote { m }
338     {
339         \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
340         { \@@_error:n { tabularnote~forbidden } }
341         {

```

`\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 these notes as a comma separated list (e.g.  $a,b,c$ ).

```

342         \int_incr:N \l_@@_number_of_notes_int

```

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

```

343         \seq_gput_right:Nn \g_@@_tabularnotes_seq { #1 }
344         \peek_meaning:NF \tabularnote
345         {

```

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.

```

346         \hbox_set:Nn \l_tmpa_box
347         {

```

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

```

348             \@@_notes_label_in_tabular:n
349             {
350                 \stepcounter { tabularnote }
351                 \@@_notes_style:n { tabularnote }
352                 \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
353                 {
354                     ,
355                     \stepcounter { tabularnote }
356                     \@@_notes_style:n { tabularnote }
357                 }
358             }
359         }

```

---

<sup>40</sup>We should try to find a solution to that problem.



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.

```

360             \addtocounter { tabularnote } { -1 }
361             \refstepcounter { tabularnote }
362             \int_zero:N \l_@@_number_of_notes_int
363             \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).

```

364             \skip_horizontal:n { \box_wd:N \l_tmpa_box }
365         }
366     }
367 }
368 }
369 }

```

## 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.

```

370 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
371 {
372     \begin { pgfscope }
373     \pgfset
374     {
375         outer~sep = \c_zero_dim ,
376         inner~sep = \c_zero_dim ,
377         minimum~size = \c_zero_dim
378     }
379     \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
380     \pgfnode
381     { rectangle }
382     { center }
383     {
384         \vbox_to_ht:nn
385         { \dim_abs:n { #5 - #3 } }
386         {
387             \vfill
388             \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
389         }
390     }
391     { #1 }
392     { }
393     \end { pgfscope }
394 }

```

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

```

395 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
396 {
397     \begin { pgfscope }
398     \pgfset
399     {
400         outer~sep = \c_zero_dim ,
401         inner~sep = \c_zero_dim ,
402         minimum~size = \c_zero_dim
403     }
404     \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }

```

```

405 \pgfpointdiff { #3 } { #2 }
406 \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
407 \pgfnode
408 { rectangle }
409 { center }
410 {
411   \vbox_to_ht:nn
412   { \dim_abs:n \l_tmpb_dim }
413   { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
414 }
415 { #1 }
416 { }
417 \end { pgfscope }
418 }

```

## 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.

```

419 \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`.

```

420 \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`).

```

421 \dim_new:N \l_@@_cell_space_top_limit_dim
422 \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.

```

423 \dim_new:N \l_@@_inter_dots_dim
424 \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em }

```

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).

```

425 \dim_new:N \l_@@_xdots_shorten_dim
426 \dim_set:Nn \l_@@_xdots_shorten_dim { 0.3 em }

```

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.

```

427 \dim_new:N \l_@@_radius_dim
428 \dim_set:Nn \l_@@_radius_dim { 0.53 pt }

```

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.

```

429 \tl_new:N \l_@@_xdots_line_style_tl
430 \tl_const:Nn \c_@@_standard_tl { standard }
431 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl

```

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

```
432 \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).

```
433 \str_new:N \l_@@_baseline_tl
434 \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`).

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

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

```
436 \bool_new:N \l_@@_parallelize_diags_bool
437 \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.

```
438 \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.

```
439 \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`.

```
440 \clist_new:N \l_@@_except_corners_clist
441 \dim_new:N \l_@@_notes_above_space_dim
442 \dim_set:Nn \l_@@_notes_above_space_dim { 1 mm }
```

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.).

```
443 \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).

```
444 \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.

```
445 \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”.

```
446 \bool_new:N \l_@@_medium_nodes_bool
447 \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`).

```
448 \dim_new:N \l_@@_left_margin_dim
449 \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`.

```
450 \dim_new:N \l_@@_extra_left_margin_dim
451 \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.

```
452 \tl_new:N \l_@@_end_of_row_tl
453 \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 “:”.

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

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

```
455 \tl_new:N \l_@@_delimiters_color_tl
```

Sometimes, we want to have several arrays vertically juxtaposed in order to have an alignment of the columns of these arrays. To achieve this goal, one may wish to use the same width for all the columns (for example with the option `columns-width` or the option `auto-columns-width` of the environment `{NiceMatrixBlock}`). However, even if we use the same type of delimiters, the width of the delimiters may be different from an array to another because the width of the delimiter is fonction of its size. That’s why we create an option called `max-delimiter-width` which will give to the delimiters the width of a delimiter (of the same type) of big size. The following boolean corresponds to this option.

```
456 \bool_new:N \l_@@_max_delimiter_width_bool
```

```
457 \keys_define:nn { NiceMatrix / xdots }
458 {
459   line-style .code:n =
460   {
461     \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.

```
462   { \cs_if_exist_p:N \tikzpicture }
463   { \str_if_eq_p:nn { #1 } { standard } }
464   { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
465   { @@_error:n { bad-option-for~line-style } }
466   } ,
467   line-style .value_required:n = true ,
468   color .tl_set:N = \l_@@_xdots_color_tl ,
469   color .value_required:n = true ,
470   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
471   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 `_`.

```
472   down .tl_set:N = \l_@@_xdots_down_tl ,
473   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).

```
474   draw-first .code:n = \prg_do_nothing: ,
475   unknown .code:n = @@_error:n { Unknown-option-for~xdots }
476 }
```

```

477 \keys_define:nn { NiceMatrix / rules }
478 {
479   color .tl_set:N = \l_@@_rules_color_tl ,
480   color .value_required:n = true ,
481   width .dim_set:N = \arrayrulewidth ,
482   width .value_required:n = true
483 }

```

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

```

484 \keys_define:nn { NiceMatrix / Global }
485 {
486   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
487   standard-cline .default:n = true ,
488   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
489   cell-space-top-limit .value_required:n = true ,
490   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
491   cell-space-bottom-limit .value_required:n = true ,
492   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
493   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
494   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
495   light-syntax .default:n = true ,
496   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
497   end-of-row .value_required:n = true ,
498   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
499   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
500   last-row .int_set:N = \l_@@_last_row_int ,
501   last-row .default:n = -1 ,
502   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
503   code-for-first-col .value_required:n = true ,
504   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
505   code-for-last-col .value_required:n = true ,
506   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
507   code-for-first-row .value_required:n = true ,
508   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
509   code-for-last-row .value_required:n = true ,
510   hlines .bool_set:N = \l_@@_hlines_bool ,
511   vlines .bool_set:N = \l_@@_vlines_bool ,
512   hvlines .code:n =
513   {
514     \bool_set_true:N \l_@@_vlines_bool
515     \bool_set_true:N \l_@@_hlines_bool
516   } ,
517   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.

```

518   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
519   renew-dots .value_forbidden:n = true ,
520   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
521   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
522   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
523   create-extra-nodes .meta:n =
524   { create-medium-nodes , create-large-nodes } ,
525   left-margin .dim_set:N = \l_@@_left_margin_dim ,
526   left-margin .default:n = \arraycolsep ,
527   right-margin .dim_set:N = \l_@@_right_margin_dim ,
528   right-margin .default:n = \arraycolsep ,
529   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
530   margin .default:n = \arraycolsep ,
531   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
532   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,

```

```

533   extra-margin .meta:n =
534     { extra-left-margin = #1 , extra-right-margin = #1 } ,
535   extra-margin .value_required:n = true ,
536 }

```

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

```

537 \keys_define:nn { NiceMatrix / Env }
538 {
539   except-corners .clist_set:N = \l_@@_except_corners_clist ,
540   except-corners .default:n = { NW , SW , NE , SE } ,
541   hvlines-except-corners .code:n =
542     {
543       \clist_set:Nn \l_@@_except_corners_clist { #1 }
544       \bool_set_true:N \l_@@_vlines_bool
545       \bool_set_true:N \l_@@_hlines_bool
546     } ,
547   hvlines-except-corners .default:n = { NW , SW , NE , SE } ,
548   code-before .code:n =
549     {
550       \tl_if_empty:nF { #1 }
551       {
552         \tl_put_right:Nn \l_@@_code_before_tl { #1 }
553         \bool_set_true:N \l_@@_code_before_bool
554       }
555     } ,

```

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

```

556   c .code:n = \tl_set:Nn \l_@@_baseline_tl c ,
557   t .code:n = \tl_set:Nn \l_@@_baseline_tl t ,
558   b .code:n = \tl_set:Nn \l_@@_baseline_tl b ,
559   baseline .tl_set:N = \l_@@_baseline_tl ,
560   baseline .value_required:n = true ,
561   columns-width .code:n =
562     \tl_if_eq:nnTF { #1 } { auto }
563     { \bool_set_true:N \l_@@_auto_columns_width_bool }
564     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
565   columns-width .value_required:n = true ,
566   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.

```

567   \legacy_if:nF { measuring@ }
568   {
569     \str_set:Nn \l_tmpa_str { #1 }
570     \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
571     { \@@_error:nn { Duplicate-name } { #1 } }
572     { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
573     \str_set_eq:NN \l_@@_name_str \l_tmpa_str
574   } ,
575   name .value_required:n = true ,
576   code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
577   code-after .value_required:n = true ,
578   colortbl-like .code:n =
579     \bool_set_true:N \l_@@_colortbl_like_bool
580     \bool_set_true:N \l_@@_code_before_bool ,
581   colortbl-like .value_forbidden:n = true
582 }
583 \keys_define:nn { NiceMatrix / notes }
584 {
585   para .bool_set:N = \l_@@_notes_para_bool ,

```

```

586 para .default:n = true ,
587 code-before .tl_set:N = \l_@@_notes_code_before_tl ,
588 code-before .value_required:n = true ,
589 code-after .tl_set:N = \l_@@_notes_code_after_tl ,
590 code-after .value_required:n = true ,
591 bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
592 bottomrule .default:n = true ,
593 style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
594 style .value_required:n = true ,
595 label-in-tabular .code:n =
596   \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
597 label-in-tabular .value_required:n = true ,
598 label-in-list .code:n =
599   \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
600 label-in-list .value_required:n = true ,
601 enumitem-keys .code:n =
602   {
603     \bool_if:NTF \c_@@_in_preamble_bool
604     {
605       \AtBeginDocument
606       {
607         \bool_if:NT \c_@@_enumitem_loaded_bool
608         { \setlist* [ tabularnotes ] { #1 } }
609       }
610     }
611     {
612       \bool_if:NT \c_@@_enumitem_loaded_bool
613       { \setlist* [ tabularnotes ] { #1 } }
614     }
615   } ,
616 enumitem-keys .value_required:n = true ,
617 enumitem-keys-para .code:n =
618   {
619     \bool_if:NTF \c_@@_in_preamble_bool
620     {
621       \AtBeginDocument
622       {
623         \bool_if:NT \c_@@_enumitem_loaded_bool
624         { \setlist* [ tabularnotes* ] { #1 } }
625       }
626     }
627     {
628       \bool_if:NT \c_@@_enumitem_loaded_bool
629       { \setlist* [ tabularnotes* ] { #1 } }
630     }
631   } ,
632 enumitem-keys-para .value_required:n = true ,
633 unknown .code:n = \@@_error:n { Unknown~key~for~notes }
634 }

```

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

```

635 \keys_define:nn { NiceMatrix }
636 {
637   NiceMatrixOptions .inherit:n =
638     { NiceMatrix / Global } ,
639   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
640   NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
641   NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
642   NiceMatrix .inherit:n =
643     {
644       NiceMatrix / Global ,
645       NiceMatrix / Env ,

```

```

646     } ,
647     NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
648     NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
649     NiceTabular .inherit:n =
650     {
651         NiceMatrix / Global ,
652         NiceMatrix / Env
653     } ,
654     NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
655     NiceTabular / rules .inherit:n = NiceMatrix / rules ,
656     NiceArray .inherit:n =
657     {
658         NiceMatrix / Global ,
659         NiceMatrix / Env ,
660     } ,
661     NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
662     NiceArray / rules .inherit:n = NiceMatrix / rules ,
663     pNiceArray .inherit:n =
664     {
665         NiceMatrix / Global ,
666         NiceMatrix / Env ,
667     } ,
668     pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
669     pNiceArray / rules .inherit:n = NiceMatrix / rules ,
670 }

```

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

```

671 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
672 {
673     delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
674     delimiters-color .value_required:n = true ,
675     last-col .code:n = \tl_if_empty:nF { #1 }
676         { \@@_error:n { last-col~non-empty~for~NiceMatrixOptions } }
677         \int_zero:N \l_@@_last_col_int ,
678     small .bool_set:N = \l_@@_small_bool ,
679     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.

```

680     renew-matrix .code:n = \@@_renew_matrix: ,
681     renew-matrix .value_forbidden:n = true ,

```

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

```

682     transparent .meta:n = { renew-dots , renew-matrix } ,
683     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}`.

```

684     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.

```

685     columns-width .code:n =
686     \tl_if_eq:nnTF { #1 } { auto }
687     { \@@_error:n { Option~auto~for~columns-width } }
688     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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



```

689 allow-duplicate-names .code:n =
690   \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
691 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`).

```

692 letter-for-dotted-lines .code:n =
693   {
694     \tl_if_single_token:nTF { #1 }
695     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
696     { \@@_error:n { Bad-value~for~letter~for~dotted~lines } }
697   } ,
698 letter-for-dotted-lines .value_required:n = true ,
699 notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
700 notes .value_required:n = true ,
701 unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
702 }
703 \str_new:N \l_@@_letter_for_dotted_lines_str
704 \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.

```

705 \NewDocumentCommand \NiceMatrixOptions { m }
706 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

707 \keys_define:nn { NiceMatrix / NiceMatrix }
708 {
709   last-col .code:n = \tl_if_empty:nTF {#1}
710     {
711       \bool_set_true:N \l_@@_last_col_without_value_bool
712       \int_set:Nn \l_@@_last_col_int { -1 }
713     }
714     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
715   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
716   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
717   small .bool_set:N = \l_@@_small_bool ,
718   small .value_forbidden:n = true ,
719   delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
720   delimiters-color .value_required:n = true ,
721   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
722 }

```

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

```

723 \keys_define:nn { NiceMatrix / NiceArray }
724 {

```

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.

```

725   small .bool_set:N = \l_@@_small_bool ,
726   small .value_forbidden:n = true ,
727   last-col .code:n = \tl_if_empty:nF { #1 }
728     { \@@_error:n { last-col~non~empty~for~NiceArray } }
729     { \int_zero:N \l_@@_last_col_int ,
730   notes / para .bool_set:N = \l_@@_notes_para_bool ,

```

```

731 notes / para .default:n = true ,
732 notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
733 notes / bottomrule .default:n = true ,
734 tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
735 tabularnote .value_required:n = true ,
736 delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
737 delimiters-color .value_required:n = true ,
738 unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
739 }

740 \keys_define:nn { NiceMatrix / pNiceArray }
741 {
742   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
743   last-col .code:n = \tl_if_empty:nF {#1}
744     { \@@_error:n { last-col~non-empty~for~NiceArray } }
745     \int_zero:N \l_@@_last_col_int ,
746   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
747   small .bool_set:N = \l_@@_small_bool ,
748   small .value_forbidden:n = true ,
749   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
750 }

```

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

```

751 \keys_define:nn { NiceMatrix / NiceTabular }
752 {
753   notes / para .bool_set:N = \l_@@_notes_para_bool ,
754   notes / para .default:n = true ,
755   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
756   notes / bottomrule .default:n = true ,
757   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
758   tabularnote .value_required:n = true ,
759   last-col .code:n = \tl_if_empty:nF {#1}
760     { \@@_error:n { last-col~non-empty~for~NiceArray } }
761     \int_zero:N \l_@@_last_col_int ,
762   unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
763 }

```

## 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}).

```

764 \cs_new_protected:Npn \@@_Cell:
765 {

```

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`).

```

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

```

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

```

767   \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.

```

768   \int_compare:nNnT \c@jCol = 1
769     { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
770   \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).

```

771   \hbox_set:Nw \l_@@_cell_box
772   \bool_if:NF \l_@@_NiceTabular_bool
773   {
774     \c_math_toggle_token
775     \bool_if:NT \l_@@_small_bool \scriptstyle
776   }

```

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.

```

777   \int_compare:nNnTF \c@iRow = 0
778   {
779     \int_compare:nNnT \c@jCol > 0
780     {
781       \l_@@_code_for_first_row_tl
782       \xglobal \colorlet { nicematrix-first-row } { . }
783     }
784   }
785   {
786     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
787     {
788       \l_@@_code_for_last_row_tl
789       \xglobal \colorlet { nicematrix-last-row } { . }
790     }
791   }
792 }

```

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.

```

793 \cs_new_protected:Npn \@@_begin_of_row:
794 {
795   \int_gincr:N \c@iRow
796   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
797   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
798   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
799   \pgfpicture
800   \pgfrememberpicturepositiononpagetrue
801   \pgfcoordinate
802   { \@@_env: - row - \int_use:N \c@iRow - base }
803   { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
804   \str_if_empty:NF \l_@@_name_str
805   {
806     \pgfnodealias
807     { \l_@@_name_str - row - \int_use:N \c@iRow - base }
808     { \@@_env: - row - \int_use:N \c@iRow - base }
809   }
810   \endpgfpicture
811 }

```

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.

```

812 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
813 {
814   \int_compare:nNnTF \c@iRow = 0
815   {
816     \dim_gset:Nn \g_@@_dp_row_zero_dim

```

```

817         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
818     \dim_gset:Nn \g_@@_ht_row_zero_dim
819     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
820 }
821 {
822     \int_compare:nNnT \c@iRow = 1
823     {
824         \dim_gset:Nn \g_@@_ht_row_one_dim
825         { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
826     }
827 }
828 }

829 \cs_new_protected:Npn \@@_rotate_cell_box:
830 {
831     \box_rotate:Nn \l_@@_cell_box { 90 }
832     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
833     {
834         \vbox_set_top:Nn \l_@@_cell_box
835         {
836             \vbox_to_zero:n { }
837             \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
838             \box_use:N \l_@@_cell_box
839         }
840     }
841     \bool_gset_false:N \g_@@_rotate_bool
842 }

843 \cs_new_protected:Npn \@@_adjust_size_box:
844 {
845     \dim_compare:nNnT \g_@@_blocks_wd_dim > \c_zero_dim
846     {
847         \box_set_wd:Nn \l_@@_cell_box
848         { \dim_max:nn { \box_wd:N \l_@@_cell_box } \g_@@_blocks_wd_dim }
849         \dim_gzero:N \g_@@_blocks_wd_dim
850     }
851     \dim_compare:nNnT \g_@@_blocks_dp_dim > \c_zero_dim
852     {
853         \box_set_dp:Nn \l_@@_cell_box
854         { \dim_max:nn { \box_dp:N \l_@@_cell_box } \g_@@_blocks_dp_dim }
855         \dim_gzero:N \g_@@_blocks_dp_dim
856     }
857     \dim_compare:nNnT \g_@@_blocks_ht_dim > \c_zero_dim
858     {
859         \box_set_ht:Nn \l_@@_cell_box
860         { \dim_max:nn { \box_ht:N \l_@@_cell_box } \g_@@_blocks_ht_dim }
861         \dim_gzero:N \g_@@_blocks_ht_dim
862     }
863 }

864 \cs_new_protected:Npn \@@_end_Cell:
865 {
866     \@@_math_toggle_token:
867     \hbox_set_end:
868     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
869     \@@_adjust_size_box:
870     \box_set_ht:Nn \l_@@_cell_box
871     { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
872     \box_set_dp:Nn \l_@@_cell_box
873     { \box_dp:N \l_@@_cell_box + \l_@@_cell_space_bottom_limit_dim }

We want to compute in \g_@@_max_cell_width_dim the width of the widest cell of the array (except
the cells of the “first column” and the “last column”).

874     \dim_gset:Nn \g_@@_max_cell_width_dim
875     { \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”.

```
876 \@@_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`, `\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.

```
877 \bool_if:NTF \g_@@_empty_cell_bool
878 { \box_use_drop:N \l_@@_cell_box }
879 {
880   \bool_lazy_or:nnTF
881     \g_@@_not_empty_cell_bool
882     { \dim_compare_p:nNn { \box_wd:N \l_@@_cell_box } > \c_zero_dim }
883     \@@_node_for_the_cell:
884     { \box_use_drop:N \l_@@_cell_box }
885   }
886   \bool_gset_false:N \g_@@_empty_cell_bool
887   \bool_gset_false:N \g_@@_not_empty_cell_bool
888 }
```

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

```
889 \cs_new_protected:Npn \@@_node_for_the_cell:
890 {
891   \pgfpicture
892   \pgfsetbaseline \c_zero_dim
893   \pgfrememberpicturepositiononpagetrue
894   \pgfset
895   {
896     inner~sep = \c_zero_dim ,
897     minimum~width = \c_zero_dim
898   }
899   \pgfnode
900   { rectangle }
901   { base }
902   { \box_use_drop:N \l_@@_cell_box }
903   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
904   { }
905   \str_if_empty:NF \l_@@_name_str
906   {
907     \pgfnodealias
908     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
909     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
910   }
911   \endpgfpicture
912 }
```

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.

```
913 \cs_new_protected:Npn \@@_instruction_of_type:nnn #1 #2 #3
914 {
```

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).

```
915   \bool_if:NTF { #1 } \tl_gput_left:cx \tl_gput_right:cx
916   { \g_@@_ #2 _ lines _ tl }
917   {
918     \use:c { @@ _ draw _ #2 : nnn }
919     { \int_use:N \c@iRow }
920     { \int_use:N \c@jCol }
921     { \exp_not:n { #3 } }
922   }
923 }
```

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`.

```
924 \cs_new_protected:Npn \@@_revtex_array:
925 {
926   \cs_set_eq:NN \@acoll \@arrayacol
927   \cs_set_eq:NN \@acolr \@arrayacol
928   \cs_set_eq:NN \@acol \@arrayacol
929   \cs_set_nopar:Npn \@halignto { }
930   \@array@array
931 }

932 \cs_new_protected:Npn \@@_array:
933 {
934   \bool_if:NTF \c_@@_revtex_bool
935   \@_revtex_array:
936   {
937     \bool_if:NTF \l_@@_NiceTabular_bool
938     { \dim_set_eq:NN \col@sep \tabcolsep }
939     { \dim_set_eq:NN \col@sep \arraycolsep }
940     \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim
941     { \cs_set_nopar:Npn \@halignto { } }
942     { \cs_set_nopar:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }
```

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

```
943   \@tabarray
944 }
```

`\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.

```

945   [ \str_if_eq:VnTF \l_@@_baseline_tl c c t ]
946   }

```

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.

```

947 \cs_set_eq:NN \@@_old_ialign: \ialign

```

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

```

948 \cs_new_protected:Npn \@@_create_row_node:
949   {

```

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

```

950   \hbox
951   {
952     \bool_if:NT \l_@@_code_before_bool
953     {
954       \vtop
955       {
956         \skip_vertical:N 0.5\arrayrulewidth
957         \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
958         \skip_vertical:N -0.5\arrayrulewidth
959       }
960     }
961     \pgfpicture
962     \pgfrememberpicturepositiononpagetrue
963     \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
964     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
965     \str_if_empty:NF \l_@@_name_str
966     {
967       \pgfnodealias
968       { \l_@@_name_str - row - \int_use:N \c@iRow }
969       { \@@_env: - row - \int_use:N \c@iRow }
970     }
971     \endpgfpicture
972   }
973 }

```

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

```

974 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
975 \cs_new_protected:Npn \@@_everycr_i:
976   {
977     \int_gzero:N \c@jCol
978     \bool_gset_false:N \g_@@_after_col_zero_bool
979     \bool_if:NF \g_@@_row_of_col_done_bool
980     {
981       \@@_create_row_node:

```

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

```

982     \bool_if:NT \l_@@_hlines_bool
983     {

```

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.

```

984       \int_compare:nNnT \c@iRow > { -1 }
985       {
986         \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@`.

```

987         { \hrule height \arrayrulewidth width \c_zero_dim }
988     }
989 }
990 }
991 }
```

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`).

```

992 \cs_set_protected:Npn \@@_newcolumnntype #1
993 {
994     \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
995     \peek_meaning:NTF [
996         { \newcol@ #1 }
997         { \newcol@ #1 [ 0 ] }
998 }
```

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

```

999 \cs_set_protected:Npn \@@_renew_dots:
1000 {
1001     \cs_set_eq:NN \ldots \@@_Ldots
1002     \cs_set_eq:NN \cdots \@@_Cdots
1003     \cs_set_eq:NN \vdots \@@_Vdots
1004     \cs_set_eq:NN \ddots \@@_Ddots
1005     \cs_set_eq:NN \iddots \@@_Iddots
1006     \cs_set_eq:NN \dots \@@_Ldots
1007     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
1008 }
```

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

```

1009 \cs_new_protected:Npn \@@_colortbl_like:
1010 {
1011     \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
1012     \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
1013     \cs_set_eq:NN \columncolor \@@_columncolor_preamble
1014 }
```

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

```

1015 \cs_new_protected:Npn \@@_pre_array:
1016 {
```

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>41</sup>.

```

1017     \bool_if:NT \c_@@_booktabs_loaded_bool
1018     { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
1019     \box_clear_new:N \l_@@_cell_box
1020     \cs_if_exist:NT \theiRow
1021     { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
1022     \int_gzero_new:N \c@iRow
1023     \cs_if_exist:NT \thejCol
```

---

<sup>41</sup>cf. `\nicematrix@redefine@check@rerun`



```

1024     { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
1025     \int_gzero_new:N \c@jCol
1026     \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}`).

```

1027     \bool_if:NT \l_@@_small_bool
1028     {
1029         \cs_set_nopar:Npn \arraystretch { 0.47 }
1030         \dim_set:Nn \arraycolsep { 1.45 pt }
1031     }

```

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`.

```

1032     \cs_set_nopar:Npn \ialign
1033     {
1034         \bool_if:NTF \c_@@_colortbl_loaded_bool
1035         {
1036             \CT@everycr
1037             {
1038                 \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
1039                 \@@_everycr:
1040             }
1041         }
1042         { \everycr { \@@_everycr: } }
1043         \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>42</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`.

```

1044         \dim_gzero_new:N \g_@@_dp_row_zero_dim
1045         \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
1046         \dim_gzero_new:N \g_@@_ht_row_zero_dim
1047         \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
1048         \dim_gzero_new:N \g_@@_ht_row_one_dim
1049         \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
1050         \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
1051         \dim_gzero_new:N \g_@@_ht_last_row_dim
1052         \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
1053         \dim_gzero_new:N \g_@@_dp_last_row_dim
1054         \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`.

```

1055         \cs_set_eq:NN \ialign \@@_old_ialign:
1056         \halign
1057     }

```

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).

```

1058     \cs_set_eq:NN \@@_old_ldots \ldots
1059     \cs_set_eq:NN \@@_old_cdots \cdots
1060     \cs_set_eq:NN \@@_old_vdots \vdots
1061     \cs_set_eq:NN \@@_old_ddots \ddots

```

---

<sup>42</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

```

1062 \cs_set_eq:NN \@_old_iddots \iddots
1063 \bool_if:NTF \l_@@_standard_cline_bool
1064 { \cs_set_eq:NN \cline \@_standard_cline }
1065 { \cs_set_eq:NN \cline \@_cline }
1066 \cs_set_eq:NN \Ldots \@_Ldots
1067 \cs_set_eq:NN \Cdots \@_Cdots
1068 \cs_set_eq:NN \Vdots \@_Vdots
1069 \cs_set_eq:NN \Ddots \@_Ddots
1070 \cs_set_eq:NN \Iddots \@_Iddots
1071 \cs_set_eq:NN \hdottedline \@_hdottedline:
1072 \cs_set_eq:NN \Hline \@_Hline:
1073 \cs_set_eq:NN \Hspace \@_Hspace:
1074 \cs_set_eq:NN \Hdotsfor \@_Hdotsfor:
1075 \cs_set_eq:NN \Vdotsfor \@_Vdotsfor:
1076 \cs_set_eq:NN \multicolumn \@_multicolumn:nnn
1077 \cs_set_eq:NN \Block \@_Block:
1078 \cs_set_eq:NN \rotate \@_rotate:
1079 \cs_set_eq:NN \OnlyMainNiceMatrix \@_OnlyMainNiceMatrix:n
1080 \cs_set_eq:NN \dotfill \@_old_dotfill:
1081 \cs_set_eq:NN \CodeAfter \@_CodeAfter:
1082 \cs_set_eq:NN \diagbox \@_diagbox:nn
1083 \cs_set_eq:NN \NotEmpty \@_NotEmpty:
1084 \bool_if:NT \l_@@_colortbl_like_bool \@_colortbl_like:
1085 \bool_if:NT \l_@@_renew_dots_bool \@_renew_dots:

```

The sequence `\g_@@_multicolumn_cells_seq` will contain the list of the cells of the array where a command `\multicolumn{n}{...}{...}` with  $n > 1$  is issued. In `\g_@@_multicolumn_sizes_seq`, the “sizes” (that is to say the values of  $n$ ) correspondant will be stored. These lists will be used for the creation of the “medium nodes” (if they are created).

```

1086 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
1087 \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).

```

1088 \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`).

```

1089 \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.

```

1090 \int_gzero_new:N \g_@@_col_total_int
1091 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1092 \@_renew_NC@rewrite@S:
1093 \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.

```

1094 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1095 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1096 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1097 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1098 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1099 \tl_gclear_new:N \g_@@_HVdotsfor_lines_tl

1100 \tl_gclear_new:N \g_nicematrix_code_before_tl
1101 }

```

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

## The environment {NiceArrayWithDelims}

```

1102 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
1103 {
1104   \@@_provide_pgfsyspdfmark:
1105   \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.

```

1106   \bgroup
1107   \tl_set:Nn \l_@@_left_delim_tl { #1 }
1108   \tl_set:Nn \l_@@_right_delim_tl { #2 }
1109   \int_gzero:N \g_@@_block_box_int
1110   \dim_zero:N \g_@@_width_last_col_dim
1111   \dim_zero:N \g_@@_width_first_col_dim
1112   \bool_gset_false:N \g_@@_row_of_col_done_bool
1113   \str_if_empty:NT \g_@@_name_env_str
1114   { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1115   \@@_adapt_S_column:
1116   \bool_if:NTF \l_@@_NiceTabular_bool
1117     \mode_leave_vertical:
1118     \@@_test_if_math_mode:
1119   \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
1120   \bool_set_true:N \l_@@_in_env_bool

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>43</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.

```

1121   \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.

```

1122   \cs_if_exist:NT \tikz@library@external@loaded
1123   {
1124     \tikzexternaldisable
1125     \cs_if_exist:NT \ifstandalone
1126     { \tikzset { external / optimize = false } }
1127   }

```

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

```

1128   \int_gincr:N \g_@@_env_int
1129   \bool_if:NF \l_@@_block_auto_columns_width_bool
1130   { \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.

```

1131   \seq_gclear:N \g_@@_blocks_seq
1132   \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`.

```

1133   \seq_gclear:N \g_@@_pos_of_stroken_blocks_seq
1134   \seq_gclear:N \g_@@_pos_of_xdots_seq

1135   \tl_if_exist:cT { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1136   {
1137     \bool_set_true:N \l_@@_code_before_bool
1138     \exp_args:NNv \tl_put_right:Nn \l_@@_code_before_tl
1139     { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1140   }

```

---

<sup>43</sup>e.g. `\color[rgb]{0.5,0.5,0}`

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.

```

1141 \bool_if:NTF \l_@@_NiceArray_bool
1142 { \keys_set:nn { NiceMatrix / NiceArray } }
1143 { \keys_set:nn { NiceMatrix / pNiceArray } }
1144 { #3 , #5 }

1145 \tl_if_empty:NF \l_@@_rules_color_tl
1146 { \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.

```

1147 \bool_if:NT \l_@@_code_before_bool
1148 {
1149   \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1150   {

```

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).

```

1151 \int_zero_new:N \c@iRow
1152 \int_set:Nn \c@iRow
1153 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1154 \int_zero_new:N \c@jCol
1155 \int_set:Nn \c@jCol
1156 { \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.

```

1157 \int_compare:nNnF \l_@@_last_row_int = { -2 }
1158 { \int_decr:N \c@iRow }
1159 \int_compare:nNnF \l_@@_last_col_int = { -2 }
1160 { \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.

```

1161 \pgfsys@markposition { \@@_env: - position }
1162 \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1163 \pgfpicture

```

First, the creation of the row nodes.

```

1164 \int_step_inline:nnn
1165 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1166 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1167 {
1168   \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1169   \pgfcoordinate { \@@_env: - row - ##1 }
1170   { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1171 }

```

Now, the creation of the col nodes.

```

1172 \int_step_inline:nnn
1173 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1174 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1175 {
1176   \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1177   \pgfcoordinate { \@@_env: - col - ##1 }
1178   { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1179 }
1180 \endpgfpicture
1181 \group_begin:
1182 \bool_if:NT \c_@@_tikz_loaded_bool

```

```

1183     {
1184         \tikzset
1185         {
1186             every-picture / .style =
1187             { overlay , name~prefix = \@@_env: - }
1188         }
1189     }
1190     \cs_set_eq:NN \cellcolor \@@_cellcolor
1191     \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
1192     \cs_set_eq:NN \rowcolor \@@_rowcolor
1193     \cs_set_eq:NN \rowcolors \@@_rowcolors
1194     \cs_set_eq:NN \columncolor \@@_columncolor
1195     \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.

```

1196     \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1197     \l_@@_code_before_tl
1198     \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1199     \group_end:
1200 }
1201 }

```

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).

```

1202     \int_compare:nNnT \l_@@_last_row_int > { -2 }
1203     {
1204         \tl_put_right:Nn \@@_update_for_first_and_last_row:
1205         {
1206             \dim_gset:Nn \g_@@_ht_last_row_dim
1207             { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1208             \dim_gset:Nn \g_@@_dp_last_row_dim
1209             { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1210         }
1211     }
1212     \int_compare:nNnT \l_@@_last_row_int = { -1 }
1213     {
1214         \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.

```

1215     \str_if_empty:NTF \l_@@_name_str
1216     {
1217         \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1218         {
1219             \int_set:Nn \l_@@_last_row_int
1220             { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1221         }
1222     }
1223     {
1224         \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1225         {
1226             \int_set:Nn \l_@@_last_row_int
1227             { \use:c { @@_last_row_ \l_@@_name_str } }
1228         }
1229     }
1230 }

```

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).

```

1231     \int_compare:nNnT \l_@@_last_col_int = { -1 }
1232     {
1233         \str_if_empty:NTF \l_@@_name_str
1234         {
1235             \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }

```

```

1236         {
1237             \int_set:Nn \l_@@_last_col_int
1238             { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1239         }
1240     }
1241     {
1242         \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1243         {
1244             \int_set:Nn \l_@@_last_col_int
1245             { \use:c { @@_last_col_ \l_@@_name_str } }
1246         }
1247     }
1248 }

```

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

```

1249 \@@_pre_array:

```

We compute the width of the two delimiters.

```

1250 \dim_zero_new:N \l_@@_left_delim_dim
1251 \dim_zero_new:N \l_@@_right_delim_dim
1252 \bool_if:NTF \l_@@_NiceArray_bool
1253 {
1254     \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1255     \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1256 }
1257 {

```

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

```

1258 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
1259 \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1260 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
1261 \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1262 }

```

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

```

1263 \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).

```

1264 \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:

```

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

```

1265 \@@_construct_preamble:n { #4 }

```

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).

```

1266 \hbox_set:Nw \l_@@_the_array_box
1267 \skip_horizontal:N \l_@@_left_margin_dim
1268 \skip_horizontal:N \l_@@_extra_left_margin_dim
1269 \c_math_toggle_token
1270 \bool_if:NTF \l_@@_light_syntax_bool
1271 { \use:c { @@-light-syntax } }
1272 { \use:c { @@-normal-syntax } }
1273 }
1274 {
1275     \bool_if:NTF \l_@@_light_syntax_bool
1276     { \use:c { end @@-light-syntax } }

```

```

1277     { \use:c { end @@-normal-syntax } }
1278     \c_math_toggle_token
1279     \skip_horizontal:N \l_@@_right_margin_dim
1280     \skip_horizontal:N \l_@@_extra_right_margin_dim
1281     \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).

```

1282     \int_compare:nNnT \l_@@_last_row_int > { -2 }
1283     {
1284         \bool_if:NF \l_@@_last_row_without_value_bool
1285         {
1286             \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1287             {
1288                 \@@_error:n { Wrong~last~row }
1289                 \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1290             }
1291         }
1292     }

```

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>44</sup>

```

1293     \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1294     \bool_if:nTF \g_@@_last_col_found_bool
1295     { \int_gdecr:N \c@jCol }
1296     {
1297         \int_compare:nNnT \l_@@_last_col_int > { -1 }
1298         { \@@_error:n { last~col~not~used } }
1299     }

```

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

```

1300     \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1301     \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. 94).

```

1302     \int_compare:nNnT \l_@@_first_col_int = 0
1303     {
1304         \skip_horizontal:N \col@sep
1305         \skip_horizontal:N \g_@@_width_first_col_dim
1306     }

```

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.

```

1307     \bool_if:NTF \l_@@_NiceArray_bool
1308     {
1309         \str_case:VnF \l_@@_baseline_tl
1310         {
1311             b \@@_use_arraybox_with_notes_b:
1312             c \@@_use_arraybox_with_notes_c:
1313         }
1314         \@@_use_arraybox_with_notes:
1315     }

```

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).

```

1316     {
1317         \int_compare:nNnTF \l_@@_first_row_int = 0

```

<sup>44</sup>We remind that the potential “first column” (exterior) has the number 0.

```

1318     {
1319         \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1320         \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1321     }
1322     { \dim_zero:N \l_tmpa_dim }

```

We compute `\l_tmpb_dim` which is the total height of the “last row” below the array (when the key `last-row` is used). A value of `-2` for `\l_@@_last_row_int` means that there is no “last row”.<sup>45</sup>

```

1323     \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1324     {
1325         \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1326         \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1327     }
1328     { \dim_zero:N \l_tmpb_dim }
1329     \hbox_set:Nn \l_tmpa_box
1330     {
1331         \c_math_toggle_token
1332         \tl_if_empty:NF \l_@@_delimiters_color_tl
1333         { \color { \l_@@_delimiters_color_tl } }
1334         \left #1
1335         \vcenter
1336         {

```

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.

```

1337         \skip_vertical:N -\l_tmpa_dim
1338         \skip_vertical:N -\arrayrulewidth
1339         \hbox
1340         {
1341             \bool_if:NTF \l_@@_NiceTabular_bool
1342             { \skip_horizontal:N -\tabcolsep }
1343             { \skip_horizontal:N -\arraycolsep }
1344             \@@_use_arraybox_with_notes_c:
1345             \bool_if:NTF \l_@@_NiceTabular_bool
1346             { \skip_horizontal:N -\tabcolsep }
1347             { \skip_horizontal:N -\arraycolsep }
1348         }

```

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

```

1349         \skip_vertical:N -\l_tmpb_dim
1350         \skip_vertical:N \arrayrulewidth
1351     }
1352     \right #2
1353     \c_math_toggle_token
1354 }

```

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.

```

1355     \bool_if:NTF \l_@@_max_delimiter_width_bool
1356     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1357     \@@_put_box_in_flow:
1358 }

```

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. 95).

```

1359     \bool_if:NT \g_@@_last_col_found_bool
1360     {
1361         \skip_horizontal:N \g_@@_width_last_col_dim
1362         \skip_horizontal:N \col@sep
1363     }
1364     \bool_if:NF \l_@@_Matrix_bool
1365     {

```

---

<sup>45</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).



```

1366     \int_compare:nNnT \c@jCol < \g_@@_static_num_of_col_int
1367     { \@@_error:n { columns-not-used } }
1368   }
1369   \group_begin:
1370   \globaldefs = 1
1371   \@@_msg_redirect_name:nn { columns-not-used } { error }
1372   \group_end:
1373   \@@_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.

```

1374   \egroup
1375   \bool_if:NT \c_@@_footnote_bool \endsavenotes
1376 }

```

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 argument of `\@@_construct_preamble:n` is the preamble as given by the final user to the environment `{NiceTabular}` (or a variant). The preamble will be constructed in `\g_@@_preamble_tl`.

```

1377 \cs_new_protected:Npn \@@_construct_preamble:n #1
1378 {

```

First, we will do an “expansion” of the preamble with the tools of the package `array` itself. This “expansion” will expand all the constructions with `*` and with all column types (defined by the user or by various packages using `\newcolumntype`).

Since we use the tools of `array` to do this expansion, we will have a programming which is not in the style of `expl3`.

We redefine the column types `w` and `W`. We use `\@@_newcolumntype` instead of `\newcolumntype` because we don’t want warnings for column types already defined. These redefinitions are in fact *protections* of the letters `w` and `W`. We don’t want these columns type expanded because we will do the patch ourselves after. We want to be able the standard column types `w` and `W` in potential `{tabular}` of `array` in some cells of our array. That’s why we do those redefinitions in a TeX group.

```

1379   \group_begin:
1380   \bool_if:NTF \l_@@_Matrix_bool
1381   { \tl_gset:Nn \g_@@_preamble_tl { #1 } }
1382   {
1383     \@@_newcolumntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1384     \@@_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`).

```

1385     \@temptokena { #1 }

```

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

```

1386     \@tempswatruue

```

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

```

1387     \@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`.

```

1388     \int_gzero_new:N \c@jCol
1389     \bool_if:NTF \l_@@_vlines_bool
1390     {
1391         \tl_gset:Nn \g_@@_preamble_tl
1392         { ! { \skip_horizontal:N \arrayrulewidth } }
1393     }
1394     { \tl_gclear:N \g_@@_preamble_tl }

```

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

```

1395     \int_zero:N \l_tmpa_int

```

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

```

1396     \exp_after:wN \@@_patch_preamble:n \the \@temptokena \q_stop
1397     \int_gset_eq:NN \g_@@_static_num_of_col_int \c@jCol
1398 }

```

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

```

1399     \bool_if:NT \l_@@_colortbl_like_bool
1400     {
1401         \regex_replace_all:NnN
1402         \c_@@_columncolor_regex
1403         { \c { @@_columncolor_preamble } }
1404         \g_@@_preamble_tl
1405     }

```

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

```

1406     \int_compare:nNnTF \l_@@_first_col_int = 0
1407     { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1408     {
1409         \bool_lazy_all:nT
1410         {
1411             \l_@@_NiceArray_bool
1412             { \bool_not_p:n \l_@@_NiceTabular_bool }
1413             { \bool_not_p:n \l_@@_vlines_bool }
1414             { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1415         }
1416         { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1417     }
1418     \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1419     { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }
1420     {
1421         \bool_lazy_all:nT
1422         {
1423             \l_@@_NiceArray_bool
1424             { \bool_not_p:n \l_@@_NiceTabular_bool }
1425             { \bool_not_p:n \l_@@_vlines_bool }
1426             { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1427         }
1428         { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1429     }

```

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`).

```

1430     \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1431     {
1432         \tl_gput_right:Nn
1433         \g_@@_preamble_tl
1434         { > { \@@_error_too_much_cols: } 1 }
1435     }

```

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

```

1436     \group_end:
1437 }

```

```

1438 \cs_new_protected:Npn \@@_patch_preamble:n #1
1439 {
1440   \str_case:nnF { #1 }
1441   {
1442     c { \@@_patch_preamble_i:n #1 }
1443     l { \@@_patch_preamble_i:n #1 }
1444     r { \@@_patch_preamble_i:n #1 }
1445     > { \@@_patch_preamble_ii:nn #1 }
1446     ! { \@@_patch_preamble_ii:nn #1 }
1447     @ { \@@_patch_preamble_ii:nn #1 }
1448     | { \@@_patch_preamble_iii:n #1 }
1449     p { \@@_patch_preamble_iv:nnn t #1 }
1450     m { \@@_patch_preamble_iv:nnn c #1 }
1451     b { \@@_patch_preamble_iv:nnn b #1 }
1452     \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1453     \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1454     \@@_true_c: { \@@_patch_preamble_vi:n #1 }
1455     C { \@@_error:nn { old~column~type } C }
1456     L { \@@_error:nn { old~column~type } L }
1457     R { \@@_error:nn { old~column~type } R }
1458     \q_stop { }
1459   }
1460   {
1461     \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1462     { \@@_patch_preamble_vii:n #1 }
1463     { \@@_fatal:nn { unknown~column~type } { #1 } }
1464   }
1465 }

```

For c, l and r

```

1466 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1467 {
1468   \tl_gput_right:Nn \g_@@_preamble_tl
1469   {
1470     > { \@@_Cell: \tl_set:Nn \l_@@_cell_type_tl { #1 } }
1471     #1
1472     < \@@_end_Cell:
1473   }

```

We increment the counter of columns.

```

1474   \int_gincr:N \c@jCol
1475   \@@_patch_preamble_viii:n
1476 }

```

For >, ! and @

```

1477 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1478 {
1479   \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
1480   \@@_patch_preamble:n
1481 }

```

For |

```

1482 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1483 {

```

\l\_tmpa\_int is the number of successive occurrences of |

```

1484   \int_incr:N \l_tmpa_int
1485   \@@_patch_preamble_iii_i:n
1486 }

```

```

1487 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1488 {
1489   \str_if_eq:nnTF { #1 } |
1490   { \@@_patch_preamble_iii:n | }

```

```

1491 {
1492   \tl_gput_right:Nx \g_@@_preamble_tl
1493   {
1494     \exp_not:N !
1495     {
1496       \skip_horizontal:n
1497       {
1498         \dim_eval:n
1499         {
1500           \arrayrulewidth * \l_tmpa_int
1501           + \doublerulesep * ( \l_tmpa_int - 1)
1502         }
1503       }
1504     }
1505   }
1506   \tl_gput_right:Nx \g_@@_internal_code_after_tl
1507   { \@@_vline:nn { \@@_succ:n \c@jCol } { \int_use:N \l_tmpa_int } }
1508   \int_zero:N \l_tmpa_int
1509   \@@_patch_preamble:n #1
1510 }
1511 }

```

For p, m and b

```

1512 \cs_new_protected:Npn \@@_patch_preamble_iv:nnn #1 #2 #3
1513 {
1514   \tl_gput_right:Nn \g_@@_preamble_tl
1515   {
1516     > {
1517       \@@_Cell:
1518       \begin { minipage } [ #1 ] { #3 }
1519       \mode_leave_vertical:
1520       \arraybackslash % added in the version 5.8
1521       \box_use:N \@arstrutbox
1522     }
1523     c
1524     < { \box_use:N \@arstrutbox \end { minipage } \@@_end_Cell: }
1525   }

```

We increment the counter of columns.

```

1526   \int_gincr:N \c@jCol
1527   \@@_patch_preamble_viii:n
1528 }

```

For w and W

```

1529 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1530 {
1531   \tl_gput_right:Nn \g_@@_preamble_tl
1532   {
1533     > {
1534       \hbox_set:Nw \l_@@_cell_box
1535       \@@_Cell:
1536       \tl_set:Nn \l_@@_cell_type_tl { #3 }
1537     }
1538     c
1539     < {
1540       \@@_end_Cell:
1541       #1
1542       \hbox_set_end:
1543       \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1544       \@@_adjust_size_box:
1545       \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
1546     }
1547   }

```

We increment the counter of columns.

```

1548 \int_gincr:N \c@jCol
1549 \@@_patch_preamble_viii:n
1550 }

```

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

```

1551 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1552 {
1553 \tl_gput_right:Nn \g_@@_preamble_tl { c }

```

We increment the counter of columns.

```

1554 \int_gincr:N \c@jCol
1555 \@@_patch_preamble_viii:n
1556 }

1557 \cs_new_protected:Npn \@@_patch_preamble_vii:n #1
1558 {
1559 \tl_gput_right:Nn \g_@@_preamble_tl
1560 { ! { \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`.

```

1561 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1562 { \@@_vdottedline:n { \int_use:N \c@jCol } }
1563 \@@_patch_preamble:n
1564 }

```

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.

```

1565 \cs_new_protected:Npn \@@_patch_preamble_viii:n #1
1566 {
1567 \str_if_eq:nnTF { #1 } { < }
1568 \@@_patch_preamble_ix:n
1569 {
1570 \bool_if:NT \l_@@_vlines_bool
1571 {
1572 \tl_gput_right:Nn \g_@@_preamble_tl
1573 { ! { \skip_horizontal:N \arrayrulewidth } }
1574 }
1575 \@@_patch_preamble:n { #1 }
1576 }
1577 }

1578 \cs_new_protected:Npn \@@_patch_preamble_ix:n #1
1579 {
1580 \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
1581 \@@_patch_preamble_viii:n
1582 }

```

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`).

```

1583 \cs_new_protected:Npn \@@_put_box_in_flow:
1584 {
1585 \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1586 \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1587 \tl_if_eq:NnTF \l_@@_baseline_tl { c }
1588 { \box_use_drop:N \l_tmpa_box }
1589 \@@_put_box_in_flow_i:
1590 }

```

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).

```

1591 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1592 {
1593   \pgfpicture
1594     \@@_qpoint:n { row - 1 }
1595     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1596     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1597     \dim_gadd:Nn \g_tmpa_dim \pgf@y
1598     \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).

```

1599   \str_if_in:NnTF \l_@@_baseline_tl { line- }
1600   {
1601     \int_set:Nn \l_tmpa_int
1602     {
1603       \str_range:Nnn
1604         \l_@@_baseline_tl
1605         6
1606         { \tl_count:V \l_@@_baseline_tl }
1607     }
1608     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1609   }
1610   {
1611     \str_case:VnF \l_@@_baseline_tl
1612     {
1613       { t } { \int_set:Nn \l_tmpa_int 1 }
1614       { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1615     }
1616     { \int_set:Nn \l_tmpa_int \l_@@_baseline_tl }
1617     \bool_lazy_or:nnT
1618     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1619     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1620     {
1621       \@@_error:n { bad~value~for~baseline }
1622       \int_set:Nn \l_tmpa_int 1
1623     }
1624     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

We take into account the position of the mathematical axis.

```

1625     \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
1626   }
1627   \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

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

```

1628   \endpgfpicture
1629   \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1630   \box_use_drop:N \l_tmpa_box
1631 }

```

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).

```

1632 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
1633 {

```

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

```

1634   \begin { minipage } [ t ] { \box_wd:N \l_@@_the_array_box }
1635   \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.

```

1636 \@@_create_extra_nodes:
1637 \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
1638 \bool_lazy_or:nnT
1639 { \int_compare_p:nNn \c@tabularnote > 0 }
1640 { ! \tl_if_empty_p:V \l_@@_tabularnote_tl }
1641 \@@_insert_tabularnotes:
1642 \end { minipage }
1643 }
1644 \cs_new_protected:Npn \@@_insert_tabularnotes:
1645 {
1646 \skip_vertical:N 0.65ex

```

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

```

1647 \group_begin:
1648 \l_@@_notes_code_before_tl
1649 \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.

```

1650 \int_compare:nNnT \c@tabularnote > 0
1651 {
1652 \bool_if:NTF \l_@@_notes_para_bool
1653 {
1654 \begin { tabularnotes* }
1655 \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1656 \end { tabularnotes* }

```

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

```

1657 \par
1658 }
1659 {
1660 \tabularnotes
1661 \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1662 \endtabularnotes
1663 }
1664 }
1665 \unskip
1666 \group_end:
1667 \bool_if:NT \l_@@_notes_bottomrule_bool
1668 {
1669 \bool_if:NTF \c_@@_booktabs_loaded_bool
1670 {

```

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

```

1671 \skip_vertical:N \aboverulesep
\CT@arc@ is the specification of color defined by colortbl but you use it even if colortbl is not loaded.
1672 { \CT@arc@ \hrule height \heavyrulewidth }
1673 }
1674 { \@@_error:n { bottomule~without~booktabs } }
1675 }
1676 \l_@@_notes_code_after_tl
1677 \seq_gclear:N \g_@@_tabularnotes_seq
1678 \int_gzero:N \c@tabularnote
1679 }

```

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`.

```

1680 \cs_new_protected:Npn \@@_use_arraybox_with_notes_b:
1681 {
1682   \pgfpicture
1683     \@@_qpoint:n { row - 1 }
1684     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1685     \@@_qpoint:n { row - \int_use:N \c@iRow - base }
1686     \dim_gsub:Nn \g_tmpa_dim \pgf@y
1687   \endpgfpicture
1688   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1689   \int_compare:nNnT \l_@@_first_row_int = 0
1690   {
1691     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1692     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1693   }
1694   \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
1695 }

```

Now, the general case.

```

1696 \cs_new_protected:Npn \@@_use_arraybox_with_notes:
1697 {

```

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

```

1698   \tl_if_eq:NnT \l_@@_baseline_tl { t }
1699   { \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`.

```

1700   \pgfpicture
1701   \@@_qpoint:n { row - 1 }
1702   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1703   \str_if_in:NnTF \l_@@_baseline_tl { line- }
1704   {
1705     \int_set:Nn \l_tmpa_int
1706     {
1707       \str_range:Nnn
1708         \l_@@_baseline_tl
1709         6
1710         { \tl_count:V \l_@@_baseline_tl }
1711     }
1712     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1713   }
1714   {
1715     \int_set:Nn \l_tmpa_int \l_@@_baseline_tl
1716     \bool_lazy_or:nnT
1717       { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1718       { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1719     {
1720       \@@_error:n { bad-value-for-baseline }
1721       \int_set:Nn \l_tmpa_int 1
1722     }
1723     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1724   }
1725   \dim_gsub:Nn \g_tmpa_dim \pgf@y
1726   \endpgfpicture
1727   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1728   \int_compare:nNnT \l_@@_first_row_int = 0
1729   {
1730     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1731     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1732   }
1733   \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
1734 }

```



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.

```
1735 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1736 {
```

We will compute the real width of both delimiters used.

```
1737 \dim_zero_new:N \l_@@_real_left_delim_dim
1738 \dim_zero_new:N \l_@@_real_right_delim_dim
1739 \hbox_set:Nn \l_tmpb_box
1740 {
1741   \c_math_toggle_token
1742   \left #1
1743   \vcenter
1744   {
1745     \vbox_to_ht:nn
1746     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1747     { }
1748   }
1749   \right .
1750   \c_math_toggle_token
1751 }
1752 \dim_set:Nn \l_@@_real_left_delim_dim
1753 { \box_wd:N \l_tmpb_box - \nullldelimiterspace }
1754 \hbox_set:Nn \l_tmpb_box
1755 {
1756   \c_math_toggle_token
1757   \left .
1758   \vbox_to_ht:nn
1759   { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1760   { }
1761   \right #2
1762   \c_math_toggle_token
1763 }
1764 \dim_set:Nn \l_@@_real_right_delim_dim
1765 { \box_wd:N \l_tmpb_box - \nullldelimiterspace }
```

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

```
1766 \skip_horizontal:N \l_@@_left_delim_dim
1767 \skip_horizontal:N -\l_@@_real_left_delim_dim
1768 \@@_put_box_in_flow:
1769 \skip_horizontal:N \l_@@_right_delim_dim
1770 \skip_horizontal:N -\l_@@_real_right_delim_dim
1771 }
```

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).

```
1772 \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).

```
1773 {
1774   \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`).

```
1775 { \exp_args:NV \@@_array: \g_@@_preamble_tl }
1776 }
1777 {
```

```

1778 \@@_create_col_nodes:
1779 \endarray
1780 }

```

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`).

```

1781 \NewDocumentEnvironment { @@-light-syntax } { b }
1782 {

```

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`.

```

1783 \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1784 \tl_map_inline:nn { #1 }
1785 {
1786   \tl_if_eq:nnT { ##1 } { & }
1787   { \@@_fatal:n { ampersand-in~light-syntax } }
1788   \tl_if_eq:nnT { ##1 } { \ }
1789   { \@@_fatal:n { double-backslash-in~light-syntax } }
1790 }

```

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`.

```

1791 \@@_light_syntax_i #1 \CodeAfter \q_stop
1792 }

```

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`.

```

1793 { }
1794 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1795 {
1796   \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).

```

1797 \seq_gclear_new:N \g_@@_rows_seq
1798 \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1799 \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.

```

1800 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1801 { \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`).

```

1802 \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.

```

1803 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1804 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1805 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1806 \@@_create_col_nodes:
1807 \endarray
1808 }
1809 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1810 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }

```

```

1811 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1812 {
1813   \seq_gclear_new:N \g_@@_cells_seq
1814   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1815   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1816   \l_tmpa_tl
1817   \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1818 }

```

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).

```

1819 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1820 {
1821   \str_if_eq:VnT \g_@@_name_env_str { #2 }
1822   { \@@_fatal:n { empty-environment } }

```

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

```

1823   \end { #2 }
1824 }

```

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 specifies the width of the columns).

```

1825 \cs_new:Npn \@@_create_col_nodes:
1826 {
1827   \crrcr
1828   \int_compare:nNnT \l_@@_first_col_int = 0
1829   {
1830     \omit
1831     \hbox_overlap_left:n
1832     {
1833       \bool_if:NT \l_@@_code_before_bool
1834       { \pgfsys@markposition { \@@_env: - col - 0 } }
1835       \pgfpicture
1836       \pgfrememberpicturepositiononpagetrue
1837       \pgfcoordinate { \@@_env: - col - 0 } \pgfpintorigin
1838       \str_if_empty:NF \l_@@_name_str
1839       { \pgfnodelalias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1840       \endpgfpicture
1841       \skip_horizontal:N 2\col@sep
1842       \skip_horizontal:N \g_@@_width_first_col_dim
1843     }
1844     &
1845   }
1846   \omit

```

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

```

1847   \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`).

```

1848   \int_compare:nNnTF \l_@@_first_col_int = 0
1849   {
1850     \bool_if:NT \l_@@_code_before_bool
1851     {
1852       \hbox
1853       {
1854         \skip_horizontal:N -0.5\arrayrulewidth
1855         \pgfsys@markposition { \@@_env: - col - 1 }
1856         \skip_horizontal:N 0.5\arrayrulewidth
1857       }
1858     }

```

```

1859 \pgfpicture
1860 \pgfrememberpicturerepositiononpagetrue
1861 \pgfcoordinate { \@@_env: - col - 1 }
1862 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1863 \str_if_empty:NF \l_@@_name_str
1864 { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1865 \endpgfpicture
1866 }
1867 {
1868 \bool_if:NT \l_@@_code_before_bool
1869 {
1870 \hbox
1871 {
1872 \skip_horizontal:N 0.5\arrayrulewidth
1873 \pgfsys@markposition { \@@_env: - col - 1 }
1874 \skip_horizontal:N -0.5\arrayrulewidth
1875 }
1876 }
1877 \pgfpicture
1878 \pgfrememberpicturerepositiononpagetrue
1879 \pgfcoordinate { \@@_env: - col - 1 }
1880 { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
1881 \str_if_empty:NF \l_@@_name_str
1882 { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1883 \endpgfpicture
1884 }

```

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.

```

1885 \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1886 \bool_if:NF \l_@@_auto_columns_width_bool
1887 { \dim_compare:nNt \l_@@_columns_width_dim > \c_zero_dim }
1888 {
1889 \bool_lazy_and:nnTF
1890 \l_@@_auto_columns_width_bool
1891 { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1892 { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1893 { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1894 \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
1895 }
1896 \skip_horizontal:N \g_tmpa_skip
1897 \hbox
1898 {
1899 \bool_if:NT \l_@@_code_before_bool
1900 {
1901 \hbox
1902 {
1903 \skip_horizontal:N -0.5\arrayrulewidth
1904 \pgfsys@markposition { \@@_env: - col - 2 }
1905 \skip_horizontal:N 0.5\arrayrulewidth
1906 }
1907 }
1908 \pgfpicture
1909 \pgfrememberpicturerepositiononpagetrue
1910 \pgfcoordinate { \@@_env: - col - 2 }
1911 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1912 \str_if_empty:NF \l_@@_name_str
1913 { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
1914 \endpgfpicture

```

```
1915 }
```

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.

```
1916 \int_gset:Nn \g_tmpa_int 1
1917 \bool_if:NTF \g_@@_last_col_found_bool
1918 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1919 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1920 {
1921   &
1922   \omit
1923   \int_gincr:N \g_tmpa_int
```

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

```
1924   \skip_horizontal:N \g_tmpa_skip
1925   \bool_if:NT \l_@@_code_before_bool
1926   {
1927     \hbox
1928     {
1929       \skip_horizontal:N -0.5\arrayrulewidth
1930       \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1931       \skip_horizontal:N 0.5\arrayrulewidth
1932     }
1933   }
```

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

```
1934   \pgfpicture
1935   \pgfrememberpicturepositiononpagetrue
1936   \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1937   { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1938   \str_if_empty:NF \l_@@_name_str
1939   {
1940     \pgfnodealias
1941     { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
1942     { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1943   }
1944   \endpgfpicture
1945 }
1946 \bool_if:NT \g_@@_last_col_found_bool
1947 {
1948   \hbox_overlap_right:n
1949   {
1950     % \skip_horizontal:N \colsep
1951     \skip_horizontal:N \g_@@_width_last_col_dim
1952     \bool_if:NT \l_@@_code_before_bool
1953     {
1954       \pgfsys@markposition
1955       { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1956     }
1957     \pgfpicture
1958     \pgfrememberpicturepositiononpagetrue
1959     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1960     \pgfpointorigin
1961     \str_if_empty:NF \l_@@_name_str
1962     {
1963       \pgfnodealias
1964       { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
1965       { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1966     }
1967     \endpgfpicture
1968   }
1969 }
1970 \cr
1971 }
```

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

```

1972 \tl_const:Nn \c_@@_preamble_first_col_tl
1973 {
1974   >
1975   {
1976     \bool_gset_true:N \g_@@_after_col_zero_bool
1977     \@@_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.

```

1978     \hbox_set:Nw \l_@@_cell_box
1979     \@@_math_toggle_token:
1980     \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”.

```

1981     \bool_lazy_and:nnT
1982     { \int_compare_p:nNn \c@iRow > 0 }
1983     {
1984       \bool_lazy_or_p:nn
1985       { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1986       { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1987     }
1988     {
1989       \l_@@_code_for_first_col_tl
1990       \xglobal \colorlet { nicematrix-first-col } { . }
1991     }
1992   }

```

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`.

```

1993   l
1994   <
1995   {
1996     \@@_math_toggle_token:
1997     \hbox_set_end:
1998     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1999     \@@_adjust_size_box:
2000     \@@_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.

```

2001     \dim_gset:Nn \g_@@_width_first_col_dim
2002     { \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.

```

2003     \hbox_overlap_left:n
2004     {
2005       \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2006       \@@_node_for_the_cell:
2007       { \box_use_drop:N \l_@@_cell_box }
2008       \skip_horizontal:N \l_@@_left_delim_dim
2009       \skip_horizontal:N \l_@@_left_margin_dim
2010       \skip_horizontal:N \l_@@_extra_left_margin_dim
2011     }
2012     \bool_gset_false:N \g_@@_empty_cell_bool
2013     \skip_horizontal:N -2\col@sep
2014   }
2015 }

```

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

```

2016 \tl_const:Nn \c_@@_preamble_last_col_tl
2017 {
2018   >
2019   {

```

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

```
2020 \bool_gset_true:N \g_@@_last_col_found_bool
2021 \int_gincr:N \c@jCol
2022 \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.

```
2023 \hbox_set:Nw \l_@@_cell_box
2024 \@@_math_toggle_token:
2025 \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”.

```
2026 \int_compare:nNnT \c@iRow > 0
2027 {
2028   \bool_lazy_or:nnT
2029   { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2030   { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2031   {
2032     \l_@@_code_for_last_col_tl
2033     \xglobal \colorlet { nicematrix-last-col } { . }
2034   }
2035 }
2036 }
2037 1
2038 <
2039 {
2040   \@@_math_toggle_token:
2041   \hbox_set_end:
2042   \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2043   \@@_adjust_size_box:
2044   \@@_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.

```
2045 \dim_gset:Nn \g_@@_width_last_col_dim
2046 { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
2047 \skip_horizontal:N -2\col@sep
```

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

```
2048 \hbox_overlap_right:n
2049 {
2050   \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2051   {
2052     \skip_horizontal:N \l_@@_right_delim_dim
2053     \skip_horizontal:N \l_@@_right_margin_dim
2054     \skip_horizontal:N \l_@@_extra_right_margin_dim
2055     \@@_node_for_the_cell:
2056   }
2057 }
2058 \bool_gset_false:N \g_@@_empty_cell_bool
2059 }
2060 }
```

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.

```
2061 \NewDocumentEnvironment { NiceArray } { }
2062 {
2063   \bool_set_true:N \l_@@_NiceArray_bool
2064   \str_if_empty:NT \g_@@_name_env_str
2065   { \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).

```

2066 \NiceArrayWithDelims . .
2067 }
2068 { \endNiceArrayWithDelims }

```

We create the variants of the environment {NiceArrayWithDelims}.

```

2069 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
2070 {
2071   \NewDocumentEnvironment { #1 NiceArray } { }
2072   {
2073     \str_if_empty:NT \g_@@_name_env_str
2074     { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
2075     \@@_test_if_math_mode:
2076     \NiceArrayWithDelims #2 #3
2077   }
2078   { \endNiceArrayWithDelims }
2079 }
2080 \@@_def_env:nnn p ( )
2081 \@@_def_env:nnn b [ ]
2082 \@@_def_env:nnn B \{ \}
2083 \@@_def_env:nnn v | |
2084 \@@_def_env:nnn V \| \|

```

## The environment {NiceMatrix} and its variants

```

2085 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
2086 {
2087   \bool_set_true:N \l_@@_Matrix_bool
2088   \use:c { #1 NiceArray }
2089   {
2090     *
2091     {
2092       \int_compare:nNnTF \l_@@_last_col_int < 0
2093       \c@MaxMatrixCols
2094       { \@@_pred:n \l_@@_last_col_int }
2095     }
2096     { > \@@_Cell: #2 < \@@_end_Cell: }
2097   }
2098 }
2099 \clist_map_inline:nn { { } , p , b , B , v , V }
2100 {
2101   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
2102   {
2103     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
2104     \tl_set:Nn \l_@@_type_of_col_tl c
2105     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
2106     \exp_args:Nne \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
2107   }
2108   { \use:c { end #1 NiceArray } }
2109 }

```

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

```

2110 \cs_new_protected:Npn \@@_NotEmpty:
2111 { \bool_gset_true:N \g_@@_not_empty_cell_bool }

```

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



```

2112 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
2113 {
2114   \str_gset:Nn \g_@@_name_env_str { NiceTabular }
2115   \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
2116   \bool_set_true:N \l_@@_NiceTabular_bool
2117   \NiceArray { #2 }
2118 }
2119 { \endNiceArray }

2120 \NewDocumentEnvironment { NiceTabular* } { m 0 { } m ! 0 { } }
2121 {
2122   \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
2123   \dim_set:Nn \l_@@_tabular_width_dim { #1 }
2124   \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2125   \bool_set_true:N \l_@@_NiceTabular_bool
2126   \NiceArray { #3 }
2127 }
2128 { \endNiceArray }

```

## After the construction of the array

```

2129 \cs_new_protected:Npn \@@_after_array:
2130 {
2131   \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.

```

2132   \bool_if:NT \g_@@_last_col_found_bool
2133   { \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`.

```

2134   \bool_if:NT \l_@@_last_col_without_value_bool
2135   {
2136     \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
2137     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2138     \iow_shipout:Nx \@mainaux
2139     {
2140       \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
2141       { \int_use:N \g_@@_col_total_int }
2142     }
2143     \str_if_empty:NF \l_@@_name_str
2144     {
2145       \iow_shipout:Nx \@mainaux
2146       {
2147         \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
2148         { \int_use:N \g_@@_col_total_int }
2149       }
2150     }
2151     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2152   }

```

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.

```

2153   \bool_if:NT \l_@@_last_row_without_value_bool
2154   {
2155     \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.

```

2156 \bool_if:NF \l_@@_light_syntax_bool
2157 {
2158   \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2159   \iow_shipout:Nx \@mainaux
2160   {
2161     \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
2162     { \int_use:N \g_@@_row_total_int }
2163   }

```

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.

```

2164 \str_if_empty:NF \l_@@_name_str
2165 {
2166   \iow_shipout:Nx \@mainaux
2167   {
2168     \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
2169     { \int_use:N \g_@@_row_total_int }
2170   }
2171 }
2172 \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2173 }
2174 }

```

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

```

2175 \bool_if:NT \l_@@_code_before_bool
2176 {
2177   \iow_now:Nn \@mainaux \ExplSyntaxOn
2178   \iow_now:Nx \@mainaux
2179   { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
2180   \iow_now:Nx \@mainaux
2181   {
2182     \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
2183     {
2184       \int_use:N \l_@@_first_row_int ,
2185       \int_use:N \g_@@_row_total_int ,
2186       \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`.

```

2187 \bool_lazy_and:nnTF
2188 { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
2189 { \bool_not_p:n \g_@@_last_col_found_bool }
2190 \@@_succ:n
2191 \int_use:N
2192 \g_@@_col_total_int
2193 }

```

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`).

```

2194 \seq_gset_from_clist:cn
2195 { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
2196 { \seq_use:Nnnn \g_@@_pos_of_blocks_seq , , , }
2197 }
2198 \iow_now:Nn \@mainaux \ExplSyntaxOff
2199 }

```

By default, the diagonal lines will be parallelized<sup>46</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.

```

2200 \bool_if:NT \l_@@_parallelize_diags_bool
2201 {
2202   \int_gzero_new:N \g_@@_ddots_int
2203   \int_gzero_new:N \g_@@_iddots_int

```

<sup>46</sup>It's possible to use the option `parallelize-diags` to disable this parallelization.

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.

```

2204     \dim_gzero_new:N \g_@@_delta_x_one_dim
2205     \dim_gzero_new:N \g_@@_delta_y_one_dim
2206     \dim_gzero_new:N \g_@@_delta_x_two_dim
2207     \dim_gzero_new:N \g_@@_delta_y_two_dim
2208   }
2209   \int_zero_new:N \l_@@_initial_i_int
2210   \int_zero_new:N \l_@@_initial_j_int
2211   \int_zero_new:N \l_@@_final_i_int
2212   \int_zero_new:N \l_@@_final_j_int
2213   \bool_set_false:N \l_@@_initial_open_bool
2214   \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.

```

2215     \bool_if:NT \l_@@_small_bool
2216     {
2217       \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2218       \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.

```

2219       \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2220     }

```

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

```

2221     \@@_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.

```

2222     \@@_compute_corners:

```

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

```

2223     \@@_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`.

```

2224     \bool_lazy_all:nT
2225     {
2226       { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
2227       { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
2228       { \seq_if_empty_p:N \l_@@_empty_corner_cells_seq }
2229     }
2230     {
2231       \cs_set_eq:NN \@@_vline_i:nn \@@_vline_i_complete:nn
2232       \cs_set_eq:NN \@@_hline_i:nn \@@_hline_i_complete:nn
2233     }
2234     \bool_if:NT \l_@@_hlines_bool \@@_draw_hlines:
2235     \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
2236     \g_@@_internal_code_after_tl
2237     \tl_gclear:N \g_@@_internal_code_after_tl

```

Now, the code-after.

```

2238 \bool_if:NT \c_@@_tikz_loaded_bool
2239 {
2240     \tikzset
2241     {
2242         every~picture / .style =
2243         {
2244             overlay ,
2245             remember~picture ,
2246             name~prefix = \@@_env: -
2247         }
2248     }
2249 }
2250 \cs_set_eq:NN \line \@@_line

```

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.

```

2251 \cs_set_eq:NN \CodeAfter \prg_do_nothing:

```

And here's the code-after:

```

2252 \g_nicematrix_code_after_tl
2253 \tl_gclear:N \g_nicematrix_code_after_tl
2254 \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.

```

2255 \tl_if_empty:NF \g_nicematrix_code_before_tl
2256 {

```

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`.

```

2257 \cs_set_protected:Npn \rectanglecolor { }
2258 \cs_set_protected:Npn \columncolor { }
2259 \iow_now:Nn \@mainaux \ExplSyntaxOn
2260 \iow_now:Nx \@mainaux
2261 {
2262     \tl_gset:cn
2263     { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
2264     { \exp_not:V \g_nicematrix_code_before_tl }
2265 }
2266 \iow_now:Nn \@mainaux \ExplSyntaxOff
2267 \bool_set_true:N \l_@@_code_before_bool
2268 }

```

```

2269 \str_gclear:N \g_@@_name_env_str
2270 \@@_restore_iRow_jCol:

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>47</sup>. This command is used by `\CT@arc@` but we use it also for compatibility with `colortbl`. But we want also to be able to use color for the rules of the array when `colortbl` is *not* loaded. That's why we do the following instruction which is in the patch of the end of arrays done by `colortbl`.

```

2271 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
2272 }

```

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

<sup>47</sup>e.g. `\color[rgb]{0.5,0.5,0}`

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).

```

2273 \cs_new_protected:Npn \@@_adjust_pos_of_blocks_seq:
2274 {
2275     \seq_gset_map_x:NNn \g_@@_pos_of_blocks_seq \g_@@_pos_of_blocks_seq
2276     { \@@_adjust_pos_of_blocks_seq_i:nnnn #1 }
2277 }

```

The following command must *not* be protected.

```

2278 \cs_new:Npn \@@_adjust_pos_of_blocks_seq_i:nnnn #1 #2 #3 #4
2279 {
2280     { #1 }
2281     { #2 }
2282     {
2283         \int_compare:nNnTF { #3 } > { 99 }
2284         { \int_use:N \c@iRow }
2285         { #3 }
2286     }
2287     {
2288         \int_compare:nNnTF { #4 } > { 99 }
2289         { \int_use:N \c@jCol }
2290         { #4 }
2291     }
2292 }

```

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).

```

2293 \AtBeginDocument
2294 {
2295     \cs_new_protected:Npx \@@_draw_dotted_lines:
2296     {
2297         \c_@@_pgfortikzpicture_tl
2298         \@@_draw_dotted_lines_i:
2299         \c_@@_endpgfortikzpicture_tl
2300     }
2301 }

```

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

```

2302 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2303 {
2304     \pgfrememberpicturerepositiononpagetrue
2305     \pgf@relevantforpicturesizefalse
2306     \g_@@_HVdotsfor_lines_tl
2307     \g_@@_Vdots_lines_tl
2308     \g_@@_Ddots_lines_tl
2309     \g_@@_Iddots_lines_tl
2310     \g_@@_Cdots_lines_tl
2311     \g_@@_Ldots_lines_tl
2312 }

2313 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2314 {
2315     \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2316     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2317 }

```

## 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 `\l_@@_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.

```
2318 \cs_new_protected:Npn \l_@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2319 {
```

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

```
2320 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
2321 \int_set:Nn \l_@@_initial_i_int { #1 }
2322 \int_set:Nn \l_@@_initial_j_int { #2 }
2323 \int_set:Nn \l_@@_final_i_int { #1 }
2324 \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.

```
2325 \bool_set_false:N \l_@@_stop_loop_bool
2326 \bool_do_until:Nn \l_@@_stop_loop_bool
2327 {
2328   \int_add:Nn \l_@@_final_i_int { #3 }
2329   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
2330 \bool_set_false:N \l_@@_final_open_bool
2331 \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
2332 {
2333   \int_compare:nNnTF { #3 } = 1
2334   { \bool_set_true:N \l_@@_final_open_bool }
2335   {
2336     \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2337     { \bool_set_true:N \l_@@_final_open_bool }
2338   }
2339 }
2340 {
2341   \int_compare:nNnTF \l_@@_final_j_int < 1
2342   {
2343     \int_compare:nNnT { #4 } = { -1 }
2344     { \bool_set_true:N \l_@@_final_open_bool }
2345   }
2346 }
```

```

2347         \int_compare:nNtT \l_@@_final_j_int > \c@jCol
2348         {
2349             \int_compare:nNtT { #4 } = 1
2350             { \bool_set_true:N \l_@@_final_open_bool }
2351         }
2352     }
2353 }
2354 \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.

```

2355     {

```

We do a step backwards.

```

2356         \int_sub:Nn \l_@@_final_i_int { #3 }
2357         \int_sub:Nn \l_@@_final_j_int { #4 }
2358         \bool_set_true:N \l_@@_stop_loop_bool
2359     }

```

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`.

```

2360     {
2361         \cs_if_exist:cTF
2362         {
2363             @@ _ dotted _
2364             \int_use:N \l_@@_final_i_int -
2365             \int_use:N \l_@@_final_j_int
2366         }
2367         {
2368             \int_sub:Nn \l_@@_final_i_int { #3 }
2369             \int_sub:Nn \l_@@_final_j_int { #4 }
2370             \bool_set_true:N \l_@@_final_open_bool
2371             \bool_set_true:N \l_@@_stop_loop_bool
2372         }
2373         {
2374             \cs_if_exist:cTF
2375             {
2376                 pgf @ sh @ ns @ \@@_env:
2377                 - \int_use:N \l_@@_final_i_int
2378                 - \int_use:N \l_@@_final_j_int
2379             }
2380             { \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.

```

2381         {
2382             \cs_set:cpn
2383             {
2384                 @@ _ dotted _
2385                 \int_use:N \l_@@_final_i_int -
2386                 \int_use:N \l_@@_final_j_int
2387             }
2388             { }
2389         }
2390     }
2391 }
2392 }

```

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

```

2393 \bool_set_false:N \l_@@_stop_loop_bool
2394 \bool_do_until:Nn \l_@@_stop_loop_bool
2395 {
2396   \int_sub:Nn \l_@@_initial_i_int { #3 }
2397   \int_sub:Nn \l_@@_initial_j_int { #4 }
2398   \bool_set_false:N \l_@@_initial_open_bool
2399   \int_compare:nNnTF \l_@@_initial_i_int < 1
2400   {
2401     \int_compare:nNnTF { #3 } = 1
2402     { \bool_set_true:N \l_@@_initial_open_bool }
2403     {
2404       \int_compare:nNnT \l_@@_initial_j_int = 0
2405       { \bool_set_true:N \l_@@_initial_open_bool }
2406     }
2407   }
2408   {
2409     \int_compare:nNnTF \l_@@_initial_j_int < 1
2410     {
2411       \int_compare:nNnT { #4 } = 1
2412       { \bool_set_true:N \l_@@_initial_open_bool }
2413     }
2414     {
2415       \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
2416       {
2417         \int_compare:nNnT { #4 } = { -1 }
2418         { \bool_set_true:N \l_@@_initial_open_bool }
2419       }
2420     }
2421   }
2422   \bool_if:NTF \l_@@_initial_open_bool
2423   {
2424     \int_add:Nn \l_@@_initial_i_int { #3 }
2425     \int_add:Nn \l_@@_initial_j_int { #4 }
2426     \bool_set_true:N \l_@@_stop_loop_bool
2427   }
2428   {
2429     \cs_if_exist:cTF
2430     {
2431       @@ _ dotted _
2432       \int_use:N \l_@@_initial_i_int -
2433       \int_use:N \l_@@_initial_j_int
2434     }
2435     {
2436       \int_add:Nn \l_@@_initial_i_int { #3 }
2437       \int_add:Nn \l_@@_initial_j_int { #4 }
2438       \bool_set_true:N \l_@@_initial_open_bool
2439       \bool_set_true:N \l_@@_stop_loop_bool
2440     }
2441   }
2442   \cs_if_exist:cTF
2443   {
2444     pgf @ sh @ ns @ \@@_env:
2445     - \int_use:N \l_@@_initial_i_int
2446     - \int_use:N \l_@@_initial_j_int
2447   }
2448   { \bool_set_true:N \l_@@_stop_loop_bool }
2449   {
2450     \cs_set:cpn
2451     {
2452       @@ _ dotted _
2453       \int_use:N \l_@@_initial_i_int -
2454       \int_use:N \l_@@_initial_j_int
2455     }

```



```

2456         { }
2457     }
2458 }
2459 }
2460 }

```

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.

```

2461 \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2462 {
2463     { \int_use:N \l_@@_initial_i_int }
2464     { \int_use:N \l_@@_initial_j_int }
2465     { \int_use:N \l_@@_final_i_int }
2466     { \int_use:N \l_@@_final_j_int }
2467 }
2468 }

2469 \cs_new_protected:Npn \@@_set_initial_coords:
2470 {
2471     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2472     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2473 }
2474 \cs_new_protected:Npn \@@_set_final_coords:
2475 {
2476     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2477     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2478 }
2479 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2480 {
2481     \pgfpointanchor
2482     {
2483         \@@_env:
2484         - \int_use:N \l_@@_initial_i_int
2485         - \int_use:N \l_@@_initial_j_int
2486     }
2487     { #1 }
2488     \@@_set_initial_coords:
2489 }
2490 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2491 {
2492     \pgfpointanchor
2493     {
2494         \@@_env:
2495         - \int_use:N \l_@@_final_i_int
2496         - \int_use:N \l_@@_final_j_int
2497     }
2498     { #1 }
2499     \@@_set_final_coords:
2500 }

```

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.

```

2501 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
2502 {
2503     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2504     {
2505         \@@_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.

```

2506     \group_begin:
2507     \int_compare:nNnTF { #1 } = 0
2508     { \color { nicematrix-first-row } }
2509     {

```

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.

```

2510         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2511         { \color { nicematrix-last-row } }
2512     }
2513     \keys_set:nn { NiceMatrix / xdots } { #3 }
2514     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2515     \@@_actually_draw_Ldots:
2516 \group_end:
2517 }
2518 }

```

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`.

```

2519 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2520 {
2521     \bool_if:NTF \l_@@_initial_open_bool
2522     {
2523         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2524         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2525         \dim_add:Nn \l_@@_x_initial_dim \col@sep
2526         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2527         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2528     }
2529     { \@@_set_initial_coords_from_anchor:n { base-east } }
2530     \bool_if:NTF \l_@@_final_open_bool
2531     {
2532         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2533         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2534         \dim_sub:Nn \l_@@_x_final_dim \col@sep
2535         \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2536         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2537     }
2538     { \@@_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 (?).

```

2539     \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2540     \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2541     \@@_draw_line:
2542 }

```

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.

```

2543 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2544 {
2545     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2546     {
2547         \@@_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.

```

2548     \group_begin:
2549     \int_compare:nNnTF { #1 } = 0
2550     { \color { nicematrix-first-row } }
2551     {

```

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.

```

2552         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2553         { \color { nicematrix-last-row } }
2554     }
2555     \keys_set:nn { NiceMatrix / xdots } { #3 }
2556     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2557     \@@_actually_draw_Cdots:
2558 \group_end:
2559 }
2560 }

```

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`.

```

2561 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2562 {
2563     \bool_if:NTF \l_@@_initial_open_bool
2564     {
2565         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2566         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2567         \dim_add:Nn \l_@@_x_initial_dim \col@sep
2568     }
2569     { \@@_set_initial_coords_from_anchor:n { mid-east } }
2570 \bool_if:NTF \l_@@_final_open_bool
2571 {
2572     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2573     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2574     \dim_sub:Nn \l_@@_x_final_dim \col@sep
2575 }
2576 { \@@_set_final_coords_from_anchor:n { mid-west } }
2577 \bool_lazy_and:nnTF
2578 \l_@@_initial_open_bool
2579 \l_@@_final_open_bool
2580 {
2581     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2582     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2583     \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2584     \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2585     \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2586 }
2587 {
2588     \bool_if:NT \l_@@_initial_open_bool
2589     { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2590     \bool_if:NT \l_@@_final_open_bool
2591     { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2592 }
2593 \@@_draw_line:
2594 }

```

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.

```

2595 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2596 {
2597   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2598   {
2599     \@@_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.

```

2600     \group_begin:
2601       \int_compare:nNnTF { #2 } = 0
2602       { \color { nicematrix-first-col } }
2603       {
2604         \int_compare:nNnT { #2 } = \l_@@_last_col_int
2605         { \color { nicematrix-last-col } }
2606       }
2607       \keys_set:nn { NiceMatrix / xdots } { #3 }
2608       \tl_if_empty:VF \l_@@_xdots_color_tl
2609       { \color { \l_@@_xdots_color_tl } }
2610       \@@_actually_draw_Vdots:
2611     \group_end:
2612   }
2613 }

```

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`.

```

2614 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2615 {

```

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

```

2616   \bool_set_false:N \l_tmpa_bool
2617   \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2618   {
2619     \@@_set_initial_coords_from_anchor:n { south-west }
2620     \@@_set_final_coords_from_anchor:n { north-west }
2621     \bool_set:Nn \l_tmpa_bool
2622     { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2623   }

```

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

```

2624   \bool_if:NTF \l_@@_initial_open_bool
2625   {
2626     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2627     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2628   }
2629   { \@@_set_initial_coords_from_anchor:n { south } }
2630   \bool_if:NTF \l_@@_final_open_bool
2631   {
2632     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2633     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2634   }

```

```

2635 { \@@_set_final_coords_from_anchor:n { north } }
2636 \bool_if:NTF \l_@@_initial_open_bool
2637 {
2638   \bool_if:NTF \l_@@_final_open_bool
2639   {
2640     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2641     \dim_set_eq:NN \l_tmpa_dim \pgf@x
2642     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2643     \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2644     \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.

```

2645   \int_compare:nNnT \l_@@_last_col_int > { -2 }
2646   {
2647     \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2648     {
2649       \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2650       \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2651       \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2652       \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2653     }
2654   }
2655 }
2656 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2657 }
2658 {
2659   \bool_if:NTF \l_@@_final_open_bool
2660   { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2661   {

```

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.

```

2662   \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2663   {
2664     \dim_set:Nn \l_@@_x_initial_dim
2665     {
2666       \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2667       \l_@@_x_initial_dim \l_@@_x_final_dim
2668     }
2669     \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2670   }
2671 }
2672 }
2673 \@@_draw_line:
2674 }

```

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.

```

2675 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2676 {
2677   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2678   {
2679     \@@_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.

```

2680   \group_begin:
2681   \keys_set:nn { NiceMatrix / xdots } { #3 }

```

```

2682         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2683         \@@_actually_draw_Ddots:
2684     \group_end:
2685 }
2686 }

```

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.`

```

2687 \cs_new_protected:Npn \@@_actually_draw_Ddots:
2688 {
2689     \bool_if:NTF \l_@@_initial_open_bool
2690     {
2691         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2692         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2693         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2694         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2695     }
2696     { \@@_set_initial_coords_from_anchor:n { south-east } } }
2697 \bool_if:NTF \l_@@_final_open_bool
2698 {
2699     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2700     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2701     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2702     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2703 }
2704 { \@@_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.

```

2705     \bool_if:NT \l_@@_parallelize_diags_bool
2706     {
2707         \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).

```

2708         \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.

```

2709     {
2710         \dim_gset:Nn \g_@@_delta_x_one_dim
2711         { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2712         \dim_gset:Nn \g_@@_delta_y_one_dim
2713         { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2714     }

```

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`.

```

2715     {
2716         \dim_set:Nn \l_@@_y_final_dim
2717         {
2718             \l_@@_y_initial_dim +
2719             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2720             \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim

```

```

2721     }
2722   }
2723 }
2724 \@@_draw_line:
2725 }

```

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.

```

2726 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2727 {
2728   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2729   {
2730     \@@_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.

```

2731     \group_begin:
2732     \keys_set:nn { NiceMatrix / xdots } { #3 }
2733     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2734     \@@_actually_draw_Iddots:
2735   \group_end:
2736 }
2737 }

```

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.`

```

2738 \cs_new_protected:Npn \@@_actually_draw_Iddots:
2739 {
2740   \bool_if:NTF \l_@@_initial_open_bool
2741   {
2742     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2743     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2744     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2745     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2746   }
2747   { \@@_set_initial_coords_from_anchor:n { south-west } }
2748   \bool_if:NTF \l_@@_final_open_bool
2749   {
2750     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2751     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2752     \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2753     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2754   }
2755   { \@@_set_final_coords_from_anchor:n { north-east } }
2756   \bool_if:NT \l_@@_parallelize_diags_bool
2757   {
2758     \int_gincr:N \g_@@_iddots_int
2759     \int_compare:nNnTF \g_@@_iddots_int = 1
2760     {
2761       \dim_gset:Nn \g_@@_delta_x_two_dim
2762       { \l_@@_x_final_dim - \l_@@_x_initial_dim }

```

```

2763         \dim_gset:Nn \g_@@_delta_y_two_dim
2764         { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2765     }
2766     {
2767         \dim_set:Nn \l_@@_y_final_dim
2768         {
2769             \l_@@_y_initial_dim +
2770             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2771             \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2772         }
2773     }
2774 }
2775 \@@_draw_line:
2776 }

```

## 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`

```

2777 \cs_new_protected:Npn \@@_draw_line:
2778 {
2779     \pgfrememberpicturepositiononpagetrue
2780     \pgf@relevantforpicturesizefalse
2781     \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
2782         \@@_draw_standard_dotted_line:
2783         \@@_draw_non_standard_dotted_line:
2784 }

```

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.

```

2785 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2786 {
2787     \begin { scope }
2788     \exp_args:No \@@_draw_non_standard_dotted_line:n
2789     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2790 }

```

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.

```

2791 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2792 {
2793     \draw
2794     [
2795         #1 ,
2796         shorten-> = \l_@@_xdots_shorten_dim ,
2797         shorten-< = \l_@@_xdots_shorten_dim ,
2798     ]
2799     ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
2800     -- node [ sloped , above ]

```



```

2801         { \c_math_toggle_token \scriptstyle \l_@@_xdots_up_tl \c_math_toggle_token }
2802     node [ sloped , below ]
2803     {
2804         \c_math_toggle_token
2805         \scriptstyle \l_@@_xdots_down_tl
2806         \c_math_toggle_token
2807     }
2808     ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2809 \end { scope }
2810 }

```

The command `\@@_draw_standard_dotted_line`: draws the line with our system of points (which give a dotted line with real round points).

```

2811 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2812 {

```

First, we put the labels.

```

2813     \bool_lazy_and:nnF
2814     { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2815     { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2816     {
2817         \pgfscope
2818         \pgftransformshift
2819         {
2820             \pgfpointlineattime { 0.5 }
2821             { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2822             { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2823         }
2824         \pgftransformrotate
2825         {
2826             \fp_eval:n
2827             {
2828                 atand
2829                 (
2830                     \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2831                     \l_@@_x_final_dim - \l_@@_x_initial_dim
2832                 )
2833             }
2834         }
2835         \pgfnode
2836         { rectangle }
2837         { south }
2838         {
2839             \c_math_toggle_token
2840             \scriptstyle \l_@@_xdots_up_tl
2841             \c_math_toggle_token
2842         }
2843         { }
2844         { \pgfusepath { } }
2845         \pgfnode
2846         { rectangle }
2847         { north }
2848         {
2849             \c_math_toggle_token
2850             \scriptstyle \l_@@_xdots_down_tl
2851             \c_math_toggle_token
2852         }
2853         { }
2854         { \pgfusepath { } }
2855         \endpgfscope
2856     }
2857     \pgfrememberpicturepositiononpagetrue
2858     \pgf@relevantforpicturesizefalse
2859     \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.

```

2860 \dim_zero_new:N \l_@@_l_dim
2861 \dim_set:Nn \l_@@_l_dim
2862 {
2863   \fp_to_dim:n
2864   {
2865     sqrt
2866     (
2867       ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
2868       +
2869       ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2870     )
2871   }
2872 }

```

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.

```

2873 \bool_lazy_or:nnF
2874 { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2875 { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2876 \@@_draw_standard_dotted_line_i:
2877 \group_end:
2878 }
2879 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2880 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2881 {

```

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

```

2882 \bool_if:NTF \l_@@_initial_open_bool
2883 {
2884   \bool_if:NTF \l_@@_final_open_bool
2885   {
2886     \int_set:Nn \l_tmpa_int
2887     { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2888   }
2889   {
2890     \int_set:Nn \l_tmpa_int
2891     {
2892       \dim_ratio:nn
2893       { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2894       \l_@@_inter_dots_dim
2895     }
2896   }
2897 }
2898 {
2899   \bool_if:NTF \l_@@_final_open_bool
2900   {
2901     \int_set:Nn \l_tmpa_int
2902     {
2903       \dim_ratio:nn
2904       { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2905       \l_@@_inter_dots_dim
2906     }
2907   }
2908   {
2909     \int_set:Nn \l_tmpa_int
2910     {
2911       \dim_ratio:nn
2912       { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2913       \l_@@_inter_dots_dim

```

```

2914     }
2915   }
2916 }

```

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

```

2917   \dim_set:Nn \l_tmpa_dim
2918   {
2919     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2920     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2921   }
2922   \dim_set:Nn \l_tmpb_dim
2923   {
2924     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2925     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2926   }

```

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`.

```

2927   \int_set:Nn \l_tmpb_int
2928   {
2929     \bool_if:NTF \l_@@_initial_open_bool
2930     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
2931     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
2932   }

```

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.

```

2933   \dim_gadd:Nn \l_@@_x_initial_dim
2934   {
2935     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2936     \dim_ratio:nn
2937     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2938     { 2 \l_@@_l_dim }
2939     * \l_tmpb_int
2940   }
2941   \dim_gadd:Nn \l_@@_y_initial_dim
2942   {
2943     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2944     \dim_ratio:nn
2945     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2946     { 2 \l_@@_l_dim }
2947     * \l_tmpb_int
2948   }
2949   \pgf@relevantforpicturesizefalse
2950   \int_step_inline:nnn 0 \l_tmpa_int
2951   {
2952     \pgfpathcircle
2953     { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2954     { \l_@@_radius_dim }
2955     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2956     \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
2957   }
2958   \pgfusepathqfill
2959 }

```

## 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 environ-

ments 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.

```

2960 \AtBeginDocument
2961 {
2962   \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
2963   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2964   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
2965     {
2966       \int_compare:nNnTF \c@jCol = 0
2967         { \@@_error:nn { in~first~col } \Ldots }
2968         {
2969           \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2970             { \@@_error:nn { in~last~col } \Ldots }
2971             {
2972               \@@_instruction_of_type:nnn \c_false_bool { \Ldots }
2973               { #1 , down = #2 , up = #3 }
2974             }
2975         }
2976       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ldots }
2977       \bool_gset_true:N \g_@@_empty_cell_bool
2978     }

2979   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
2980     {
2981       \int_compare:nNnTF \c@jCol = 0
2982         { \@@_error:nn { in~first~col } \Cdots }
2983         {
2984           \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2985             { \@@_error:nn { in~last~col } \Cdots }
2986             {
2987               \@@_instruction_of_type:nnn \c_false_bool { \Cdots }
2988               { #1 , down = #2 , up = #3 }
2989             }
2990         }
2991       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_cdots }
2992       \bool_gset_true:N \g_@@_empty_cell_bool
2993     }

2994   \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
2995     {
2996       \int_compare:nNnTF \c@iRow = 0
2997         { \@@_error:nn { in~first~row } \Vdots }
2998         {
2999           \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
3000             { \@@_error:nn { in~last~row } \Vdots }
3001             {
3002               \@@_instruction_of_type:nnn \c_false_bool { \Vdots }
3003               { #1 , down = #2 , up = #3 }
3004             }
3005         }
3006       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_vdots }
3007       \bool_gset_true:N \g_@@_empty_cell_bool
3008     }

```

```

3009 \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
3010 {
3011   \int_case:nnF \c@iRow
3012   {
3013     0 { \@@_error:nn { in~first~row } \Ddots }
3014     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
3015   }
3016   {
3017     \int_case:nnF \c@jCol
3018     {
3019       0 { \@@_error:nn { in~first~col } \Ddots }
3020       \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
3021     }
3022     {
3023       \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3024       \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Ddots }
3025       { #1 , down = #2 , up = #3 }
3026     }
3027   }
3028 }
3029 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ddots }
3030 \bool_gset_true:N \g_@@_empty_cell_bool
3031 }

3032 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
3033 {
3034   \int_case:nnF \c@iRow
3035   {
3036     0 { \@@_error:nn { in~first~row } \Iddots }
3037     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
3038   }
3039   {
3040     \int_case:nnF \c@jCol
3041     {
3042       0 { \@@_error:nn { in~first~col } \Iddots }
3043       \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
3044     }
3045     {
3046       \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3047       \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Iddots }
3048       { #1 , down = #2 , up = #3 }
3049     }
3050   }
3051 }
3052 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_iddots }
3053 \bool_gset_true:N \g_@@_empty_cell_bool
3054 }
3055 }

```

End of the \AtBeginDocument.

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

```

3056 \keys_define:nn { NiceMatrix / Ddots }
3057 {
3058   draw-first .bool_set:N = \l_@@_draw_first_bool ,
3059   draw-first .default:n = true ,
3060   draw-first .value_forbidden:n = true
3061 }

```

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

```

3062 \cs_new_protected:Npn \@@_Hspace:
3063 {

```

```

3064 \bool_gset_true:N \g_@@_empty_cell_bool
3065 \hspace
3066 }

```

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

```

3067 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
3068 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
3069 {
3070 % \begin{macrocode}
3071 % We have to act in an expandable way since it will begin by a |\multicolumn|.
3072 % \end{macrocode}
3073 \exp_args:NNe
3074 \@@_old_multicolumn
3075 { #1 }

```

We will have to replace \tl\_lower\_case:n in the future since it seems to be deprecated.

```

3076 {
3077 \exp_args:Ne \str_case:nn { \str_foldcase:n { #2 } }
3078 {
3079 1 { > \@@_Cell: 1 < \@@_end_Cell: }
3080 r { > \@@_Cell: r < \@@_end_Cell: }
3081 c { > \@@_Cell: c < \@@_end_Cell: }
3082 { 1 | } { > \@@_Cell: 1 < \@@_end_Cell: | }
3083 { r | } { > \@@_Cell: r < \@@_end_Cell: | }
3084 { c | } { > \@@_Cell: c < \@@_end_Cell: | }
3085 { | 1 } { | > \@@_Cell: 1 < \@@_end_Cell: }
3086 { | r } { | > \@@_Cell: r < \@@_end_Cell: }
3087 { | c } { | > \@@_Cell: c < \@@_end_Cell: }
3088 { | 1 | } { | > \@@_Cell: 1 < \@@_end_Cell: | }
3089 { | r | } { | > \@@_Cell: r < \@@_end_Cell: | }
3090 { | c | } { | > \@@_Cell: c < \@@_end_Cell: | }
3091 }
3092 }
3093 { #3 }

```

The \peek\_remove\_spaces:n is mandatory.

```

3094 \peek_remove_spaces:n
3095 {
3096 \int_compare:nNnT #1 > 1
3097 {
3098 \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
3099 { \int_use:N \c@iRow - \int_use:N \c@jCol }
3100 \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
3101 \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
3102 {
3103 { \int_use:N \c@iRow }
3104 { \int_use:N \c@jCol }
3105 { \int_use:N \c@iRow }
3106 { \int_eval:n { \c@jCol + #1 - 1 } }
3107 }
3108 }
3109 \int_gadd:Nn \c@jCol { #1 - 1 }
3110 \int_compare:nNnT \c@jCol > \g_@@_col_total_int
3111 { \int_gset_eq:NN \g_@@_col_total_int \c@jCol }
3112 }
3113 }

```

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.

```

3114 \cs_new:Npn \@@_Hdotsfor:

```

```

3115 {
3116   \bool_lazy_and:nnTF
3117   { \int_compare_p:nNn \c@jCol = 0 }
3118   { \int_compare_p:nNn \l_@@_first_col_int = 0 }
3119   {
3120     \bool_if:NTF \g_@@_after_col_zero_bool
3121     {
3122       \multicolumn { 1 } { c } { }
3123       \@@_Hdotsfor_i
3124     }
3125     { \@@_fatal:n { Hdotsfor~in~col~0 } }
3126   }
3127   {
3128     \multicolumn { 1 } { c } { }
3129     \@@_Hdotsfor_i
3130   }
3131 }

```

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:`).

```

3132 \AtBeginDocument
3133 {
3134   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3135   \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.

```

3136   \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
3137   {
3138     \tl_gput_right:Nx \g_@@_HVDotsfor_lines_tl
3139     {
3140       \@@_Hdotsfor:nnnn
3141       { \int_use:N \c@iRow }
3142       { \int_use:N \c@jCol }
3143       { #2 }
3144       {
3145         #1 , #3 ,
3146         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3147       }
3148     }
3149     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }
3150   }
3151 }

```

Enf of `\AtBeginDocument`.

```

3152 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
3153 {
3154   \bool_set_false:N \l_@@_initial_open_bool
3155   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

3156   \int_set:Nn \l_@@_initial_i_int { #1 }
3157   \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

3158   \int_compare:nNnTF #2 = 1
3159   {
3160     \int_set:Nn \l_@@_initial_j_int 1
3161     \bool_set_true:N \l_@@_initial_open_bool
3162   }
3163   {
3164     \cs_if_exist:cTF
3165     {

```

```

3166         pgf @ sh @ ns @ \@@_env:
3167         - \int_use:N \l_@@_initial_i_int
3168         - \int_eval:n { #2 - 1 }
3169     }
3170     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
3171     {
3172         \int_set:Nn \l_@@_initial_j_int { #2 }
3173         \bool_set_true:N \l_@@_initial_open_bool
3174     }
3175 }
3176 \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
3177 {
3178     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3179     \bool_set_true:N \l_@@_final_open_bool
3180 }
3181 {
3182     \cs_if_exist:cTF
3183     {
3184         pgf @ sh @ ns @ \@@_env:
3185         - \int_use:N \l_@@_final_i_int
3186         - \int_eval:n { #2 + #3 }
3187     }
3188     { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
3189     {
3190         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3191         \bool_set_true:N \l_@@_final_open_bool
3192     }
3193 }
3194 \group_begin:
3195 \int_compare:nNnTF { #1 } = 0
3196 { \color { nicematrix-first-row } }
3197 {
3198     \int_compare:nNnT { #1 } = \g_@@_row_total_int
3199     { \color { nicematrix-last-row } }
3200 }
3201 \keys_set:nn { NiceMatrix / xdots } { #4 }
3202 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3203 \@@_actually_draw_Ldots:
3204 \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).

```

3205     \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
3206     { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
3207 }

3208 \AtBeginDocument
3209 {
3210     \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3211     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3212     \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
3213     {
3214         \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
3215         {
3216             \@@_Vdotsfor:nnnn
3217             { \int_use:N \c@iRow }
3218             { \int_use:N \c@jCol }
3219             { #2 }
3220             {
3221                 #1 , #3 ,
3222                 down = \exp_not:n { #4 } , up = \exp_not:n { #5 }

```



```

3223     }
3224   }
3225 }
3226 }

```

Enf of \AtBeginDocument.

```

3227 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
3228 {
3229   \bool_set_false:N \l_@@_initial_open_bool
3230   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it's easy.

```

3231   \int_set:Nn \l_@@_initial_j_int { #2 }
3232   \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

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

```

3233   \int_compare:nNnTF #1 = 1
3234   {
3235     \int_set:Nn \l_@@_initial_i_int 1
3236     \bool_set_true:N \l_@@_initial_open_bool
3237   }
3238   {
3239     \cs_if_exist:cTF
3240     {
3241       pgf @ sh @ ns @ \@@_env:
3242       - \int_eval:n { #1 - 1 }
3243       - \int_use:N \l_@@_initial_j_int
3244     }
3245     { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
3246     {
3247       \int_set:Nn \l_@@_initial_i_int { #1 }
3248       \bool_set_true:N \l_@@_initial_open_bool
3249     }
3250   }
3251   \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
3252   {
3253     \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3254     \bool_set_true:N \l_@@_final_open_bool
3255   }
3256   {
3257     \cs_if_exist:cTF
3258     {
3259       pgf @ sh @ ns @ \@@_env:
3260       - \int_eval:n { #1 + #3 }
3261       - \int_use:N \l_@@_final_j_int
3262     }
3263     { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
3264     {
3265       \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3266       \bool_set_true:N \l_@@_final_open_bool
3267     }
3268   }
3269   \group_begin:
3270   \int_compare:nNnTF { #2 } = 0
3271   { \color { nicematrix-first-col } }
3272   {
3273     \int_compare:nNnT { #2 } = \g_@@_col_total_int
3274     { \color { nicematrix-last-col } }
3275   }
3276   \keys_set:nn { NiceMatrix / xdots } { #4 }
3277   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3278   \@@_actually_draw_Vdots:
3279   \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).

```

3280 \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
3281 { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
3282 }

```

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

```

3283 \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>48</sup>

```

3284 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
3285 { \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).

```

3286 \AtBeginDocument
3287 {
3288 \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
3289 \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3290 \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
3291 {
3292 \group_begin:
3293 \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
3294 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3295 \use:e
3296 {
3297 \@@_line_i:nn
3298 { \@@_double_int_eval:n #2 \q_stop }
3299 { \@@_double_int_eval:n #3 \q_stop }
3300 }
3301 \group_end:
3302 }
3303 }
3304 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3305 {
3306 \bool_set_false:N \l_@@_initial_open_bool
3307 \bool_set_false:N \l_@@_final_open_bool
3308 \bool_if:nTF
3309 {
3310 \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3311 ||
3312 \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3313 }
3314 {
3315 \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
3316 }

```

---

<sup>48</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`.

```

3317     { \@@_draw_line_ii:nn { #1 } { #2 } }
3318   }
3319 \AtBeginDocument
3320 {
3321   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3322   {

```

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:`.

```

3323     \c_@@_pgfortikzpicture_tl
3324     \@@_draw_line_iii:nn { #1 } { #2 }
3325     \c_@@_endpgfortikzpicture_tl
3326   }
3327 }

```

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

```

3328 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3329 {
3330   \pgfrememberpicturepositiononpagetrue
3331   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3332   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3333   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3334   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3335   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3336   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3337   \@@_draw_line:
3338 }

```

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

In the beginning of the code-before, the command `\@@_rowcolor:nn` will be linked to `\rowcolor` and the command `\@@_columncolor:nn` to `\columncolor`.

```

3339 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3340 {
3341   \tl_set:Nn \l_tmpa_tl { #1 }
3342   \tl_set:Nn \l_tmpb_tl { #2 }
3343 }

```

The following command uses two implicit arguments : `\l_@@_rows_tl` and `\l_@@_cols_tl` which are specifications for a set of row and a set of columns. The cartesian product is filled with the current color.

```

3344 \cs_new_protected:Npn \@@_cartesian_fill:
3345 {

```

We begin the loop over the columns.

```

3346   \clist_map_inline:Nn \l_@@_cols_tl
3347   {
3348     \tl_set:Nn \l_tmpa_tl { ##1 }
3349     \tl_if_in:NnTF \l_tmpa_tl { - }
3350     { \@@_cut_on_hyphen:w ##1 \q_stop }
3351     { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3352     \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3353     \tl_if_empty:NT \l_tmpb_tl
3354     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3355     \int_compare:nNnT \l_tmpb_tl > \c@jCol
3356     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }

```

```

3357 \@@_qpoint:n { col - \l_tmpa_tl }
3358 \bool_lazy_and:nnTF
3359 { \str_if_eq_p:Vn \l_tmpa_tl 0 }
3360 { \int_compare_p:nNn \l_@@_first_col_int > 0 }
3361 { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3362 { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3363 \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3364 \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }

```

We begin the loop over the rows

```

3365 \clist_map_inline:Nn \l_@@_rows_tl
3366 {
3367   \tl_set:Nn \l_tmpa_tl { ####1 }
3368   \tl_if_in:NnTF \l_tmpa_tl { - }
3369   { \@@_cut_on_hyphen:w ####1 \q_stop }
3370   { \@@_cut_on_hyphen:w ####1 - ####1 \q_stop }
3371   \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3372   \tl_if_empty:NT \l_tmpb_tl
3373   { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3374   \int_compare:nNnT \l_tmpb_tl > \c@iRow
3375   { \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.

```

3376 \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3377 \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3378 \@@_qpoint:n { row - \l_tmpa_tl }
3379 \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3380 \pgfpathrectanglecorners
3381 { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3382 { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3383 }
3384 }
3385 \pgfusepathqfill
3386 }

```

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

```

3387 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3388 {
3389   \tl_if_blank:nF { #2 }
3390   {
3391     \pgfpicture
3392     \pgf@relevantforpicturesizefalse
3393     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3394     \tl_set:Nn \l_@@_rows_tl { #3 }
3395     \tl_set:Nn \l_@@_cols_tl { - }

```

The command \@@\_cartesian\_fill: takes in two implicit arguments: \l\_@@\_cols\_tl and \l\_@@\_row\_tl.

```

3396 \@@_cartesian_fill:
3397 \endpgfpicture
3398 }
3399 }

```

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

```

3400 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3401 {
3402   \tl_if_blank:nF { #2 }
3403   {
3404     \pgfpicture
3405     \pgf@relevantforpicturesizefalse
3406     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3407     \tl_set:Nn \l_@@_rows_tl { - }
3408     \tl_set:Nn \l_@@_cols_tl { #3 }

```

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

```

3409     \@@_cartesian_fill:
3410     \endpgfpicture
3411   }
3412 }

```

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

```

3413 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3414 {
3415   \tl_if_blank:nF { #2 }
3416   {
3417     \pgfpicture
3418     \pgf@relevantforpicturesizefalse
3419     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3420     \@@_rectanglecolor_i:nn { #3 } { #4 }
3421     \endpgfpicture
3422   }
3423 }

```

```

3424 \cs_new_protected:Npn \@@_rectanglecolor_i:nn #1 #2
3425 {
3426   \@@_cut_on_hyphen:w #1 \q_stop
3427   \tl_clear_new:N \l_tmpc_tl
3428   \tl_clear_new:N \l_tmpd_tl
3429   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
3430   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
3431   \@@_cut_on_hyphen:w #2 \q_stop
3432   \tl_set:Nx \l_@@_rows_tl { \l_tmpc_tl - \l_tmpa_tl }
3433   \tl_set:Nx \l_@@_cols_tl { \l_tmpd_tl - \l_tmpb_tl }

```

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

```

3434     \@@_cartesian_fill:
3435   }

```

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

```

3436 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3437 {
3438   \tl_if_blank:nF { #2 }
3439   {
3440     \pgfpicture
3441     \pgf@relevantforpicturesizefalse
3442     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3443     \clist_map_inline:nn { #3 }
3444     { \@@_rectanglecolor_i:nn { ##1 } { ##1 } }
3445     \pgfusepathqfill
3446     \endpgfpicture
3447   }
3448 }

```

```

3449 \keys_define:nn { NiceMatrix / rowcolors }
3450 {
3451   respect-blocks .bool_set:N = \l_@@_respect_blocks_bool ,
3452   respect-blocks .default:n = true ,
3453   cols .tl_set:N = \l_@@_cols_tl ,
3454   restart .bool_set:N = \l_@@_rowcolors_restart_bool ,
3455   restart .default:n = true ,
3456   unknown .code:n = \@@_error:n { Unknown-option-for-rowcolors }
3457 }

```

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.

```
3458 \NewDocumentCommand \@@_rowcolors { 0 { } m m m 0 { } }
3459 {
```

The group is for the options.

```
3460 \group_begin:
3461 \tl_clear_new:N \l_@@_cols_tl
3462 \tl_set:Nn \l_@@_cols_tl { - }
3463 \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.

```
3464 \bool_set_true:N \l_tmpa_bool
3465 \bool_lazy_and:nnT
3466 \l_@@_respect_blocks_bool
3467 {
3468 \cs_if_exist_p:c
3469 { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq } }
3470 {
```

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`).

```
3471 \seq_set_eq:Nc \l_tmpb_seq
3472 { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3473 \seq_set_filter:Nn \l_tmpa_seq \l_tmpb_seq
3474 { \@@_not_in_exterior_p:nnnn ##1 }
3475 }
3476 \pgfpicture
3477 \pgf@relevantforpicturesizefalse
3478 \clist_map_inline:nn { #2 }
3479 {
3480 \tl_set:Nn \l_tmpa_tl { ##1 }
3481 \tl_if_in:NnTF \l_tmpa_tl { - }
3482 { \@@_cut_on_hyphen:w ##1 \q_stop }
3483 { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
```

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

```
3484 \int_set:Nn \l_tmpa_int \l_tmpa_tl
3485 \bool_if:NTF \l_@@_rowcolors_restart_bool
3486 { \bool_set_true:N \l_tmpa_bool }
3487 { \bool_set:Nn \l_tmpa_bool { \int_if_odd_p:n { \l_tmpa_tl } } }
3488 \int_zero_new:N \l_tmpc_int
3489 \int_set:Nn \l_tmpc_int \l_tmpb_tl
3490 \int_do_until:nNnn \l_tmpa_int > \l_tmpc_int
3491 {
```

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

```
3492 \int_set_eq:NN \l_tmpb_int \l_tmpa_int
```

If the key `respect-blocks` is in force, we have to adjust that value (of course).

```
3493 \bool_lazy_and:nnT
3494 \l_@@_respect_blocks_bool
3495 {
3496 \cs_if_exist_p:c
3497 { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3498 }
3499 {
3500 \seq_set_filter:Nn \l_tmpb_seq \l_tmpa_seq
3501 { \@@_intersect_our_row_p:nnnn ####1 }
3502 \seq_map_inline:Nn \l_tmpb_seq { \@@_rowcolors_i:nnnn ####1 }
```

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

```

3503     }
3504     \tl_set:Nx \l_@@_rows_tl
3505     { \int_use:N \l_tmpa_int - \int_use:N \l_tmpb_int }
3506     \bool_if:NTF \l_tmpa_bool
3507     {
3508         \tl_if_blank:nF { #3 }
3509         {
3510             \tl_if_empty:nTF { #1 }
3511             \color
3512             { \color [ #1 ] }
3513             { #3 }

```

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

```

3514         \@@_cartesian_fill:
3515     }
3516     \bool_set_false:N \l_tmpa_bool
3517 }
3518 {
3519     \tl_if_blank:nF { #4 }
3520     {
3521         \tl_if_empty:nTF { #1 }
3522         \color
3523         { \color [ #1 ] }
3524         { #4 }

```

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

```

3525         \@@_cartesian_fill:
3526     }
3527     \bool_set_true:N \l_tmpa_bool
3528 }
3529     \int_set:Nn \l_tmpa_int { \l_tmpb_int + 1 }
3530 }
3531 }
3532 \endpgfpicture
3533 \group_end:
3534 }

```

```

3535 \cs_new_protected:Npn \@@_rowcolors_i:nnnn #1 #2 #3 #4
3536 {
3537     \int_compare:nNnT { #3 } > \l_tmpb_int
3538     { \int_set:Nn \l_tmpb_int { #3 } }
3539 }

```

```

3540 \prg_new_conditional:Nnn \@@_not_in_exterior:nnnn p
3541 {
3542     \bool_lazy_or:nnTF
3543     { \int_compare_p:nNn { #4 } = \c_zero_int }
3544     { \int_compare_p:nNn { #2 } = { \@@_succ:n { \c@jCol } } }
3545     \prg_return_false:
3546     \prg_return_true:
3547 }

```

The following command return true when the block intersects the row `\l_tmpa_int`.

```

3548 \prg_new_conditional:Nnn \@@_intersect_our_row:nnnn p
3549 {
3550     \bool_if:nTF
3551     {
3552         \int_compare_p:n { #1 <= \l_tmpa_int }

```

```

3553         &&
3554         \int_compare_p:n { \l_tmpa_int <= #3 }
3555     }
3556     \prg_return_true:
3557     \prg_return_false:
3558 }

3559 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3560 {
3561     \int_step_inline:nn { \int_use:N \c@iRow }
3562     {
3563         \int_step_inline:nn { \int_use:N \c@jCol }
3564         {
3565             \int_if_even:nTF { ####1 + ##1 }
3566             { \@@_cellcolor [ #1 ] { #2 } }
3567             { \@@_cellcolor [ #1 ] { #3 } }
3568             { ##1 - ####1 }
3569         }
3570     }
3571 }

```

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

```

3572 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
3573 {
3574     \tl_gput_right:Nx \g_nicematrix_code_before_tl
3575     { \cellcolor [ #1 ] { #2 } { \int_use:N \c@iRow - \int_use:N \c@jCol } }
3576 }

```

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

```

3577 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
3578 {
3579     \tl_gput_right:Nx \g_nicematrix_code_before_tl
3580     {
3581         \exp_not:N \rectanglecolor [ #1 ] { #2 }
3582         { \int_use:N \c@iRow - \int_use:N \c@jCol }
3583         { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
3584     }
3585 }

```

```

3586 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
3587 {
3588     \int_compare:nNnT \c@iRow = 1
3589     {

```

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).

```

3590         \tl_gput_left:Nx \g_nicematrix_code_before_tl
3591         { \exp_not:N \columncolor [ #1 ] { #2 } { \int_use:N \c@jCol } }
3592     }
3593 }

```



## 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`.

```
3594 \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`.

```
3595 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
3596 {
3597   \int_compare:nNnTF \l_@@_first_col_int = 0
3598   { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3599   {
3600     \int_compare:nNnTF \c@jCol = 0
3601     {
3602       \int_compare:nNnF \c@iRow = { -1 }
3603       { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
3604     }
3605     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3606   }
3607 }
```

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.

```
3608 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
3609 {
3610   \int_compare:nNnF \c@iRow = 0
3611   { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
3612 }
```

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 `|`.

```
3613 \cs_new_protected:Npn \@@_vline:nn #1 #2
3614 {
```

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).

```
3615   \int_compare:nNnT { #1 } < { \c@jCol + 2 }
3616   {
3617     \pgfpicture
3618     \@@_vline_i:nn { #1 } { #2 }
3619     \endpgfpicture
3620   }
3621 }
```

```
3622 \cs_new_protected:Npn \@@_vline_i:nn #1 #2
3623 {
```

`\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`.

```
3624   \tl_set:Nx \l_tmpb_tl { #1 }
3625   \tl_clear_new:N \l_tmppc_tl
```

```

3626 \int_step_variable:nNn \c@iRow \l_tmpa_tl
3627 {

```

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.

```

3628 \bool_gset_true:N \g_tmpa_bool
3629 \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3630 { \@@_test_if_vline_in_block:nnnn ##1 }
3631 \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3632 { \@@_test_if_vline_in_block:nnnn ##1 }
3633 \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
3634 { \@@_test_if_vline_in_stroken_block:nnnn ##1 }
3635 \clist_if_empty:NF \l_@@_except_corners_clist
3636 \@@_test_in_corner_v:
3637 \bool_if:NTF \g_tmpa_bool
3638 {
3639 \tl_if_empty:NT \l_tmpc_tl

```

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

```

3640 { \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl }
3641 }
3642 {
3643 \tl_if_empty:NF \l_tmpc_tl
3644 {
3645 \@@_vline_ii:nnnn
3646 { #1 }
3647 { #2 }
3648 \l_tmpc_tl
3649 { \int_eval:n { \l_tmpa_tl - 1 } }
3650 \tl_clear:N \l_tmpc_tl
3651 }
3652 }
3653 }
3654 \tl_if_empty:NF \l_tmpc_tl
3655 {
3656 \@@_vline_ii:nnnn
3657 { #1 }
3658 { #2 }
3659 \l_tmpc_tl
3660 { \int_use:N \c@iRow }
3661 \tl_clear:N \l_tmpc_tl
3662 }
3663 }

```

```

3664 \cs_new_protected:Npn \@@_test_in_corner_v:
3665 {
3666 \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
3667 {
3668 \seq_if_in:NxT
3669 \l_@@_empty_corner_cells_seq
3670 { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3671 { \bool_set_false:N \g_tmpa_bool }
3672 }
3673 {
3674 \seq_if_in:NxT
3675 \l_@@_empty_corner_cells_seq
3676 { \l_tmpa_tl - \l_tmpb_tl }
3677 {
3678 \int_compare:nNnTF \l_tmpb_tl = 1
3679 { \bool_set_false:N \g_tmpa_bool }
3680 {

```

```

3681         \seq_if_in:NxT
3682         \l_@@_empty_corner_cells_seq
3683         { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3684         { \bool_set_false:N \g_tmpa_bool }
3685     }
3686 }
3687 }
3688 }

```

#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.

```

3689 \cs_new_protected:Npn \@@_vline_ii:nnnn #1 #2 #3 #4
3690 {
3691     \pgfrememberpicturepositiononpagetrue
3692     \pgf@relevantforpicturesizefalse
3693     \@@_qpoint:n { row - #3 }
3694     \dim_set_eq:NN \l_tmpa_dim \pgf@y
3695     \@@_qpoint:n { col - #1 }
3696     \dim_set_eq:NN \l_tmpb_dim \pgf@x
3697     \@@_qpoint:n { row - \@@_succ:n { #4 } }
3698     \dim_set_eq:NN \l_tmpc_dim \pgf@y
3699     \bool_lazy_and:nnT
3700     { \int_compare_p:nNn { #2 } > 1 }
3701     { ! \tl_if_blank_p:V \CT@drsc@ }
3702     {
3703         \group_begin:
3704         \CT@drsc@
3705         \dim_add:Nn \l_tmpa_dim { 0.5 \arrayrulewidth }
3706         \dim_sub:Nn \l_tmpc_dim { 0.5 \arrayrulewidth }
3707         \dim_set:Nn \l_tmpd_dim
3708         { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3709         \pgfpathrectanglecorners
3710         { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3711         { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
3712         \pgfusepathqfill
3713         \group_end:
3714     }
3715     \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3716     \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3717     \prg_replicate:nn { #2 - 1 }
3718     {
3719         \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3720         \dim_sub:Nn \l_tmpb_dim \doublerulesep
3721         \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3722         \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3723     }
3724     \CT@arc@
3725     \pgfsetlinewidth { 1.1 \arrayrulewidth }
3726     \pgfsetrectcap
3727     \pgfusepathqstroke
3728 }

```

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).

```

3729 \cs_new_protected:Npn \@@_vline_i_complete:nn #1 #2
3730 { \@@_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).

```

3731 \cs_new_protected:Npn \@@_draw_vlines:
3732 {
3733   \int_step_inline:nnn
3734     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3735     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3736     { \@@_vline:nn { ##1 } 1 }
3737 }

```

## 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.

```

3738 \cs_new_protected:Npn \@@_hline:nn #1 #2
3739 {
3740   \pgfpicture
3741     \@@_hline_i:nn { #1 } { #2 }
3742   \endpgfpicture
3743 }
3744 \cs_new_protected:Npn \@@_hline_i:nn #1 #2
3745 {

```

\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.

```

3746   \tl_set:Nn \l_tmpa_tl { #1 }
3747   \tl_clear_new:N \l_tmpc_tl
3748   \int_step_variable:nNn \c@jCol \l_tmpb_tl
3749   {

```

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.

```

3750     \bool_gset_true:N \g_tmpa_bool
3751     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3752       { \@@_test_if_hline_in_block:nnnn ##1 }
3753     \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3754       { \@@_test_if_hline_in_block:nnnn ##1 }
3755     \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
3756       { \@@_test_if_hline_in_stroken_block:nnnn ##1 }
3757     \clist_if_empty:NF \l_@@_except_corners_clist \@@_test_in_corner_h:
3758     \bool_if:NTF \g_tmpa_bool
3759     {
3760       \tl_if_empty:NT \l_tmpc_tl

```

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

```

3761       { \tl_set_eq:NN \l_tmpc_tl \l_tmpb_tl }
3762     }
3763   {
3764     \tl_if_empty:NF \l_tmpc_tl
3765     {
3766       \@@_hline_ii:nnnn
3767         { #1 }
3768         { #2 }
3769       \l_tmpc_tl
3770       { \int_eval:n { \l_tmpb_tl - 1 } }
3771       \tl_clear:N \l_tmpc_tl
3772     }
3773   }
3774 }
3775 \tl_if_empty:NF \l_tmpc_tl
3776 {

```

```

3777 \@@_hline_ii:nnnn
3778 { #1 }
3779 { #2 }
3780 \l_tmpc_tl
3781 { \int_use:N \c@jCol }
3782 \tl_clear:N \l_tmpc_tl
3783 }
3784 }

3785 \cs_new_protected:Npn \@@_test_in_corner_h:
3786 {
3787   \int_compare:nNnTF \l_tmpa_tl = { \@@_succ:n \c@iRow }
3788   {
3789     \seq_if_in:NxT
3790     \l_@@_empty_corner_cells_seq
3791     { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3792     { \bool_set_false:N \g_tmpa_bool }
3793   }
3794   {
3795     \seq_if_in:NxT
3796     \l_@@_empty_corner_cells_seq
3797     { \l_tmpa_tl - \l_tmpb_tl }
3798     {
3799       \int_compare:nNnTF \l_tmpa_tl = 1
3800       { \bool_set_false:N \g_tmpa_bool }
3801       {
3802         \seq_if_in:NxT
3803         \l_@@_empty_corner_cells_seq
3804         { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3805         { \bool_set_false:N \g_tmpa_bool }
3806       }
3807     }
3808   }
3809 }

```

#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.

```

3810 \cs_new_protected:Npn \@@_hline_ii:nnnn #1 #2 #3 #4
3811 {
3812   \pgfrememberpicturepositiononpagetrue
3813   \pgf@relevantforpicturesizefalse
3814   \@@_qpoint:n { col - #3 }
3815   \dim_set_eq:NN \l_tmpa_dim \pgf@x
3816   \@@_qpoint:n { row - #1 }
3817   \dim_set_eq:NN \l_tmpb_dim \pgf@y
3818   \@@_qpoint:n { col - \@@_succ:n { #4 } }
3819   \dim_set_eq:NN \l_tmpc_dim \pgf@x
3820   \bool_lazy_and:nnT
3821   { \int_compare_p:nNn { #2 } > 1 }
3822   { ! \tl_if_blank_p:V \CT@drsc@ }
3823   {
3824     \group_begin:
3825     \CT@drsc@
3826     \dim_set:Nn \l_tmpd_dim
3827     { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3828     \pgfpathrectanglecorners
3829     { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3830     { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3831     \pgfusepathqfill
3832     \group_end:
3833   }
3834   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }

```

```

3835 \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3836 \prg_replicate:nn { #2 - 1 }
3837 {
3838   \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3839   \dim_sub:Nn \l_tmpb_dim \doublerulesep
3840   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3841   \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3842 }
3843 \CT@arc@
3844 \pgfsetlinewidth { 1.1 \arrayrulewidth }
3845 \pgfsetrectcap
3846 \pgfusepathqstroke
3847 }

3848 \cs_new_protected:Npn \@@_hline_i_complete:nn #1 #2
3849 { \@@_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).

```

3850 \cs_new_protected:Npn \@@_draw_hlines:
3851 {
3852   \int_step_inline:nnn
3853   { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3854   { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3855   { \@@_hline:nn { ##1 } 1 }
3856 }

```

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

```

3857 \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.

```

3858 \cs_set:Npn \@@_Hline_i:n #1
3859 {
3860   \peek_meaning_ignore_spaces:NTF \Hline
3861   { \@@_Hline_ii:nn { #1 + 1 } }
3862   { \@@_Hline_iii:n { #1 } }
3863 }

3864 \cs_set:Npn \@@_Hline_ii:nn #1 #2 { \@@_Hline_i:n { #1 } }

3865 \cs_set:Npn \@@_Hline_iii:n #1
3866 {
3867   \skip_vertical:n
3868   {
3869     \arrayrulewidth * ( #1 )
3870     + \doublerulesep * ( \int_max:nn 0 { #1 - 1 } )
3871   }
3872   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3873   { \@@_hline:nn { \@@_succ:n { \c@iRow } } { #1 } }
3874   \ifnum 0 = ` { \fi }
3875 }

```

## 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.

```

3876 \cs_new_protected:Npn \@@_test_if_hline_in_block:nnnn #1 #2 #3 #4
3877 {

```

```

3878 \bool_lazy_all:nT
3879 {
3880   { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
3881   { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3882   { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3883   { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3884 }
3885 { \bool_gset_false:N \g_tmpa_bool }
3886 }

```

The same for vertical rules.

```

3887 \cs_new_protected:Npn \@@_test_if_vline_in_block:nnnn #1 #2 #3 #4
3888 {
3889   \bool_lazy_all:nT
3890   {
3891     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3892     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3893     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
3894     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3895   }
3896   { \bool_gset_false:N \g_tmpa_bool }
3897 }
3898 \cs_new_protected:Npn \@@_test_if_hline_in_stroken_block:nnnn #1 #2 #3 #4
3899 {
3900   \bool_lazy_all:nT
3901   {
3902     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3903     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 2 } }
3904     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3905     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3906   }
3907   { \bool_gset_false:N \g_tmpa_bool }
3908 }
3909 \cs_new_protected:Npn \@@_test_if_vline_in_stroken_block:nnnn #1 #2 #3 #4
3910 {
3911   \bool_lazy_all:nT
3912   {
3913     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3914     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3915     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3916     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 2 } }
3917   }
3918   { \bool_gset_false:N \g_tmpa_bool }
3919 }

```

## 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.

```

3920 \cs_new_protected:Npn \@@_compute_corners:
3921 {

```

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.

```

3922 \seq_clear_new:N \l_@@_empty_corner_cells_seq
3923 \clist_map_inline:Nn \l_@@_except_corners_clist
3924 {
3925   \str_case:nnF { ##1 }
3926   {
3927     { NW }
3928     { \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol }

```

```

3929         { NE }
3930         { \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1 }
3931         { SW }
3932         { \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol }
3933         { SE }
3934         { \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1 }
3935     }
3936     { \@@_error:nn { bad~corner } { ##1 } }
3937 }
3938 }

```

“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.

```

3939 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
3940 {

```

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.

```

3941     \bool_set_false:N \l_tmpa_bool
3942     \int_zero_new:N \l_@@_last_empty_row_int
3943     \int_set:Nn \l_@@_last_empty_row_int { #1 }
3944     \int_step_inline:nnnn { #1 } { #3 } { #5 }
3945     {
3946         \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
3947         \bool_lazy_or:nnTF
3948         {
3949             \cs_if_exist_p:c
3950             { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
3951         }
3952         \l_tmpb_bool
3953         { \bool_set_true:N \l_tmpa_bool }
3954         {
3955             \bool_if:NF \l_tmpa_bool
3956             { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
3957         }
3958     }

```

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

```

3959     \bool_set_false:N \l_tmpa_bool
3960     \int_zero_new:N \l_@@_last_empty_column_int
3961     \int_set:Nn \l_@@_last_empty_column_int { #2 }
3962     \int_step_inline:nnnn { #2 } { #4 } { #6 }
3963     {
3964         \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
3965         \bool_lazy_or:nnTF
3966         \l_tmpb_bool
3967         {
3968             \cs_if_exist_p:c
3969             { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
3970         }
3971         { \bool_set_true:N \l_tmpa_bool }

```



```

3972     {
3973         \bool_if:NF \l_tmpa_bool
3974         { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
3975     }
3976 }

```

Now, we loop over the rows.

```

3977 \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
3978 {

```

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

```

3979     \bool_set_false:N \l_tmpa_bool
3980     \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
3981     {
3982         \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
3983         \bool_lazy_or:nnTF
3984         \l_tmpb_bool
3985         {
3986             \cs_if_exist_p:c
3987             { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
3988         }
3989         { \bool_set_true:N \l_tmpa_bool }
3990         {
3991             \bool_if:NF \l_tmpa_bool
3992             {
3993                 \int_set:Nn \l_@@_last_empty_column_int { #####1 }
3994                 \seq_put_right:Nn
3995                 \l_@@_empty_corner_cells_seq
3996                 { ##1 - #####1 }
3997             }
3998         }
3999     }
4000 }
4001 }

```

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`).

```

4002 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
4003 {
4004     \int_set:Nn \l_tmpa_int { #1 }
4005     \int_set:Nn \l_tmpb_int { #2 }
4006     \bool_set_false:N \l_tmpb_bool
4007     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4008     { \@@_test_if_cell_in_block:nnnnnn \l_tmpa_int \l_tmpb_int ##1 }
4009 }
4010 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnn #1 #2 #3 #4 #5 #6
4011 {
4012     \int_compare:nNnT { #3 } < { \@@_succ:n { #1 } }
4013     {
4014         \int_compare:nNnT { #1 } < { \@@_succ:n { #5 } }
4015         {
4016             \int_compare:nNnT { #4 } < { \@@_succ:n { #2 } }
4017             {
4018                 \int_compare:nNnT { #2 } < { \@@_succ:n { #6 } }
4019                 { \bool_set_true:N \l_tmpb_bool }
4020             }
4021         }
4022     }
4023 }

```

## 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`.

```
4024 \cs_new:Npn \@@_hdottedline:
4025 {
4026   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
4027   \@@_hdottedline_i:
4028 }
```

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

```
4029 \cs_new_protected:Npn \@@_hdottedline_i:
4030 {
```

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).

```
4031   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4032   { \@@_hdottedline:n { \int_use:N \c@iRow } }
4033 }
```

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.

```
4034 \AtBeginDocument
4035 {
```

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}`).

```
4036   \cs_new_protected:Npx \@@_hdottedline:n #1
4037   {
4038     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
4039     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
4040     \c_@@_pgfortikzpicture_tl
4041     \@@_hdottedline_i:n { #1 }
4042     \c_@@_endpgfortikzpicture_tl
4043   }
4044 }
```

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

```
4045 \cs_new_protected:Npn \@@_hdottedline_i:n #1
4046 {
4047   \pgfrememberpicturepositiononpagetrue
4048   \@@_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).

```
4049   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
4050   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
4051   \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
```

The dotted line will be extended if the user uses `margin` (or `left-margin` and `right-margin`).

The aim is that, by standard the dotted line fits between square brackets (`\hline` doesn't).

```
\begin{bNiceMatrix}
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & 4 \\
\hdottedline
1 & 2 & 3 & 4
\end{bNiceMatrix}
```

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

But, if the user uses `margin`, the dotted line extends to have the same width as a `\hline`.

```
\begin{bNiceMatrix}[margin]
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & 4 \\
\hdottedline
1 & 2 & 3 & 4
\end{bNiceMatrix}
```

$$\left[ \begin{array}{cccc} 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 4 \\ \hdottedline 1 & 2 & 3 & 4 \end{array} \right]$$

```
4052 \@@_qpoint:n { col - 1 }
4053 \dim_set:Nn \l_@@_x_initial_dim
4054 {
4055 \pgf@x +
```

We do a reduction by `\arraycolsep` for the environments with delimiters (and not for the other).

```
4056 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4057 - \l_@@_left_margin_dim
4058 }
4059 \@@_qpoint:n { col - \@@_succ:n \c@jCol }
4060 \dim_set:Nn \l_@@_x_final_dim
4061 {
4062 \pgf@x -
4063 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4064 + \l_@@_right_margin_dim
4065 }
```

For reasons purely aesthetic, we do an adjustment in the case of a rounded bracket. The correction by `0.5 \l_@@_inter_dots_dim` is *ad hoc* for a better result.

```
4066 \tl_set:Nn \l_tmpa_tl { ( }
4067 \tl_if_eq:NMF \l_@@_left_delim_tl \l_tmpa_tl
4068 { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
4069 \tl_set:Nn \l_tmpa_tl { ) }
4070 \tl_if_eq:NMF \l_@@_right_delim_tl \l_tmpa_tl
4071 { \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`.

```
4072 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4073 \@@_draw_line:
4074 }
```

## Vertical dotted lines

```
4075 \cs_new_protected:Npn \@@_vdottedline:n #1
4076 {
4077 \bool_set_true:N \l_@@_initial_open_bool
4078 \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”.

```
4079 \bool_if:NTF \c_@@_tikz_loaded_bool
4080 {
4081 \tikzpicture
4082 \@@_vdottedline_i:n { #1 }
4083 \endtikzpicture
4084 }
4085 {
4086 \pgfpicture
4087 \@@_vdottedline_i:n { #1 }
4088 \endpgfpicture
4089 }
4090 }
```

```

4091 \cs_new_protected:Npn \@@_vdottedline_i:n #1
4092 {

```

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.

```

4093 \CT@arc@
4094 \pgfrememberpicturepositiononpagetrue
4095 \@@_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).

```

4096 \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
4097 \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
4098 \@@_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.

```

4099 \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
4100 \@@_qpoint:n { row - \@@_succ:n \c@iRow }
4101 \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`.

```

4102 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4103 \@@_draw_line:
4104 }

```

## 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.

```

4105 \bool_new:N \l_@@_block_auto_columns_width_bool

```

As of now, there is only one option available for the environment `{NiceMatrixBlock}`.

```

4106 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
4107 {
4108   auto-columns-width .code:n =
4109   {
4110     \bool_set_true:N \l_@@_block_auto_columns_width_bool
4111     \dim_gzero_new:N \g_@@_max_cell_width_dim
4112     \bool_set_true:N \l_@@_auto_columns_width_bool
4113   }
4114 }

4115 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
4116 {
4117   \int_gincr:N \g_@@_NiceMatrixBlock_int
4118   \dim_zero:N \l_@@_columns_width_dim
4119   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
4120   \bool_if:NT \l_@@_block_auto_columns_width_bool
4121   {
4122     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4123     {
4124       \exp_args:NNc \dim_set:Nn \l_@@_columns_width_dim
4125       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4126     }
4127   }
4128 }

```

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`).

```

4129 {
4130   \bool_if:NT \l_@@_block_auto_columns_width_bool
4131   {
4132     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
4133     \iow_shipout:Nx \@mainaux
4134     {
4135       \cs_gset:cpn
4136       { @@ _ max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }
4137       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
4138     }
4139     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
4140   }
4141 }

```

## The extra nodes

First, two variants of the functions `\dim_min:nn` and `\dim_max:nn`.

```

4142 \cs_generate_variant:Nn \dim_min:nn { v n }
4143 \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).

```

4144 \cs_new_protected:Npn \@@_create_extra_nodes:
4145 {
4146   \bool_if:nTF \l_@@_medium_nodes_bool
4147   {
4148     \bool_if:nTF \l_@@_large_nodes_bool
4149     \@@_create_medium_and_large_nodes:
4150     \@@_create_medium_nodes:
4151   }
4152   { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
4153 }

```

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`.

```

4154 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
4155 {

```

```

4156 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4157 {
4158   \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
4159   \dim_set_eq:cn { l_@@_row_\@@_i: _min_dim } \c_max_dim
4160   \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
4161   \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
4162 }
4163 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4164 {
4165   \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
4166   \dim_set_eq:cn { l_@@_column_\@@_j: _min_dim } \c_max_dim
4167   \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
4168   \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
4169 }

```

We begin the two nested loops over the rows and the columns of the array.

```

4170 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4171 {
4172   \int_step_variable:nnNn
4173   \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.

```

4174 {
4175   \cs_if_exist:cT
4176   { 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`.

```

4177 {
4178   \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
4179   \dim_set:cn { l_@@_row_\@@_i: _min_dim }
4180   { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
4181   \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4182   {
4183     \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
4184     { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
4185   }

```

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`.

```

4186   \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
4187   \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
4188   { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
4189   \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4190   {
4191     \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
4192     { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
4193   }
4194 }
4195 }
4196 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

4197 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4198 {
4199   \dim_compare:nnNt
4200   { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
4201   {
4202     \@@_qpoint:n { row - \@@_i: - base }
4203     \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
4204     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
4205   }
4206 }

```

```

4207 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4208 {
4209   \dim_compare:nNnT
4210     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
4211     {
4212       \@@_qpoint:n { col - \@@_j: }
4213       \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
4214       \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
4215     }
4216   }
4217 }

```

Here is the command `\@@_create_medium_nodes:`. When this command is used, the “medium nodes” are created.

```

4218 \cs_new_protected:Npn \@@_create_medium_nodes:
4219 {
4220   \pgfpicture
4221     \pgfrememberpicturepositiononpagetrue
4222     \pgf@relevantforpicturesizefalse
4223     \@@_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”.

```

4224   \tl_set:Nn \l_@@_suffix_tl { -medium }
4225   \@@_create_nodes:
4226   \endpgfpicture
4227 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones<sup>49</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:`.

```

4228 \cs_new_protected:Npn \@@_create_large_nodes:
4229 {
4230   \pgfpicture
4231     \pgfrememberpicturepositiononpagetrue
4232     \pgf@relevantforpicturesizefalse
4233     \@@_computations_for_medium_nodes:
4234     \@@_computations_for_large_nodes:
4235     \tl_set:Nn \l_@@_suffix_tl { -large }
4236     \@@_create_nodes:
4237   \endpgfpicture
4238 }
4239 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
4240 {
4241   \pgfpicture
4242     \pgfrememberpicturepositiononpagetrue
4243     \pgf@relevantforpicturesizefalse
4244     \@@_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”.

```

4245   \tl_set:Nn \l_@@_suffix_tl { -medium }
4246   \@@_create_nodes:
4247   \@@_computations_for_large_nodes:
4248   \tl_set:Nn \l_@@_suffix_tl { -large }
4249   \@@_create_nodes:
4250   \endpgfpicture
4251 }

```

---

<sup>49</sup>If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

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.

```

4252 \cs_new_protected:Npn \@@_computations_for_large_nodes:
4253 {
4254   \int_set:Nn \l_@@_first_row_int 1
4255   \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`.

```

4256   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
4257   {
4258     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
4259     {
4260       (
4261         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
4262         \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4263       )
4264       / 2
4265     }
4266     \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4267     { l_@@_row _ \@@_i: _ min _ dim }
4268   }
4269   \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
4270   {
4271     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
4272     {
4273       (
4274         \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
4275         \dim_use:c
4276         { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4277       )
4278       / 2
4279     }
4280     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4281     { l_@@_column _ \@@_j: _ max _ dim }
4282   }

```

Here, we have to use `\dim_sub:cn` because of the number 1 in the name.

```

4283   \dim_sub:cn
4284   { l_@@_column _ 1 _ min _ dim }
4285   \l_@@_left_margin_dim
4286   \dim_add:cn
4287   { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
4288   \l_@@_right_margin_dim
4289 }

```

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).

```

4290 \cs_new_protected:Npn \@@_create_nodes:
4291 {
4292   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4293   {
4294     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4295     {

```

We draw the rectangular node for the cell (`\@@_i-\@@_j`).

```

4296       \@@_pgf_rect_node:nnnnn
4297       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4298       { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }

```



```

4299         { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
4300         { \dim_use:c { l_@@_column_ \@@_j: _max_dim } }
4301         { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
4302     \str_if_empty:NF \l_@@_name_str
4303     {
4304         \pgfnodealias
4305         { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4306         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4307     }
4308 }
4309 }

```

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  $n$ .

```

4310 \seq_mapthread_function:NNN
4311   \g_@@_multicolumn_cells_seq
4312   \g_@@_multicolumn_sizes_seq
4313   \@@_node_for_multicolumn:nn
4314 }

4315 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
4316 {
4317   \cs_set_nopar:Npn \@@_i: { #1 }
4318   \cs_set_nopar:Npn \@@_j: { #2 }
4319 }

```

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”).

```

4320 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
4321 {
4322   \@@_extract_coords_values: #1 \q_stop
4323   \@@_pgf_rect_node:nnnnn
4324     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4325     { \dim_use:c { l_@@_column_ \@@_j: _min _ dim } }
4326     { \dim_use:c { l_@@_row_ \@@_i: _min _ dim } }
4327     { \dim_use:c { l_@@_column_ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
4328     { \dim_use:c { l_@@_row_ \@@_i: _max _ dim } }
4329   \str_if_empty:NF \l_@@_name_str
4330   {
4331     \pgfnodealias
4332     { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4333     { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
4334   }
4335 }

```

## 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.

```

4336 \keys_define:nn { NiceMatrix / Block / FirstPass }
4337 {
4338   l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4339   l .value_forbidden:n = true ,
4340   r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4341   r .value_forbidden:n = true ,
4342   c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4343   c .value_forbidden:n = true ,

```

The two following lines will be uncommented when we will give to the key `color` its new definition.

```
4344 % color .tl_set:N = \l_@@_color_tl ,
4345 % color .value_required:n = true ,
4346 }
```

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.

```
4347 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
4348 {
```

If the first mandatory argument of the command (which is the size of the block with the syntax  $i-j$ ) has not been provided by the user, you use `1-1` (that is to say a block of only one cell).

```
4349 \tl_if_blank:nTF { #2 } { \@@_Block_i 1-1 \q_stop } { \@@_Block_i #2 \q_stop }
4350 { #1 } { #3 } { #4 }
4351 }
```

With the following construction, we extract the values of  $i$  and  $j$  in the first mandatory argument of the command.

```
4352 \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.

```
4353 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
4354 {
4355   \int_compare:nNnTF { #2 } = 1
4356   {
4357     \tl_if_empty:NTF \l_@@_cell_type_tl
4358     { \tl_set:Nn \l_@@_pos_of_block_tl c }
4359     { \tl_set_eq:NN \l_@@_pos_of_block_tl \l_@@_cell_type_tl }
4360   }
4361   { \tl_set:Nn \l_@@_pos_of_block_tl c }

4362   \keys_set:known:n { NiceMatrix / Block / FirstPass } { #3 }
```

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*).

```
4363 \tl_if_empty:NTF { #1 }
4364 { \int_set:Nn \l_tmpa_int { 100 } }
4365 { \int_set:Nn \l_tmpa_int { #1 } }
4366 \tl_if_empty:NTF { #2 }
4367 { \int_set:Nn \l_tmpb_int { 100 } }
4368 { \int_set:Nn \l_tmpb_int { #2 } }

4369 \tl_set:Nx \l_tmpa_tl
4370 {
4371   { \int_use:N \c@iRow }
4372   { \int_use:N \c@jCol }
4373   { \int_eval:n { \c@iRow + \l_tmpa_int - 1 } }
4374   { \int_eval:n { \c@jCol + \l_tmpb_int - 1 } }
4375 }
```

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_iii:nnnnn` and `\@@_Block_iv:nnnnn` (the five arguments of those macros are provided by currying).

```

4376 \bool_lazy_or:nnTF
4377 { \int_compare_p:nNn { \l_tmpa_int } = 1 }
4378 { \int_compare_p:nNn { \l_tmpb_int } = 1 }
4379 { \exp_args:Nxx \@@_Block_iii:nnnnn }
4380 { \exp_args:Nxx \@@_Block_iv:nnnnn }
4381 { \l_tmpa_int } { \l_tmpb_int } { #3 } { #4 } { #5 }
4382 }

```

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).

```

4383 \cs_new_protected:Npn \@@_Block_iii:nnnnn #1 #2 #3 #4 #5
4384 {
4385   \int_gincr:N \g_@@_block_box_int
4386   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4387   {
4388     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4389     {
4390       \@@_actually_diagbox:nnnnnn
4391       { \int_use:N \c@iRow }
4392       { \int_use:N \c@jCol }
4393       { \int_eval:n { \c@iRow + #1 - 1 } }
4394       { \int_eval:n { \c@jCol + #2 - 1 } }
4395       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4396     }
4397   }
4398   \box_gclear_new:c
4399   { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4400   \hbox_gset:cn
4401   { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4402   {

```

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.

```

4403   \tl_if_empty:NTF \l_@@_color_tl
4404   { \int_compare:nNnT { #2 } = 1 \set@color }
4405   { \color { \l_@@_color_tl } }
4406   \group_begin:
4407   \cs_set:Npn \arraystretch { 1 }
4408   \dim_set_eq:NN \extrarowheight \c_zero_dim
4409   #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).

```

4410   \bool_if:NT \g_@@_rotate_bool { \tl_set:Nn \l_@@_pos_of_block_tl c }
4411   \bool_if:NTF \l_@@_NiceTabular_bool
4412   {
4413     \exp_args:Nxx \begin { tabular }
4414     { @ { } \l_@@_pos_of_block_tl @ { } }
4415     #5
4416     \end { tabular }
4417   }
4418   {
4419     \c_math_toggle_token
4420     \exp_args:Nxx \begin { array }
4421     { @ { } \l_@@_pos_of_block_tl @ { } }

```

```

4422         #5
4423         \end { array }
4424         \c_math_toggle_token
4425     }
4426     \group_end:
4427 }
4428 \bool_if:NT \g_@@_rotate_bool
4429 {
4430     \box_grotate:cn
4431     { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4432     { 90 }
4433     \bool_gset_false:N \g_@@_rotate_bool
4434 }

```

If we are in a mono-column block, we take into account the width of that block for the width of the column.

```

4435 \int_compare:nNnT { #2 } = 1
4436 {
4437     \dim_gset:Nn \g_@@_blocks_wd_dim
4438     {
4439         \dim_max:nn
4440         \g_@@_blocks_wd_dim
4441         {
4442             \box_wd:c
4443             { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4444         }
4445     }
4446 }

```

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.

```

4447 \int_compare:nNnT { #1 } = 1
4448 {
4449     \dim_gset:Nn \g_@@_blocks_ht_dim
4450     {
4451         \dim_max:nn
4452         \g_@@_blocks_ht_dim
4453         {
4454             \box_ht:c
4455             { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4456         }
4457     }
4458     \dim_gset:Nn \g_@@_blocks_dp_dim
4459     {
4460         \dim_max:nn
4461         \g_@@_blocks_dp_dim
4462         {
4463             \box_dp:c
4464             { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4465         }
4466     }
4467 }
4468 \seq_gput_right:Nx \g_@@_blocks_seq
4469 {
4470     \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.

```

4471 { \exp_not:n { #3 } , \l_@@_pos_of_block_tl }
4472 {
4473     \box_use_drop:c
4474     { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }

```

```

4475     }
4476   }
4477 }

```

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.

```

4478 \cs_new_protected:Npn \@@_Block_iv:nnnnn #1 #2 #3 #4 #5
4479 {
4480   \seq_gput_right:Nx \g_@@_blocks_seq
4481   {
4482     \l_tmpa_tl
4483     { \exp_not:n { #3 } }
4484     \exp_not:n
4485     {
4486       {
4487         \bool_if:NTF \l_@@_NiceTabular_bool
4488         {
4489           \group_begin:
4490           \cs_set:Npn \arraystretch { 1 }
4491           \dim_set_eq:NN \extrarowheight \c_zero_dim
4492           #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).

```

4493       \bool_if:NT \g_@@_rotate_bool
4494       { \tl_set:Nn \l_@@_pos_of_block_tl c }
4495       \exp_args:Nnx \begin { tabular }
4496       { @ { } \l_@@_pos_of_block_tl @ { } }
4497       #5
4498       \end { tabular }
4499       \group_end:
4500     }
4501   {
4502     \group_begin:
4503     \cs_set:Npn \arraystretch { 1 }
4504     \dim_set_eq:NN \extrarowheight \c_zero_dim
4505     #4
4506     \bool_if:NT \g_@@_rotate_bool
4507     { \tl_set:Nn \l_@@_pos_of_block_tl c }
4508     \c_math_toggle_token
4509     \exp_args:Nnx \begin { array }
4510     { @ { } \l_@@_pos_of_block_tl @ { } } #5 \end { array }
4511     \c_math_toggle_token
4512     \group_end:
4513   }
4514 }
4515 }
4516 }
4517 }

```

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).

```

4518 \keys_define:nn { NiceMatrix / Block / SecondPass }
4519 {
4520   fill .tl_set:N = \l_@@_fill_tl ,
4521   fill .value_required:n = true ,

```

```

4522 draw .tl_set:N = \l_@@_draw_tl ,
4523 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.

```

4524 color .code:n =
4525   \@@_error:n { Key-color~for~Block }
4526   \tl_set:Nn \l_@@_fill_tl { #1 } ,
4527   % color .code:n = \color { #1 } \tl_set:Nn \l_@@_draw_tl { #1 } ,
4528   color .value_required:n = true ,
4529   line-width .dim_set:N = \l_@@_line_width_dim ,
4530   line-width .value_required:n = true ,
4531   l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4532   l .value_forbidden:n = true ,
4533   r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4534   r .value_forbidden:n = true ,
4535   c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4536   c .value_forbidden:n = true ,
4537   unknown .code:n = \@@_error:n { Unknown~key~for~Block }
4538 }

```

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.

```

4539 \cs_new_protected:Npn \@@_draw_blocks:
4540 {
4541   \cs_set_eq:NN \ialign \@@_old_ialign:
4542   \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iii:nnnnnn ##1 }
4543 }
4544 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
4545 {
4546   % The integer |\l_@@_last_row_int| will be the last row of the block and
4547   % |\l_@@_last_col_int| its last column.
4548   % \begin{macrocode}
4549   \int_zero_new:N \l_@@_last_row_int
4550   \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.

```

4551   \int_compare:nNnTF { #3 } > { 99 }
4552     { \int_set_eq:NN \l_@@_last_row_int \c@iRow }
4553     { \int_set:Nn \l_@@_last_row_int { #3 } }
4554   \int_compare:nNnTF { #4 } > { 99 }
4555     { \int_set_eq:NN \l_@@_last_col_int \c@jCol }
4556     { \int_set:Nn \l_@@_last_col_int { #4 } }
4557   \bool_lazy_or:nnTF
4558     { \int_compare_p:nNn \l_@@_last_row_int > \g_@@_row_total_int }
4559     { \int_compare_p:nNn \l_@@_last_col_int > \g_@@_col_total_int }
4560     {
4561       \int_compare:nTF
4562         { \l_@@_last_col_int <= \g_@@_static_num_of_col_int }
4563         {
4564           \msg_error:nnnn { nicematrix } { Block~too~large~2 } { #1 } { #2 }
4565           \@@_msg_redirect_name:nn { Block~too~large~2 } { none }
4566           \group_begin:
4567           \globaldefs = 1
4568           \@@_msg_redirect_name:nn { columns~not~used } { none }
4569           \group_end:

```

```

4570     }
4571     { \msg_error:nnnn { nicematrix } { Block~too~large~1 } { #1 } { #2 } }
4572   }
4573   { \@@_Block_iv:nnnnnn { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } }
4574 }
4575 \cs_new_protected:Npn \@@_Block_iv:nnnnnn #1 #2 #3 #4 #5 #6
4576 {

```

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).

```

4577   \seq_gput_left:Nn \g_@@_pos_of_blocks_seq { { #1 } { #2 } { #3 } { #4 } }

```

The group is for the keys.

```

4578   \group_begin:
4579   \keys_set:nn { NiceMatrix / Block / SecondPass } { #5 }

4580   \bool_lazy_or:nnT
4581   { ! \tl_if_empty_p:N \l_@@_draw_tl }
4582   { \dim_compare_p:nNn \l_@@_line_width_dim > \c_zero_dim }
4583   {
4584     \tl_gput_right:Nx \g_nicematrix_code_after_tl
4585     {
4586       \@@_stroke_block:nnn
4587       { \exp_not:n { #5 } }
4588       { #1 - #2 }
4589       { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
4590     }
4591     \seq_gput_right:Nn \g_@@_pos_of_stroken_blocks_seq
4592     { { #1 } { #2 } { #3 } { #4 } }
4593   }

4594   \tl_if_empty:NF \l_@@_fill_tl
4595   {
4596     \tl_gput_right:Nx \g_nicematrix_code_before_tl
4597     {
4598       \exp_not:N \rectanglecolor
4599       { \exp_not:V \l_@@_fill_tl }
4600       { #1 - #2 }
4601       { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
4602     }
4603   }

4604   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4605   {
4606     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4607     {
4608       \@@_actually_diagbox:nnnnnn
4609       { #1 }
4610       { #2 }
4611       { \int_use:N \l_@@_last_row_int }
4612       { \int_use:N \l_@@_last_col_int }
4613       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4614     }
4615   }

4616   \hbox_set:Nn \l_@@_cell_box { #6 }
4617   \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.

```

4618 \pgfpicture
4619 \pgfrememberpicturepositiononpagetrue
4620 \pgf@relevantforpicturesizefalse
4621 \@@_qpoint:n { row - #1 }
4622 \dim_set_eq:NN \l_tmpa_dim \pgf@y
4623 \@@_qpoint:n { col - #2 }
4624 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4625 \@@_qpoint:n { row - \@@_succ:n { \l_@@_last_row_int } }
4626 \dim_set_eq:NN \l_tmpc_dim \pgf@y
4627 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
4628 \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.

```

4629 \begin { pgfscope }
4630 \@@_pgf_rect_node:nnnnn
4631 { \@@_env: - #1 - #2 - block }
4632 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
4633 \end { pgfscope }

```

We construct the short node.

```

4634 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
4635 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4636 {

```

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.

```

4637 \cs_if_exist:cT
4638 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
4639 {
4640 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4641 {
4642 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
4643 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
4644 }
4645 }
4646 }

```

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.

```

4647 \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
4648 {
4649 \@@_qpoint:n { col - #2 }
4650 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4651 }
4652 \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
4653 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4654 {

```



```

4655 \cs_if_exist:cT
4656 { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
4657 {
4658   \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4659   {
4660     \pgfpointanchor
4661     { \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
4662     { east }
4663     \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
4664   }
4665 }
4666 }
4667 \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
4668 {
4669   \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
4670   \dim_set_eq:NN \l_tmpd_dim \pgf@x
4671 }
4672 \@@_pgf_rect_node:nnnnn
4673 { \@@_env: - #1 - #2 - block - short }
4674 \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.

```

4675 \bool_if:NT \l_@@_medium_nodes_bool
4676 {
4677   \@@_pgf_rect_node:nnn
4678   { \@@_env: - #1 - #2 - block - medium }
4679   { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
4680   {
4681     \pgfpointanchor
4682     { \@@_env:
4683       - \int_use:N \l_@@_last_row_int
4684       - \int_use:N \l_@@_last_col_int - medium
4685     }
4686     { south-east }
4687   }
4688 }

```

Now, we will put the label of the block beginning with the case of a `\Block` of one row.

```

4689 \int_compare:nNnTF { #1 } = { #3 }
4690 {

```

We take into account the case of a block of one row in the “first row” or the “last row”.

```

4691 \int_compare:nNnTF { #1 } = 0
4692 { \l_@@_code_for_first_row_tl }
4693 {
4694   \int_compare:nNnT { #1 } = \l_@@_last_row_int
4695   \l_@@_code_for_last_row_tl
4696 }

```

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`.

```

4697 \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

We retrieve (in `\pgf@x`) the  $x$ -value of the center of the block.

```

4698 \pgfpointanchor
4699 { \@@_env: - #1 - #2 - block - short }
4700 {
4701   \str_case:Vn \l_@@_pos_of_block_tl
4702   {
4703     c { center }
4704     l { west }
4705     r { east }

```

```

4706     }
4707 }

```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```

4708 \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
4709 \pgfset { inner-sep = \c_zero_dim }
4710 \pgfnode
4711 { rectangle }
4712 {
4713     \str_case:Vn \l_@@_pos_of_block_tl
4714     {
4715         c { base }
4716         l { base~west }
4717         r { base~east }
4718     }
4719 }
4720 { \box_use_drop:N \l_@@_cell_box } { } { }
4721 }

```

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`).

```

4722 {

```

If we are in the first column, we must put the block as if it was with the key `r`.

```

4723 \int_compare:nNnT { #2 } = 0
4724 { \tl_set:Nn \l_@@_pos_of_block_tl r }
4725 \bool_if:nT \g_@@_last_col_found_bool
4726 {
4727     \int_compare:nNnT { #2 } = \g_@@_col_total_int
4728     { \tl_set:Nn \l_@@_pos_of_block_tl l }
4729 }
4730 \pgftransformshift
4731 {
4732     \pgfpointanchor
4733     { \@@_env: - #1 - #2 - block - short }
4734     {
4735         \str_case:Vn \l_@@_pos_of_block_tl
4736         {
4737             c { center }
4738             l { west }
4739             r { east }
4740         }
4741     }
4742 }
4743 \pgfset { inner-sep = \c_zero_dim }
4744 \pgfnode
4745 { rectangle }
4746 {
4747     \str_case:Vn \l_@@_pos_of_block_tl
4748     {
4749         c { center }
4750         l { west }
4751         r { east }
4752     }
4753 }
4754 { \box_use_drop:N \l_@@_cell_box } { } { }
4755 }
4756 \endpgfpicture
4757 \group_end:
4758 }

```

The first argument of `\@@_stroke_block:nnn` is a list of options for the rectangle that you will stoke. 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).

```

4759 \cs_new_protected:Npn \@@_stroke_block:nnn #1 #2 #3
4760 {
4761   \group_begin:
4762   \tl_clear:N \l_@@_draw_tl
4763   \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
4764   \keys_set_known:nn { NiceMatrix / BlockStroke } { #1 }
4765   \pgfpicture
4766   \pgfrememberpicturerepositiononpagetrue
4767   \pgf@relevantforpicturesizefalse
4768   \tl_if_empty:NF \l_@@_draw_tl
4769   {

```

If the user has used the key `color` of the command `\Block` without value, the color fixed by `\arrayrulecolor` is used.

```

4770     \str_if_eq:VnTF \l_@@_draw_tl { default }
4771     { \CT@arc@ }
4772     { \pgfsetstrokecolor { \l_@@_draw_tl } }
4773   }
4774   \@@_cut_on_hyphen:w #2 \q_stop
4775   \bool_lazy_and:nnT
4776   { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
4777   { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
4778   {
4779     \@@_qpoint:n { row - \l_tmpa_tl }
4780     \dim_set:Nn \l_tmpb_dim { \pgf@y }
4781     \@@_qpoint:n { col - \l_tmpb_tl }
4782     \dim_set:Nn \l_tmpc_dim { \pgf@x }
4783     \@@_cut_on_hyphen:w #3 \q_stop
4784     \int_compare:nNnT \l_tmpa_tl > \c@iRow
4785     { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
4786     \int_compare:nNnT \l_tmpb_tl > \c@jCol
4787     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
4788     \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
4789     \dim_set:Nn \l_tmpa_dim { \pgf@y }
4790     \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
4791     \dim_set:Nn \l_tmpd_dim { \pgf@x }
4792     \pgfpathrectanglecorners
4793     { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4794     { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
4795     \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }
4796     \pgfusepathqstroke
4797   }
4798   \endpgfpicture
4799   \group_end:
4800 }

```

Here is the set of keys for the command `\@@_stroke_block:nnn`.

```

4801 \keys_define:nn { NiceMatrix / BlockStroke }
4802 {

```

We will uncomment the following line when we will give to the key `color` of the command `\Block` its new definition.

```

4803   % color .tl_set:N = \l_@@_draw_tl ,
4804   draw .tl_set:N = \l_@@_draw_tl ,
4805   draw .default:n = default ,
4806   line-width .dim_set:N = \l_@@_line_width_dim ,
4807 }

```

## How to draw the dotted lines transparently

```

4808 \cs_set_protected:Npn \@@_renew_matrix:
4809 {

```

```

4810 \RenewDocumentEnvironment { pmatrix } { }
4811 { \pNiceMatrix }
4812 { \endpNiceMatrix }
4813 \RenewDocumentEnvironment { vmatrix } { }
4814 { \vNiceMatrix }
4815 { \endvNiceMatrix }
4816 \RenewDocumentEnvironment { Vmatrix } { }
4817 { \VNiceMatrix }
4818 { \endVNiceMatrix }
4819 \RenewDocumentEnvironment { bmatrix } { }
4820 { \bNiceMatrix }
4821 { \endbNiceMatrix }
4822 \RenewDocumentEnvironment { Bmatrix } { }
4823 { \BNiceMatrix }
4824 { \endBNiceMatrix }
4825 }

```

## Automatic arrays

```

4826 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
4827 {
4828   \int_set:Nn \l_@@_nb_rows_int { #1 }
4829   \int_set:Nn \l_@@_nb_cols_int { #2 }
4830 }
4831 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
4832 {
4833   \int_zero_new:N \l_@@_nb_rows_int
4834   \int_zero_new:N \l_@@_nb_cols_int
4835   \@@_set_size:n #4 \q_stop
4836   \begin { NiceArrayWithDelims } { #1 } { #2 }
4837     { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
4838   \int_compare:nNnT \l_@@_first_row_int = 0
4839     {
4840       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4841       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4842       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4843     }
4844   \prg_replicate:nn \l_@@_nb_rows_int
4845     {
4846       \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`).

```

4847   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
4848   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4849 }
4850 \int_compare:nNnT \l_@@_last_row_int > { -2 }
4851 {
4852   \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4853   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4854   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4855 }
4856 \end { NiceArrayWithDelims }
4857 }
4858 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
4859 {
4860   \cs_set_protected:cpn { #1 AutoNiceMatrix }
4861   {
4862     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
4863     \AutoNiceMatrixWithDelims { #2 } { #3 }
4864   }
4865 }

```

```

4866 \@@_define_com:nnn p ( )
4867 \@@_define_com:nnn b [ ]
4868 \@@_define_com:nnn v | |
4869 \@@_define_com:nnn V \| \|
4870 \@@_define_com:nnn B \{ \}

```

We define also an command `\AutoNiceMatrix` similar to the environment `{NiceMatrix}`.

```

4871 \NewDocumentCommand \AutoNiceMatrix { O { } m O { } m ! O { } }
4872 {
4873   \group_begin:
4874     \bool_set_true:N \l_@@_NiceArray_bool
4875     \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
4876   \group_end:
4877 }

```

## The redefinition of the command `\dotfill`

```

4878 \cs_set_eq:NN \@@_old_dotfill \dotfill
4879 \cs_new_protected:Npn \@@_dotfill:
4880 {

```

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}`).

```

4881   \@@_old_dotfill
4882   \bool_if:NT \l_@@_NiceTabular_bool
4883     { \group_insert_after:N \@@_dotfill_ii: }
4884     { \group_insert_after:N \@@_dotfill_i: }
4885 }
4886 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
4887 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }

```

Now, if the box is not empty (unfortunately, we can’t actually test whether the box is empty and that’s why we only consider it’s width), we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in the cell of the array, and it will extend, since it is no longer in `\l_@@_cell_box`.

```

4888 \cs_new_protected:Npn \@@_dotfill_iii:
4889 { \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`.

```

4890 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
4891 {
4892   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4893   {
4894     \@@_actually_diagbox:nnnnnn
4895     { \int_use:N \c@iRow }
4896     { \int_use:N \c@jCol }
4897     { \int_use:N \c@iRow }
4898     { \int_use:N \c@jCol }
4899     { \exp_not:n { #1 } }
4900     { \exp_not:n { #2 } }
4901   }

```

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`.

```

4902   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
4903   {
4904     { \int_use:N \c@iRow }
4905     { \int_use:N \c@jCol }
4906     { \int_use:N \c@iRow }
4907     { \int_use:N \c@jCol }
4908   }
4909 }

```

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.

```

4910 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
4911 {
4912   \pgfpicture
4913   \pgf@relevantforpicturesizefalse
4914   \pgfrememberpicturepositiononpagetrue
4915   \@@_qpoint:n { row - #1 }
4916   \dim_set_eq:NN \l_tmpa_dim \pgf@y
4917   \@@_qpoint:n { col - #2 }
4918   \dim_set_eq:NN \l_tmpb_dim \pgf@x
4919   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4920   \@@_qpoint:n { row - \@@_succ:n { #3 } }
4921   \dim_set_eq:NN \l_tmpc_dim \pgf@y
4922   \@@_qpoint:n { col - \@@_succ:n { #4 } }
4923   \dim_set_eq:NN \l_tmpd_dim \pgf@x
4924   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
4925   {

```

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.

```

4926   \CT@arc@
4927   \pgfsetroundcap
4928   \pgfusepathqstroke
4929 }
4930 \pgfset { inner~sep = 1 pt }
4931 \pgfscope
4932 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4933 \pgfnode { rectangle } { south-west }
4934 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
4935 \endpgfscope
4936 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
4937 \pgfnode { rectangle } { north-east }
4938 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
4939 \endpgfpicture
4940 }

```

## The keyword `\CodeAfter`

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. 90.

In the environments of `nicematrix`, `\CodeAfter` will be linked to the following command `\@@_CodeAfter:`. That macro must *not* be protected since it begins with `\omit`.

```

4941 \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`.

```

4942 \cs_new_protected:Npn \@@_CodeAfter_i:n #1 \end
4943 {
4944   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
4945   \@@_CodeAfter_ii:n
4946 }

```

We catch the argument of the command `\end` (in `#1`).

```
4947 \cs_new_protected:Npn \@@_CodeAfter_ii:n #1
4948 {
```

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.

```
4949 \str_if_eq:eeTF \currentenvir { #1 }
4950 { \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`.

```
4951 {
4952   \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
4953   \@@_CodeAfter_i:n
4954 }
4955 }
```

## 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.

```
4956 \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.

```
4957 \bool_new:N \c_@@_footnote_bool

4958 \@@_msg_new:nnn { Unknown~option~for~package }
4959 {
4960   The~key~'\l_keys_key_str'~is~unknown. \\
4961   If~you~go~on,~it~will~be~ignored. \\
4962   For~a~list~of~the~available~keys,~type~H~<return>.
4963 }
4964 {
4965   The~available~keys~are~(in~alphabetic~order):~
4966   define-L-C-R,~
4967   footnote,~
4968   footnotehyper,~
4969   renew-dots,~
4970   renew-matrix~and~
4971   transparent.
4972 }

4973 \keys_define:nn { NiceMatrix / Package }
4974 {
4975   define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
4976   define-L-C-R .default:n = true ,
4977   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
4978   renew-dots .value_forbidden:n = true ,
4979   renew-matrix .code:n = \@@_renew_matrix: ,
4980   renew-matrix .value_forbidden:n = true ,
4981   transparent .meta:n = { renew-dots , renew-matrix } ,
4982   transparent .value_forbidden:n = true,
4983   footnote .bool_set:N = \c_@@_footnote_bool ,
4984   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
4985   unknown .code:n = \@@_error:n { Unknown~option~for~package }
```

```

4986 }
4987 \ProcessKeysOptions { NiceMatrix / Package }

4988 \@@_msg_new:nn { footnote-with-footnotehyper-package }
4989 {
4990   You~can't~use~the~option~'footnote'~because~the~package~
4991   footnotehyper~has~already~been~loaded.~
4992   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
4993   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
4994   of~the~package~footnotehyper.\\
4995   If~you~go~on,~the~package~footnote~won't~be~loaded.
4996 }

4997 \@@_msg_new:nn { footnotehyper-with-footnote-package }
4998 {
4999   You~can't~use~the~option~'footnotehyper'~because~the~package~
5000   footnote~has~already~been~loaded.~
5001   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
5002   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
5003   of~the~package~footnote.\\
5004   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
5005 }

5006 \bool_if:NT \c_@@_footnote_bool
5007 {
5008   \@ifclassloaded { beamer }
5009   { \msg_info:nn { nicematrix } { Option~incompatible~with~Beamer } }
5010   {
5011     \@ifpackageloaded { footnotehyper }
5012     { \@@_error:n { footnote-with-footnotehyper-package } }
5013     { \usepackage { footnote } }
5014   }
5015 }

5016 \bool_if:NT \c_@@_footnotehyper_bool
5017 {
5018   \@ifclassloaded { beamer }
5019   { \@@_info:n { Option~incompatible~with~Beamer } }
5020   {
5021     \@ifpackageloaded { footnote }
5022     { \@@_error:n { footnotehyper-with-footnote-package } }
5023     { \usepackage { footnotehyper } }
5024   }
5025   \bool_set_true:N \c_@@_footnote_bool
5026 }

```

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

```

5027 \seq_new:N \c_@@_types_of_matrix_seq
5028 \seq_set_from_clist:Nn \c_@@_types_of_matrix_seq
5029 {
5030   NiceMatrix ,
5031   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
5032 }
5033 \seq_set_map_x:NNn \c_@@_types_of_matrix_seq \c_@@_types_of_matrix_seq
5034 { \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`.

```

5035 \cs_new_protected:Npn \@@_error_too_much_cols:
5036 {
5037   \seq_if_in:NVTf \c_@@_types_of_matrix_seq \g_@@_name_env_str
5038   {
5039     \int_compare:nNnTF \l_@@_last_col_int = { -2 }
5040     { \@@_fatal:n { too~much~cols~for~matrix } }
5041     {
5042       \bool_if:NF \l_@@_last_col_without_value_bool
5043       { \@@_fatal:n { too~much~cols~for~matrix~with~last~col } }
5044     }
5045   }
5046   { \@@_fatal:n { too~much~cols~for~array } }
5047 }

```

The following command must *not* be protected since it's used in an error message.

```

5048 \cs_new:Npn \@@_message_hdotsfor:
5049 {
5050   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
5051   { ~Maybe~your~use~of~\token_to_str:N \Hdotsfor\ is~incorrect.}
5052 }
5053 \@@_msg_new:nn { too~much~cols~for~matrix~with~last~col }
5054 {
5055   You~try~to~use~more~columns~than~allowed~by~your~
5056   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~
5057   columns~is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus~the~
5058   exterior~columns).~This~error~is~fatal.
5059 }
5060 \@@_msg_new:nn { too~much~cols~for~matrix }
5061 {
5062   You~try~to~use~more~columns~than~allowed~by~your~
5063   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall~that~the~maximal~
5064   number~of~columns~for~a~matrix~is~fixed~by~the~LaTeX~counter~
5065   'MaxMatrixCols'.~Its~actual~value~is~\int_use:N \c@MaxMatrixCols.~
5066   This~error~is~fatal.
5067 }

```

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`.

```

5068 \@@_msg_new:nn { too~much~cols~for~array }
5069 {
5070   You~try~to~use~more~columns~than~allowed~by~your~
5071   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~columns~is~
5072   \int_use:N \g_@@_static_num_of_col_int\
5073   ~~~~~(plus~the~potential~exterior~ones).~
5074   This~error~is~fatal.
5075 }
5076 \@@_msg_new:nn { last~col~not~used }
5077 {
5078   The~key~'last~col'~is~in~force~but~you~have~not~used~that~last~column~
5079   in~your~\@@_full_name_env:~.~However,~you~can~go~on.
5080 }
5081 \@@_msg_new:nn { columns~not~used }
5082 {
5083   The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
5084   \g_@@_static_num_of_col_int\ columns~but~you~use~only~\int_use:N \c@jCol.\
5085   You~can~go~on~but~the~columns~you~did~not~used~won't~be~created.
5086 }
5087 \@@_msg_new:nn { in~first~col }
5088 {
5089   You~can't~use~the~command~\#1 in~the~first~column~(number~0)~of~the~array.\

```

```

5090     If-you-go-on,~this-command-will-be-ignored.
5091 }

5092 \@_msg_new:nn { in~last~col }
5093 {
5094     You-can't-use-the-command~#1 in~the~last~column~(exterior)~of~the~array.\\
5095     If-you-go-on,~this-command-will-be-ignored.
5096 }

5097 \@_msg_new:nn { in~first~row }
5098 {
5099     You-can't-use-the-command~#1 in~the~first~row~(number~0)~of~the~array.\\
5100     If-you-go-on,~this-command-will-be-ignored.
5101 }

5102 \@_msg_new:nn { in~last~row }
5103 {
5104     You-can't-use-the-command~#1 in~the~last~row~(exterior)~of~the~array.\\
5105     If-you-go-on,~this-command-will-be-ignored.
5106 }

5107 \@_msg_new:nn { bad-option-for~line-style }
5108 {
5109     Since-you-haven't-loaded-Tikz,~the-only-value-you-can-give-to~'line-style'~
5110     is~'standard'.~If-you-go-on,~this-key-will-be-ignored.
5111 }

5112 \@_msg_new:nn { Unknown-option-for~xdots }
5113 {
5114     As-for-now-there-is-only-three-key-available-here:~'color',~'line-style'~
5115     and~'shorten'~(and-you-try-to-use~'\l_keys_key_str').~If-you-go-on,~
5116     this-key-will-be-ignored.
5117 }

5118 \@_msg_new:nn { Unknown-option-for~rowcolors }
5119 {
5120     As-for-now-there-is-only-two-keys-available-here:~'cols'~and~'respect-blocks'~
5121     (and-you-try-to-use~'\l_keys_key_str').~If-you-go-on,~
5122     this-key-will-be-ignored.
5123 }

5124 \@_msg_new:nn { Key~color~for~Block }
5125 {
5126     The-key~'color'~for~the-command~\token_to_str:N \Block\
5127     is-deprecated:~you-should-use~'fill'~instead.\\
5128     You-can-go-on-for~this-time-but-remember~that~that-key~
5129     will-be-deleted-in-a-future-version.
5130 }

5131 \@_msg_new:nn { ampersand~in~light-syntax }
5132 {
5133     You-can't-use-an-ampersand~(\token_to_str &)~to~separate~columns~because
5134     ~you-have-used~the~key~'light-syntax'.~This-error-is-fatal.
5135 }

5136 \@_msg_new:nn { double-backslash~in~light-syntax }
5137 {
5138     You-can't-use~\token_to_str:N \\~to~separate~rows~because~you~have~used~
5139     the~key~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
5140     (set~by~the~key~'end-of-row').~This-error-is-fatal.
5141 }

5142 \@_msg_new:nn { standard-cline~in~document }
5143 {
5144     The-key~'standard-cline'~is-available-only~in~the~preamble.\\
5145     If-you-go-on~this-command-will-be-ignored.
5146 }

5147 \@_msg_new:nn { old-column~type }
5148 {

```

```

5149 The~column~type~'#1'~is~no~longer~defined~in~'nicematrix'.~
5150 Since~version~5.0,~you~have~to~use~'l',~'c'~and~'r'~instead~of~'L',~
5151 'C'~and~'R'.~You~can~also~use~the~key~'define-L-C-R'.\\
5152 This~error~is~fatal.
5153 }
5154 \@@_msg_new:nn { bad-value-for-baseline }
5155 {
5156 The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
5157 valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
5158 \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\\
5159 If~you~go~on,~a~value~of~1~will~be~used.
5160 }
5161 \@@_msg_new:nn { empty-environment }
5162 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }
5163 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
5164 {
5165 Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code-after'~
5166 can't~be~executed~because~a~cell~doesn't~exist.\\
5167 If~you~go~on~this~command~will~be~ignored.
5168 }
5169 \@@_msg_new:nn { Hdotsfor~in~col-0 }
5170 {
5171 You~can't~use~\token_to_str:N \Hdotsfor\ in~an~exterior~column~of~
5172 the~array.~This~error~is~fatal.
5173 }
5174 \@@_msg_new:nn { bad-corner }
5175 {
5176 #1~is~an~incorrect~specification~for~a~corner~(in~the~keys~
5177 'except-corners'~and~'hvlines-except-corners').~The~available~
5178 values~are:~NW,~SW,~NE~and~SE.\\
5179 If~you~go~on,~this~specification~of~corner~will~be~ignored.
5180 }
5181 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
5182 {
5183 In~the~\@@_full_name_env:,~you~must~use~the~key~
5184 'last-col'~without~value.\\
5185 However,~you~can~go~on~for~this~time~
5186 (the~value~'\l_keys_value_tl'~will~be~ignored).
5187 }
5188 \@@_msg_new:nn { last-col-non-empty-for-NiceMatrixOptions }
5189 {
5190 In~\NiceMatrixoptions,~you~must~use~the~key~
5191 'last-col'~without~value.\\
5192 However,~you~can~go~on~for~this~time~
5193 (the~value~'\l_keys_value_tl'~will~be~ignored).
5194 }
5195 \@@_msg_new:nn { Block-too-large-1 }
5196 {
5197 You~try~to~draw~a~block~in~the~cell~#1-#2~of~your~matrix~but~the~matrix~is~
5198 too~small~for~that~block. \\
5199 }
5200 \@@_msg_new:nn { Block-too-large-2 }
5201 {
5202 The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
5203 \g_@@_static_num_of_col_int\
5204 columns~but~you~use~only~\int_use:N \c@jCol\ and~that's~why~a~block~
5205 specified~in~the~cell~#1-#2~can't~be~drawn.~You~should~add~some~ampersands~
5206 (&)~at~the~end~of~the~first~row~of~your~
5207 \@@_full_name_env:.\\
5208 If~you~go~on,~this~block~and~maybe~others~will~be~ignored.

```

```

5209     }
5210 \@@_msg_new:nn { unknown-column-type }
5211 {
5212     The-column-type~'#1'~in-your~\@@_full_name_env:\
5213     is-unknown. \\
5214     This-error-is-fatal.
5215 }
5216 \@@_msg_new:nn { tabularnote-forbidden }
5217 {
5218     You-can't-use-the-command~\token_to_str:N\tabularnote\
5219     ~in-a~\@@_full_name_env:~This-command-is-available-only-in-
5220     \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
5221     If-you-go-on,~this-command-will-be-ignored.
5222 }
5223 \@@_msg_new:nn { bottomule-without-booktabs }
5224 {
5225     You-can't-use-the-key~'tabular/bottomrule'~because~you~haven't~
5226     loaded~'booktabs'.\\
5227     If-you-go-on,~this-key-will-be-ignored.
5228 }
5229 \@@_msg_new:nn { enumitem-not-loaded }
5230 {
5231     You-can't-use-the-command~\token_to_str:N\tabularnote\
5232     ~because~you~haven't~loaded~'enumitem'.\\
5233     If-you-go-on,~this-command-will-be-ignored.
5234 }
5235 \@@_msg_new:nn { Wrong-last-row }
5236 {
5237     You-have-used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
5238     \@@_full_name_env:\ seems-to-have~\int_use:N \c@iRow \ rows.~
5239     If-you-go-on,~the-value-of~\int_use:N \c@iRow \ will-be-used-for-
5240     last-row.~You-can-avoid-this-problem-by-using~'last-row'~
5241     without-value~(more-compilations-might-be-necessary).
5242 }
5243 \@@_msg_new:nn { Yet-in-env }
5244 { Environments-of-nicematrix-can't-be-nested.\\ This-error-is-fatal. }
5245 \@@_msg_new:nn { Outside-math-mode }
5246 {
5247     The~\@@_full_name_env:\ can-be-used-only~in-math-mode~
5248     (and-not~in~\token_to_str:N \vcenter).\\
5249     This-error-is-fatal.
5250 }
5251 \@@_msg_new:nn { Bad-value-for-letter-for-dotted-lines }
5252 {
5253     The-value-of~key~'\l_keys_key_str'~must-be-of-length~1.\\
5254     If-you-go-on,~it-will-be-ignored.
5255 }
5256 \@@_msg_new:nnn { Unknown-key-for-Block }
5257 {
5258     The-key~'\l_keys_key_str'~is-unknown~for~the-command~\token_to_str:N
5259     \Block.\\ If-you-go-on,~it-will-be-ignored. \\
5260     For-a-list-of-the-available-keys,~type-H~<return>.
5261 }
5262 {
5263     The-available-keys-are~(in-alphabetic-order):~,~,~c,~draw,~fill,~l,~
5264     line-width~and~r.
5265 }
5266 \@@_msg_new:nnn { Unknown-key-for-notes }
5267 {

```

```

5268 The~key~'\l_keys_key_str'~is~unknown.\\
5269 If~you~go~on,~it~will~be~ignored. \\
5270 For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
5271 }
5272 {
5273 The~available~keys~are~(in~alphabetic~order):~
5274 bottomrule,~
5275 code~after,~
5276 code~before,~
5277 enumitem~keys,~
5278 enumitem~keys~para,~
5279 para,~
5280 label~in~list,~
5281 label~in~tabular~and~
5282 style.
5283 }
5284 \@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
5285 {
5286 The~key~'\l_keys_key_str'~is~unknown~for~the~command~
5287 \token_to_str:N \NiceMatrixOptions. \\
5288 If~you~go~on,~it~will~be~ignored. \\
5289 For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5290 }
5291 {
5292 The~available~keys~are~(in~alphabetic~order):~
5293 allow~duplicate~names,~
5294 cell~space~bottom~limit,~
5295 cell~space~top~limit,~
5296 code~for~first~col,~
5297 code~for~first~row,~
5298 code~for~last~col,~
5299 code~for~last~row,~
5300 create~extra~nodes,~
5301 create~medium~nodes,~
5302 create~large~nodes,~
5303 delimiters~color,~
5304 end~of~row,~
5305 first~col,~
5306 first~row,~
5307 hlines,~
5308 hvlines,~
5309 hvlines~except~corners,~
5310 last~col,~
5311 last~row,~
5312 left~margin,~
5313 letter~for~dotted~lines,~
5314 light~syntax,~
5315 notes~(several subkeys),~
5316 nullify~dots,~
5317 renew~dots,~
5318 renew~matrix,~
5319 right~margin,~
5320 small,~
5321 transparent,~
5322 vlines,~
5323 xdots/color,~
5324 xdots/shorten~and~
5325 xdots/line~style.
5326 }
5327 \@_msg_new:nnn { Unknown~option~for~NiceArray }
5328 {
5329 The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
5330 \{NiceArray\}. \\

```

```

5331     If~you~go~on,~it~will~be~ignored.  \
5332     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5333 }
5334 {
5335     The~available~keys~are~(in~alphabetic~order):~
5336     b,~
5337     baseline,~
5338     c,~
5339     cell-space-bottom-limit,~
5340     cell-space-top-limit,~
5341     code-after,~
5342     code-for-first-col,~
5343     code-for-first-row,~
5344     code-for-last-col,~
5345     code-for-last-row,~
5346     colortbl-like,~
5347     columns-width,~
5348     create-extra-nodes,~
5349     create-medium-nodes,~
5350     create-large-nodes,~
5351     delimiters-color,~
5352     extra-left-margin,~
5353     extra-right-margin,~
5354     first-col,~
5355     first-row,~
5356     hlines,~
5357     hvlines,~
5358     hvlines-except-corners,~
5359     last-col,~
5360     last-row,~
5361     left-margin,~
5362     light-syntax,~
5363     name,~
5364     notes/bottomrule,~
5365     notes/para,~
5366     nullify-dots,~
5367     renew-dots,~
5368     right-margin,~
5369     rules/color,~
5370     rules/width,~
5371     small,~
5372     t,~
5373     vlines,~
5374     xdots/color,~
5375     xdots/shorten~and~
5376     xdots/line-style.
5377 }

```

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).

```

5378 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
5379 {
5380     The~key~'\l_keys_key_str'~is~unknown~for~the~
5381     \@@_full_name_env:.  \
5382     If~you~go~on,~it~will~be~ignored.  \
5383     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5384 }
5385 {
5386     The~available~keys~are~(in~alphabetic~order):~
5387     b,~
5388     baseline,~
5389     c,~
5390     cell-space-bottom-limit,~
5391     cell-space-top-limit,~

```

```

5392 code-after,~
5393 code-for-first-col,~
5394 code-for-first-row,~
5395 code-for-last-col,~
5396 code-for-last-row,~
5397 colortbl-like,~
5398 columns-width,~
5399 create-extra-nodes,~
5400 create-medium-nodes,~
5401 create-large-nodes,~
5402 delimiters-color,~
5403 extra-left-margin,~
5404 extra-right-margin,~
5405 first-col,~
5406 first-row,~
5407 hlines,~
5408 hvlines,~
5409 hvlines-except-corners,~
5410 l,~
5411 last-col,~
5412 last-row,~
5413 left-margin,~
5414 light-syntax,~
5415 name,~
5416 nullify-dots,~
5417 r,~
5418 renew-dots,~
5419 right-margin,~
5420 rules/color,~
5421 rules/width,~
5422 small,~
5423 t,~
5424 vl原因,~
5425 xdots/color,~
5426 xdots/shorten-and~
5427 xdots/line-style.
5428 }
5429 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
5430 {
5431   The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
5432   \{NiceTabular\}. \\
5433   If~you~go~on,~it~will~be~ignored. \\
5434   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5435 }
5436 {
5437   The~available~keys~are~(in~alphabetic~order):~
5438   b,~
5439   baseline,~
5440   c,~
5441   cell-space-bottom-limit,~
5442   cell-space-top-limit,~
5443   code-after,~
5444   code-for-first-col,~
5445   code-for-first-row,~
5446   code-for-last-col,~
5447   code-for-last-row,~
5448   colortbl-like,~
5449   columns-width,~
5450   create-extra-nodes,~
5451   create-medium-nodes,~
5452   create-large-nodes,~
5453   extra-left-margin,~
5454   extra-right-margin,~

```

```

5455 first-col,~
5456 first-row,~
5457 hlines,~
5458 hvlines,~
5459 hvlines-except-corners,~
5460 last-col,~
5461 last-row,~
5462 left-margin,~
5463 light-syntax,~
5464 name,~
5465 notes/bottomrule,~
5466 notes/para,~
5467 nullify-dots,~
5468 renew-dots,~
5469 right-margin,~
5470 rules/color,~
5471 rules/width,~
5472 t,~
5473 vlines,~
5474 xdots/color,~
5475 xdots/shorten~and~
5476 xdots/line-style.
5477 }

5478 \@@_msg_new:nnn { Duplicate-name }
5479 {
5480   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
5481   the~same~environment~name~twice.~You~can~go~on,~but,~
5482   maybe,~you~will~have~incorrect~results~especially~
5483   if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
5484   message~again,~use~the~key~'allow-duplicate-names'~in~
5485   '\token_to_str:N \NiceMatrixOptions'.\\
5486   For~a~list~of~the~names~already~used,~type~H~<return>. \\
5487 }
5488 {
5489   The~names~already~defined~in~this~document~are:~
5490   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
5491 }

5492 \@@_msg_new:nn { Option-auto-for-columns-width }
5493 {
5494   You~can't~give~the~value~'auto'~to~the~key~'columns-width'~here.~
5495   If~you~go~on,~the~key~will~be~ignored.
5496 }

```

## 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>50</sup>, Tikz externalization is now deactivated in the environments of the package `nicematrix`.<sup>51</sup>

## Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\left( \begin{array}{ccc} & C_j & \\ 0 & \vdots & 0 \\ & a & \cdots \\ 0 & & 0 \end{array} \right) L_i$$

## Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See [www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)

<sup>50</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>51</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 2.1.4 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>52</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.

## Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

---

<sup>52</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 3.11 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.

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<b>@@ commands:</b>	
\@@_Block: .....	1077, 4347
\@@_Block_i .....	4349, 4352
\@@_Block_ii:nnnnn .....	4352, 4353
\@@_Block_iii:nnnnn .....	4379, 4383
\@@_Block_iii:nnnnnn .....	4542, 4544
\@@_Block_iv:nnnnn .....	4380, 4478
\@@_Block_iv:nnnnnn .....	4573, 4575
\@@_Cdots .....	1002, 1067, 2979
\g_@@_Cdots_lines_tl .....	1094, 2310
\@@_Cell: .....	194, 764, 1470, 1517, 1535, 2096, 3079, 3080, 3081, 3082, 3083, 3084, 3085, 3086, 3087, 3088, 3089, 3090
\@@_CodeAfter: .....	1081, 4941
\@@_CodeAfter_i:n ....	766, 4941, 4942, 4953
\@@_CodeAfter_ii:n .....	4945, 4947
\@@_Ddots .....	1004, 1069, 3009
\g_@@_Ddots_lines_tl .....	1097, 2308
\g_@@_HVdotsfor_lines_tl .....	1099, 2306, 3138, 3214, 5050
\@@_Hdotsfor: .....	1007, 1074, 3114
\@@_Hdotsfor:nnnn .....	3140, 3152
\@@_Hdotsfor_i .....	3123, 3129, 3136
\@@_Hline: .....	1072, 3857
\@@_Hline_i:n .....	3857, 3858, 3864
\@@_Hline_ii:nn .....	3861, 3864
\@@_Hline_iii:n .....	3862, 3865
\@@_Hspace: .....	1073, 3062
\@@_Iddots .....	1005, 1070, 3032
\g_@@_Iddots_lines_tl .....	1098, 2309
\@@_Ldots .....	1001, 1006, 1066, 2964
\g_@@_Ldots_lines_tl .....	1095, 2311
\l_@@_Matrix_bool ....	240, 1364, 1380, 2087
\l_@@_NiceArray_bool .....	237, 339, 1141, 1252, 1307, 1411, 1423, 2063, 3734, 3735, 3853, 3854, 4056, 4063, 4874
\g_@@_NiceMatrixBlock_int .....	228, 4117, 4122, 4125, 4136
\l_@@_NiceTabular_bool .....	150, 238, 772, 937, 1116, 1196, 1198, 1341, 1345, 1412, 1424, 2116, 2125, 4411, 4487, 4882
\@@_NotEmpty: .....	1083, 2110
\@@_OnlyMainNiceMatrix:n .....	1079, 3595
\@@_OnlyMainNiceMatrix_i:n .....	3598, 3605, 3608
\@@_Vdots .....	1003, 1068, 2994
\g_@@_Vdots_lines_tl .....	1096, 2307
\@@_Vdotsfor: .....	1075, 3212
\@@_Vdotsfor:nnnn .....	3216, 3227
\@@_W: .....	1384, 1453
\@@_actually_diagbox:nnnnnn .....	4390, 4608, 4894, 4910
\@@_actually_draw_Cdots: .....	2557, 2561
\@@_actually_draw_Ddots: .....	2683, 2687
\@@_actually_draw_Iddots: .....	2734, 2738
\@@_actually_draw_Ldots: ..	2515, 2519, 3203
\@@_actually_draw_Vdots: ..	2610, 2614, 3278
\@@_adapt_S_column: .....	164, 179, 1115
\@@_adjust_pos_of_blocks_seq: ..	2223, 2273
\@@_adjust_pos_of_blocks_seq_i:nnnn ..	2276, 2278
\@@_adjust_size_box: .....	843, 869, 1544, 1999, 2043
\@@_after_array: .....	1373, 2129
\g_@@_after_col_zero_bool .....	263, 978, 1976, 3120
\@@_analyze_end:Nn .....	1774, 1819
\l_@@_argspec_tl .....	2962, 2963, 2964, 2979, 2994, 3009, 3032, 3134, 3135, 3136, 3210, 3211, 3212, 3288, 3289, 3290
\@@_array: .....	932, 1775, 1802
\l_@@_auto_columns_width_bool .....	444, 563, 1886, 1890, 4112
\l_@@_baseline_tl .....	433, 434, 556, 557, 558, 559, 945, 1309, 1587, 1599, 1604, 1606, 1611, 1616, 1698, 1699, 1703, 1708, 1710, 1715
\@@_begin_of_NiceMatrix:nn ....	2085, 2106
\@@_begin_of_row: .....	769, 793, 1977
\l_@@_block_auto_columns_width_bool ..	1129, 1891, 4105, 4110, 4120, 4130
\g_@@_block_box_int .....	290, 1109, 4385, 4399, 4401, 4431, 4443, 4455, 4464, 4474
\g_@@_blocks_dp_dim .....	234, 851, 854, 855, 4458, 4461
\g_@@_blocks_ht_dim .....	233, 857, 860, 861, 4449, 4452

\g_@@_blocks_seq .....	278, 1131, 1637, 4468, 4480, 4542
\g_@@_blocks_wd_dim .....	232, 845, 848, 849, 4437, 4440
\c_@@_booktabs_loaded_bool	24, 30, 1017, 1669
\@@_cartesian_fill: .....	3344, 3396, 3409, 3434, 3514, 3525
\l_@@_cell_box	771, 817, 819, 825, 831, 834, 838, 847, 848, 853, 854, 859, 860, 870, 871, 872, 873, 875, 878, 882, 884, 902, 1019, 1207, 1209, 1534, 1545, 1978, 2002, 2005, 2007, 2023, 2046, 2050, 4616, 4720, 4754, 4889
\l_@@_cell_space_bottom_limit_dim	422, 490, 873
\l_@@_cell_space_top_limit_dim	421, 488, 871
\l_@@_cell_type_tl .....	230, 231, 1470, 1536, 4357, 4359
\@@_cellcolor .....	1190, 3436, 3566, 3567
\@@_cellcolor_tabular .....	1011, 3572
\g_@@_cells_seq .....	1813, 1814, 1815, 1817
\@@_chessboardcolors .....	1195, 3559
\@@_cline .....	129, 1065
\@@_cline_i:nn .....	130, 131, 143, 146
\@@_cline_i:w .....	131, 132
\l_@@_code_before_bool .....	267, 553, 580, 952, 1137, 1147, 1833, 1850, 1868, 1899, 1925, 1952, 2175, 2267
\l_@@_code_before_tl	266, 552, 1138, 1197
\l_@@_code_for_first_col_tl	502, 1989
\l_@@_code_for_first_row_tl	506, 781, 4692
\l_@@_code_for_last_col_tl	504, 2032
\l_@@_code_for_last_row_tl	508, 788, 4695
\g_@@_col_total_int	770, 1090, 1293, 1918, 1919, 1955, 1959, 1964, 1965, 2022, 2133, 2136, 2141, 2148, 2192, 2647, 3110, 3111, 3273, 4163, 4173, 4207, 4294, 4559, 4727
\l_@@_color_tl .....	285, 4344, 4403, 4405
\@@_colortbl_like: .....	1009, 1084
\l_@@_colortbl_like_bool	419, 579, 1084, 1399
\c_@@_colortbl_loaded_bool	82, 86, 1034
\l_@@_cols_tl .....	3346, 3395, 3408, 3433, 3453, 3461, 3462
\@@_columncolor .....	1194, 3400
\@@_columncolor_preamble .....	1013, 3586
\c_@@_columncolor_regex .....	203, 1402
\l_@@_columns_width_dim .....	229, 564, 688, 1887, 1893, 4118, 4124
\g_@@_com_or_env_str .....	251, 254
\@@_computations_for_large_nodes: ..	4234, 4247, 4252
\@@_computations_for_medium_nodes: ..	4154, 4223, 4233, 4244
\@@_compute_a_corner:nnnnnn .....	3928, 3930, 3932, 3934, 3939
\@@_compute_corners: .....	2222, 3920
\@@_construct_preamble:n .....	1265, 1377
\@@_create_col_nodes: .....	1778, 1806, 1825
\@@_create_extra_nodes: .....	1636, 4144
\@@_create_large_nodes: .....	4152, 4228
\@@_create_medium_and_large_nodes: ..	4149, 4239
\@@_create_medium_nodes: .....	4150, 4218
\@@_create_nodes: .....	4225, 4236, 4246, 4249, 4290
\@@_create_row_node: .....	948, 981, 1018
\@@_cut_on_hyphen:w .....	3339, 3350, 3351, 3369, 3370, 3426, 3431, 3482, 4774, 4783
\g_@@_ddots_int .....	2202, 2707, 2708
\@@_def_env:nnn .....	2069, 2080, 2081, 2082, 2083, 2084
\@@_define_L_C_R: .....	216, 1264
\c_@@_define_L_C_R_bool	215, 1264, 4975
\@@_define_com:nnn .....	4858, 4866, 4867, 4868, 4869, 4870
\l_@@_delimiters_color_tl .....	455, 673, 719, 736, 1332, 1333
\g_@@_delta_x_one_dim	2204, 2710, 2720
\g_@@_delta_x_two_dim	2206, 2761, 2771
\g_@@_delta_y_one_dim	2205, 2712, 2720
\g_@@_delta_y_two_dim	2207, 2763, 2771
\@@_diagbox:nn .....	1082, 4890
\@@_dotfill: .....	4879
\@@_dotfill_i: .....	4884, 4886
\@@_dotfill_ii: .....	4883, 4886, 4887
\@@_dotfill_iii: .....	4887, 4888
\@@_double_int_eval:n	3284, 3298, 3299
\g_@@_dp_ante_last_row_dim	796, 1050
\g_@@_dp_last_row_dim .....	796, 797, 1053, 1054, 1208, 1209, 1326
\g_@@_dp_row_zero_dim .....	816, 817, 1044, 1045, 1319, 1692, 1731
\@@_draw_Cdots:nnn .....	2543
\@@_draw_Ddots:nnn .....	2675
\@@_draw_Iddots:nnn .....	2726
\@@_draw_Ldots:nnn .....	2501
\@@_draw_Vdots:nnn .....	2595
\@@_draw_blocks: .....	1637, 4539
\@@_draw_dotted_lines: .....	2221, 2295
\@@_draw_dotted_lines_i: .....	2298, 2302
\l_@@_draw_first_bool	289, 3024, 3047, 3058
\@@_draw_hlines: .....	2234, 3850
\@@_draw_line: .....	2541, 2593, 2673, 2724, 2775, 2777, 3337, 4073, 4103
\@@_draw_line_ii:nn .....	3317, 3321
\@@_draw_line_iii:nn .....	3324, 3328
\@@_draw_non_standard_dotted_line: ..	2783, 2785
\@@_draw_non_standard_dotted_line:n	2788, 2791
\@@_draw_standard_dotted_line: ..	2782, 2811
\@@_draw_standard_dotted_line_i: ..	2876, 2880
\l_@@_draw_tl .....	284, 4522, 4527, 4581, 4762, 4768, 4770, 4772, 4803, 4804
\@@_draw_vlines: .....	2235, 3731
\g_@@_empty_cell_bool	274, 877, 886, 2012, 2058, 2977, 2992, 3007, 3030, 3053, 3064
\l_@@_empty_corner_cells_seq	2228, 3669, 3675, 3682, 3790, 3796, 3803, 3922, 3995
\@@_end_Cell: .....	196, 864, 1472, 1524, 1540, 2096, 3079, 3080, 3081, 3082, 3083, 3084, 3085, 3086, 3087, 3088, 3089, 3090
\l_@@_end_of_row_tl .....	452, 453, 496, 1798, 1799, 5139
\c_@@_endpgfortikzpicture_tl .....	39, 43, 2299, 3325, 4042
\c_@@_enumitem_loaded_bool .....	25, 33, 312, 607, 612, 623, 628

\@@_env: .....	223, 227, 802, 808, 903, 909, 957, 963, 969, 1161, 1162, 1168, 1169, 1176, 1177, 1187, 1834, 1837, 1839, 1855, 1861, 1864, 1873, 1879, 1882, 1904, 1910, 1913, 1930, 1936, 1942, 1955, 1959, 1965, 2246, 2376, 2444, 2483, 2494, 3166, 3184, 3241, 3259, 3310, 3312, 3331, 3334, 3950, 3969, 3987, 4176, 4178, 4186, 4297, 4306, 4324, 4631, 4638, 4642, 4656, 4661, 4673, 4678, 4679, 4682, 4699, 4733
\g_@@_env_int ..	222, 223, 225, 1128, 1135, 1139, 1149, 1153, 1156, 1165, 1166, 1173, 1174, 1217, 1220, 1235, 1238, 2140, 2161, 2179, 2182, 2195, 2263, 3469, 3472, 3497, 4333
\@@_error:n .....	12, 315, 340, 465, 475, 633, 676, 687, 696, 701, 721, 728, 738, 744, 749, 760, 762, 1288, 1298, 1367, 1621, 1674, 1720, 3456, 4525, 4537, 4985, 5012, 5022
\@@_error:nn ...	13, 571, 1455, 1456, 1457, 2967, 2970, 2982, 2985, 2997, 3000, 3013, 3014, 3019, 3020, 3036, 3037, 3042, 3043, 3936
\@@_error:nnn .....	14, 3315
\@@_error_too_much_cols: .....	1434, 5035
\@@_everycr: .....	974, 1039, 1042
\@@_everycr_i: .....	974, 975
\l_@@_except_corners_clist .....	440, 539, 543, 3635, 3757, 3923
\l_@@_exterior_arraycolsep_bool .....	435, 684, 1414, 1426
\l_@@_extra_left_margin_dim .....	450, 531, 1268, 2010
\l_@@_extra_right_margin_dim .....	451, 532, 1280, 2054, 2650
\@@_extract_coords_values: ....	4315, 4322
\@@_fatal:n .....	15, 245, 1119, 1783, 1787, 1789, 1822, 3125, 5040, 5043, 5046
\@@_fatal:nn .....	16, 1463
\l_@@_fill_tl ...	283, 4520, 4526, 4594, 4599
\l_@@_final_i_int .....	2211, 2323, 2328, 2331, 2356, 2364, 2368, 2377, 2385, 2465, 2495, 2535, 2632, 2699, 2750, 3157, 3185, 3253, 3263, 3265
\l_@@_final_j_int .....	2212, 2324, 2329, 2336, 2341, 2347, 2357, 2365, 2369, 2378, 2386, 2466, 2496, 2532, 2572, 2701, 2752, 3178, 3188, 3190, 3232, 3261
\l_@@_final_open_bool .....	2214, 2330, 2334, 2337, 2344, 2350, 2354, 2370, 2530, 2570, 2579, 2590, 2617, 2630, 2638, 2659, 2697, 2748, 2884, 2899, 2930, 2931, 3155, 3179, 3191, 3230, 3254, 3266, 3307, 4039, 4078
\@@_find_extremities_of_line:nnnn ...	2318, 2505, 2547, 2599, 2679, 2730
\l_@@_first_col_int .....	117, 130, 293, 294, 498, 742, 769, 1302, 1406, 1828, 1848, 2186, 3118, 3360, 3597, 4163, 4173, 4207, 4255, 4294, 4840, 4846, 4852
\l_@@_first_row_int .....	291, 292, 499, 746, 1088, 1317, 1618, 1689, 1717, 1728, 2184, 4156, 4170, 4197, 4254, 4292, 4635, 4653, 4838, 5157
\c_@@_footnote_bool .....	1105, 1375, 4957, 4983, 5006, 5025
\c_@@_footnotehyper_bool ..	4956, 4984, 5016
\@@_full_name_env: .....	252, 5056, 5063, 5071, 5079, 5083, 5162, 5183, 5202, 5207, 5212, 5219, 5238, 5247, 5381
\@@_hdottedline: .....	1071, 4024
\@@_hdottedline:n .....	4032, 4036
\@@_hdottedline_i: .....	4027, 4029
\@@_hdottedline_i:n .....	4041, 4045
\@@_hline:nn .....	3738, 3855, 3873
\@@_hline_i:nn .....	2232, 3741, 3744
\@@_hline_i_complete:nn .....	2232, 3848
\@@_hline_ii:nnnn ...	3766, 3777, 3810, 3849
\l_@@_hlines_bool	439, 510, 515, 545, 982, 2234
\g_@@_ht_last_row_dim .....	798, 1051, 1052, 1206, 1207, 1325
\g_@@_ht_row_one_dim ..	824, 825, 1048, 1049
\g_@@_ht_row_zero_dim .....	818, 819, 1046, 1047, 1320, 1691, 1730
\@@_i: .....	4156, 4158, 4159, 4160, 4161, 4170, 4176, 4178, 4179, 4180, 4181, 4186, 4187, 4188, 4189, 4197, 4200, 4202, 4203, 4204, 4256, 4258, 4261, 4262, 4266, 4267, 4292, 4297, 4299, 4301, 4305, 4306, 4317, 4324, 4326, 4328, 4332, 4333
\g_@@_iddots_int .....	2203, 2758, 2759
\l_@@_in_env_bool ....	236, 339, 1119, 1120
\c_@@_in_preamble_bool ..	21, 22, 23, 603, 619
\@@_info:n .....	5019
\l_@@_initial_i_int .....	2209, 2321, 2396, 2399, 2424, 2432, 2436, 2445, 2453, 2463, 2484, 2526, 2581, 2583, 2626, 2691, 2742, 3156, 3157, 3167, 3235, 3245, 3247
\l_@@_initial_j_int .....	2210, 2322, 2397, 2404, 2409, 2415, 2425, 2433, 2437, 2446, 2454, 2464, 2485, 2523, 2565, 2640, 2642, 2647, 2693, 2744, 3160, 3170, 3172, 3231, 3232, 3243
\l_@@_initial_open_bool .....	2213, 2398, 2402, 2405, 2412, 2418, 2422, 2438, 2521, 2563, 2578, 2588, 2617, 2624, 2636, 2689, 2740, 2882, 2929, 3154, 3161, 3173, 3229, 3236, 3248, 3306, 4038, 4077
\@@_insert_tabularnotes: .....	1641, 1644
\@@_instruction_of_type:nnn .....	913, 2972, 2987, 3002, 3024, 3047
\l_@@_inter_dots_dim .....	423, 424, 2218, 2887, 2894, 2905, 2913, 2920, 2925, 2937, 2945, 4068, 4071, 4099, 4101
\g_@@_internal_code_after_tl	259, 1506, 1561, 2236, 2237, 3872, 4031, 4388, 4606, 4892
\@@_intersect_our_row:nnnn .....	3548
\@@_intersect_our_row_p:nnnn .....	3501
\@@_j: .....	4163, 4165, 4166, 4167, 4168, 4173, 4176, 4178, 4181, 4183, 4184, 4186, 4189, 4191, 4192, 4207, 4210, 4212, 4213, 4214, 4269, 4271, 4274, 4276, 4280, 4281, 4294, 4297, 4298, 4300, 4305, 4306, 4318, 4324, 4325, 4327, 4332, 4333
\l_@@_l_dim .....	2860, 2861, 2874, 2875, 2887, 2893, 2904, 2912, 2920, 2925, 2937, 2938, 2945, 2946
\l_@@_large_nodes_bool	447, 522, 4148, 4152



<code>\g_@@_last_col_found_bool</code> ..	301, 1093, 1294, 1359, 1917, 1946, 2020, 2132, 2189, 4725	<code>\g_@@_multicolumn_sizes_seq</code>	1087, 3100, 4312
<code>\l_@@_last_col_int</code> ..	299, 300, 677, 712, 714, 729, 745, 761, 1159, 1231, 1237, 1244, 1297, 1418, 2092, 2094, 2133, 2136, 2188, 2604, 2645, 2969, 2984, 3020, 3043, 4547, 4550, 4555, 4556, 4559, 4562, 4589, 4601, 4612, 4627, 4656, 4661, 4669, 4684, 4842, 4848, 4854, 5039, 5057	<code>\g_@@_name_env_str</code> .....	250, 255, 256, 1113, 1114, 1821, 2064, 2065, 2073, 2074, 2103, 2114, 2122, 2269, 4862, 5037
<code>\l_@@_last_col_without_value_bool</code> ..	298, 711, 2134, 5042	<code>\l_@@_name_str</code> .....	445, 573, 804, 807, 905, 908, 965, 968, 1215, 1224, 1227, 1233, 1242, 1245, 1838, 1839, 1863, 1864, 1881, 1882, 1912, 1913, 1938, 1941, 1961, 1964, 2143, 2147, 2164, 2168, 4302, 4305, 4329, 4332
<code>\l_@@_last_empty_column_int</code> .....	3960, 3961, 3974, 3980, 3993	<code>\g_@@_names_seq</code> .....	235, 570, 572, 5490
<code>\l_@@_last_empty_row_int</code> .....	3942, 3943, 3956, 3977	<code>\l_@@_nb_cols_int</code> .....	4829, 4834, 4837, 4841, 4847, 4853
<code>\l_@@_last_row_int</code> .....	295, 296, 500, 786, 832, 986, 1157, 1202, 1212, 1219, 1226, 1282, 1286, 1289, 1301, 1323, 1800, 1801, 1985, 1986, 2029, 2030, 2155, 2510, 2552, 2999, 3014, 3037, 3603, 3611, 4546, 4549, 4552, 4553, 4558, 4589, 4601, 4611, 4625, 4683, 4694, 4850, 5237	<code>\l_@@_nb_rows_int</code> .....	4828, 4833, 4844
<code>\l_@@_last_row_without_value_bool</code> ..	297, 1214, 1284, 2153	<code>\@@_newcolumnntype</code> .....	992, 1383, 1384
<code>\l_@@_left_delim_dim</code> .....	1250, 1254, 1259, 1766, 2008	<code>\@@_node_for_multicolumn:nn</code> ....	4313, 4320
<code>\l_@@_left_delim_tl</code> .....	1107, 4067	<code>\@@_node_for_the_cell:</code>	883, 889, 2006, 2055
<code>\l_@@_left_margin_dim</code> .....	448, 525, 1267, 2009, 4057, 4285	<code>\@@_node_position:</code> ..	1168, 1170, 1176, 1178
<code>\l_@@_letter_for_dotted_lines_str</code> ..	695, 703, 704, 1461	<code>\g_@@_not_empty_cell_bool</code>	265, 881, 887, 2111
<code>\l_@@_light_syntax_bool</code> .....	432, 494, 1270, 1275, 2156	<code>\@@_not_in_exterior:nnnn</code> .....	3540
<code>\@@_light_syntax_i</code> .....	1791, 1794	<code>\@@_not_in_exterior_p:nnnn</code> .....	3474
<code>\@@_line</code> .....	2250, 3290	<code>\l_@@_notes_above_space_dim</code> .....	441, 442
<code>\@@_line_i:nn</code> .....	3297, 3304	<code>\l_@@_notes_bottomrule_bool</code> .....	591, 732, 755, 1667
<code>\l_@@_line_width_dim</code> .....	286, 4529, 4582, 4763, 4795, 4806	<code>\l_@@_notes_code_after_tl</code> .....	589, 1676
<code>\@@_line_with_light_syntax:n</code> ..	1805, 1809	<code>\l_@@_notes_code_before_tl</code> .....	587, 1648
<code>\@@_line_with_light_syntax_i:n</code> .....	1804, 1810, 1811	<code>\@@_notes_label_in_list:n</code>	308, 327, 335, 599
<code>\@@_math_toggle_token:</code> .....	149, 866, 1979, 1996, 2024, 2040, 4934, 4938	<code>\@@_notes_label_in_tabular:n</code> .	307, 348, 596
<code>\g_@@_max_cell_width_dim</code> .....	874, 875, 1130, 1892, 4111, 4137	<code>\l_@@_notes_para_bool</code> ..	585, 730, 753, 1652
<code>\l_@@_max_delimiter_width_bool</code> .....	456, 493, 1355	<code>\@@_notes_style:n</code> .....	306, 309, 327, 335, 351, 356, 593
<code>\c_@@_max_l_dim</code> .....	2874, 2879	<code>\l_@@_nullify_dots_bool</code> .....	443, 520, 2976, 2991, 3006, 3029, 3052
<code>\l_@@_medium_nodes_bool</code>	446, 521, 4146, 4675	<code>\l_@@_number_of_notes_int</code>	305, 342, 352, 362
<code>\@@_message_hdotsfor:</code>	5048, 5056, 5063, 5071	<code>\@@_old_CT@arc@</code> .....	1121, 2271
<code>\@@_msg_new:nn</code> .....	17, 4988, 4997, 5053, 5060, 5068, 5076, 5081, 5087, 5092, 5097, 5102, 5107, 5112, 5118, 5124, 5131, 5136, 5142, 5147, 5154, 5161, 5163, 5169, 5174, 5181, 5188, 5195, 5200, 5210, 5216, 5223, 5229, 5235, 5243, 5245, 5251, 5492	<code>\@@_old_arraycolsep_dim</code> .....	275
<code>\@@_msg_new:nnn</code> .....	18, 4958, 5256, 5266, 5284, 5327, 5378, 5429, 5478	<code>\@@_old_cdots</code> .....	1059, 2991
<code>\@@_msg_redirect_name:nn</code> .....	19, 690, 1371, 4565, 4568	<code>\@@_old_ddots</code> .....	1061, 3029
<code>\@@_multicolumn:nnn</code> .....	1076, 3068	<code>\@@_old_dotfill</code> .....	4878, 4881, 4889
<code>\g_@@_multicolumn_cells_seq</code> .....	1086, 3098, 4181, 4189, 4311, 4640, 4658	<code>\@@_old_dotfill:</code> .....	1080
		<code>\l_@@_old_iRow_int</code> .....	260, 1021, 2315
		<code>\@@_old_ialign:</code> .....	947, 1055, 4541
		<code>\@@_old_iddots</code> .....	1062, 3052
		<code>\l_@@_old_jCol_int</code> .....	261, 1024, 2316
		<code>\@@_old_ldots</code> .....	1058, 2976
		<code>\@@_old_multicolumn</code> .....	3067, 3074
		<code>\@@_old_pgftutil@check@rerun</code> .....	75, 79
		<code>\@@_old_vdots</code> .....	1060, 3006
		<code>\l_@@_parallelize_diags_bool</code> .....	436, 437, 517, 2200, 2705, 2756
		<code>\@@_patch_preamble:n</code> .....	1396, 1438, 1480, 1509, 1563, 1575
		<code>\@@_patch_preamble_i:n</code>	1442, 1443, 1444, 1466
		<code>\@@_patch_preamble_ii:nn</code> .....	1445, 1446, 1447, 1477
		<code>\@@_patch_preamble_iii:n</code> .	1448, 1482, 1490
		<code>\@@_patch_preamble_iii_i:n</code> ....	1485, 1487
		<code>\@@_patch_preamble_iv:nnn</code> .....	1449, 1450, 1451, 1512
		<code>\@@_patch_preamble_ix:n</code> .....	1568, 1578
		<code>\@@_patch_preamble_v:nnnn</code>	1452, 1453, 1529
		<code>\@@_patch_preamble_vi:n</code> .....	1454, 1551
		<code>\@@_patch_preamble_vii:n</code> .....	1462, 1557

<code>\@@_patch_preamble_viii:n</code> .....	<code>\g_@@_rotate_bool</code> .....
..... 1475, 1527, 1549, 1555, 1565, 1581	..... 241, 841, 868, 1543, 1998,
<code>\@@_pgf_rect_node:nnn</code> .....	2042, 3283, 4410, 4428, 4433, 4493, 4506, 4617
<code>\@@_pgf_rect_node:nnnnn</code> .....	<code>\@@_rotate_cell_box:</code> .....
..... 370, 4296, 4323, 4630, 4672	..... 829, 868, 1543, 1998, 2042, 4617
<code>\c_@@_pgfortikzpicture_tl</code> .....	<code>\g_@@_row_of_col_done_bool</code> .....
..... 38, 42, 2297, 3323, 4040	..... 264, 979, 1112, 1847
<code>\@@_picture_position:</code> .... 1162, 1170, 1178	<code>\g_@@_row_total_int</code> .....
<code>\l_@@_pos_of_block_tl</code> ... 287, 288, 4338,	..... 1089, 1300,
4340, 4342, 4358, 4359, 4361, 4410, 4414,	1619, 1718, 2155, 2162, 2169, 2185, 3198,
4421, 4471, 4494, 4496, 4507, 4510, 4531,	4156, 4170, 4197, 4292, 4558, 4635, 4653, 5158
4533, 4535, 4701, 4713, 4724, 4728, 4735, 4747	<code>\@@_rowcolor</code> .....
<code>\g_@@_pos_of_blocks_seq</code> 279, 1132, 2196,	..... 1192, 3387
2226, 2275, 3101, 3629, 3751, 4007, 4577, 4902	<code>\@@_rowcolor_tabular</code> .....
<code>\g_@@_pos_of_stroken_blocks_seq</code> .....	..... 1012, 3577
..... 281, 1133, 3633, 3755, 4591	<code>\@@_rowcolors</code> .....
<code>\g_@@_pos_of_xdots_seq</code> .....	..... 1193, 3458
..... 280, 1134, 2227, 2461, 3631, 3753	<code>\@@_rowcolors_i:nnnn</code> .....
<code>\@@_pre_array:</code> .....	..... 3502, 3535
..... 1015, 1249	<code>\l_@@_rowcolors_restart_bool</code> ...
<code>\c_@@_preamble_first_col_tl</code> .... 1407, 1972	..... 3454, 3485
<code>\c_@@_preamble_last_col_tl</code> .... 1419, 2016	<code>\g_@@_rows_seq</code> . 1797, 1799, 1801, 1803, 1805
<code>\g_@@_preamble_tl</code> .....	<code>\l_@@_rows_tl</code> .. 3365, 3394, 3407, 3432, 3504
..... 1381, 1391, 1394, 1404, 1407,	<code>\l_@@_rules_color_tl</code> .. 262, 479, 1145, 1146
1416, 1419, 1428, 1433, 1468, 1479, 1492,	<code>\@@_set_CT@arc@:</code> .....
1514, 1531, 1553, 1559, 1572, 1580, 1775, 1802	..... 151, 1146
<code>\@@_pred:n</code> .....	<code>\@@_set_CT@arc@_i:</code> .....
..... 118, 148, 2094, 3670, 3683, 3791, 3804	..... 152, 153
<code>\@@_provide_pgfsyspdfmark:</code> . 204, 213, 1104	<code>\@@_set_CT@arc@_ii:</code> .....
<code>\@@_put_box_in_flow:</code> .... 1357, 1583, 1768	..... 152, 155
<code>\@@_put_box_in_flow_bis:nn</code> .... 1356, 1735	<code>\@@_set_final_coords:</code> .....
<code>\@@_put_box_in_flow_i:</code> .....	..... 2474, 2499
..... 1589, 1591	<code>\@@_set_final_coords_from_anchor:n</code> ..
<code>\@@_qpoint:n</code> .....	... 2490, 2538, 2576, 2620, 2635, 2704, 2755
..... 226,	<code>\@@_set_initial_coords:</code> .....
1594, 1596, 1608, 1624, 1683, 1685, 1701,	..... 2469, 2488
1712, 1723, 2523, 2526, 2532, 2535, 2565,	<code>\@@_set_initial_coords_from_anchor:n</code> .
2572, 2581, 2583, 2626, 2632, 2640, 2642,	... 2479, 2529, 2569, 2619, 2629, 2696, 2747
2691, 2693, 2699, 2701, 2742, 2744, 2750,	<code>\@@_set_size:n</code> .....
2752, 3331, 3334, 3357, 3363, 3376, 3378,	..... 4826, 4835
3693, 3695, 3697, 3814, 3816, 3818, 4048,	<code>\c_@@_siunitx_loaded_bool</code> 157, 161, 166, 184
4052, 4059, 4095, 4098, 4100, 4202, 4212,	<code>\l_@@_small_bool</code> .....
4621, 4623, 4625, 4627, 4649, 4669, 4697,	..... 678,
4779, 4781, 4788, 4790, 4915, 4917, 4920, 4922	717, 725, 747, 775, 1027, 1980, 2025, 2215
<code>\l_@@_radius_dim</code> .....	<code>\@@_standard_cline</code> .....
..... 427, 428, 1560,	..... 114, 1064
2217, 2539, 2540, 2954, 4026, 4050, 4096, 4097	<code>\@@_standard_cline:w</code> .....
<code>\l_@@_real_left_delim_dim</code> 1737, 1752, 1767	..... 114, 115
<code>\l_@@_real_right_delim_dim</code> 1738, 1764, 1770	<code>\l_@@_standard_cline_bool</code> .. 420, 486, 1063
<code>\@@_rectanglecolor</code> .....	<code>\c_@@_standard_tl</code> 430, 431, 2781, 4072, 4102
..... 1191, 3413	<code>\g_@@_static_num_of_col_int</code> .....
<code>\@@_rectanglecolor_i:nn</code> .. 3420, 3424, 3444	..... 282, 1366, 1397, 4562, 5072, 5084, 5203
<code>\@@_renew_NC@rewrite@S:</code> .... 185, 187, 1092	<code>\l_@@_stop_loop_bool</code> .....
<code>\@@_renew_dots:</code> .....	..... 2325, 2326,
..... 999, 1085	2358, 2371, 2380, 2393, 2394, 2426, 2439, 2448
<code>\l_@@_renew_dots_bool</code> .....	<code>\@@_stroke_block:nnn</code> .....
..... 518, 1085, 4977	..... 4586, 4759
<code>\@@_renew_matrix:</code> .....	<code>\@@_succ:n</code> .....
..... 680, 4808, 4979	..... 143,
<code>\l_@@_respect_blocks_bool</code> 3451, 3466, 3494	147, 957, 963, 1507, 1596, 1930, 1936,
<code>\@@_restore_iRow_jCol:</code> .....	1941, 1942, 1955, 1959, 1964, 1965, 2190,
..... 2270, 2313	2532, 2572, 2583, 2632, 2642, 2699, 2701,
<code>\@@_revtex_array:</code> .....	2744, 2750, 3363, 3376, 3544, 3666, 3697,
..... 924, 935	3735, 3787, 3818, 3854, 3873, 4012, 4014,
<code>\c_@@_revtex_bool</code> .....	4016, 4018, 4059, 4100, 4262, 4266, 4276,
..... 46, 48, 51, 934	4280, 4625, 4627, 4669, 4788, 4790, 4920, 4922
<code>\l_@@_right_delim_dim</code> .....	<code>\l_@@_suffix_tl</code> .....
..... 1251, 1255, 1261, 1769, 2052	..... 4224, 4235,
<code>\l_@@_right_delim_tl</code> .....	4245, 4248, 4297, 4305, 4306, 4324, 4332, 4333
..... 1108, 4070	<code>\c_@@_table_collect_begin_tl</code> . 174, 176, 194
<code>\l_@@_right_margin_dim</code> .....	<code>\c_@@_table_print_tl</code> .....
..... 449, 527, 1279, 2053, 2649, 4064, 4288	..... 177, 178, 196
<code>\@@_rotate:</code> .....	<code>\l_@@_tabular_width_dim</code> .....
..... 1078, 3283	..... 239, 940, 942, 1430, 2123
	<code>\l_@@_tabularnote_tl</code> 304, 734, 757, 1640, 1649
	<code>\g_@@_tabularnotes_seq</code> .....
	..... 303, 343, 1655, 1661, 1677
	<code>\@@_test_if_cell_in_a_block:nn</code> .....
	..... 3946, 3964, 3982, 4002
	<code>\@@_test_if_cell_in_block:nnnnnnn</code> ...
	..... 4008, 4010
	<code>\@@_test_if_hline_in_block:nnnn</code> .....
	..... 3752, 3754, 3876

<code>\@@_test_if_hline_in_stroken_block:nnnn</code>	<code>\@</code> . . . . . 1788, 1810,
. . . . . 3756, 3898	4842, 4848, 4854, 4960, 4961, 4994, 5003,
<code>\@@_test_if_math_mode: . . . . 242, 1118, 2075</code>	5084, 5089, 5094, 5099, 5104, 5127, 5138,
<code>\@@_test_if_vline_in_block:nnnn . . . . .</code>	5144, 5151, 5158, 5166, 5178, 5184, 5191,
. . . . . 3630, 3632, 3887	5198, 5207, 5213, 5220, 5226, 5232, 5244,
<code>\@@_test_if_vline_in_stroken_block:nnnn</code>	5248, 5253, 5259, 5268, 5269, 5287, 5288,
. . . . . 3634, 3909	5330, 5331, 5381, 5382, 5432, 5433, 5485, 5486
<code>\@@_test_in_corner_h: . . . . . 3757, 3785</code>	<code>\{</code> . . . . . 256, 2082, 4870, 5165, 5220, 5330, 5432
<code>\@@_test_in_corner_v: . . . . . 3636, 3664</code>	<code>\}</code> . . . . . 256, 2082, 4870, 5165, 5220, 5330, 5432
<code>\l_@@_the_array_box . . 1263, 1266, 1634, 1635</code>	<code>\l</code> . . . . . 2084, 4869
<code>\c_@@_tikz_loaded_bool . . . . .</code>	
. . . . . 26, 37, 1182, 2238, 4079	
<code>\@@_true_c: . . . . . 195, 1454</code>	<code>\sq</code> . . . . . 5051, 5056, 5063, 5071, 5072, 5083,
<code>\l_@@_type_of_col_tl . . 715, 716, 2104, 2106</code>	5084, 5126, 5157, 5158, 5162, 5171, 5202,
<code>\c_@@_types_of_matrix_seq . . . . .</code>	5203, 5204, 5212, 5218, 5231, 5238, 5239, 5247
. . . . . 5027, 5028, 5033, 5037	
<code>\@@_update_for_first_and_last_row: . .</code>	<b>A</b>
. . . . . 812, 876, 1204, 2000, 2044	<code>\aboverulesep</code> . . . . . 1671
<code>\@@_use_arraybox_with_notes: . . . 1314, 1696</code>	<code>\addtocounter</code> . . . . . 360
<code>\@@_use_arraybox_with_notes_b: . 1311, 1680</code>	<code>\alph</code> . . . . . 306
<code>\@@_use_arraybox_with_notes_c: . . . . .</code>	<code>\arraybackslash</code> . . . . . 1520
. . . . . 1312, 1344, 1632, 1694, 1733	<code>\arraycolsep</code> . . . . . 526, 528, 530,
<code>\@@_vdottedline:n . . . . . 1562, 4075</code>	939, 1030, 1254, 1255, 1343, 1347, 4056, 4063
<code>\@@_vdottedline_i:n . . . . . 4082, 4087, 4091</code>	<code>\arrayrulecolor</code> . . . . . 89
<code>\@@_vline:nn . . . . . 1507, 3613, 3736</code>	<code>\arrayrulewidth</code> . . . . .
<code>\@@_vline_i:nn . . . . . 2231, 3618, 3622</code>	. . 122, 127, 139, 481, 803, 956, 958, 964,
<code>\@@_vline_i_complete:nn . . . . . 2231, 3729</code>	987, 1338, 1350, 1392, 1500, 1573, 1688,
<code>\@@_vline_ii:nnnn . . . 3645, 3656, 3689, 3730</code>	1727, 1854, 1856, 1862, 1872, 1874, 1880,
<code>\l_@@_vlines_bool . . . . . 438,</code>	1903, 1905, 1911, 1929, 1931, 1937, 3361,
511, 514, 544, 1389, 1413, 1425, 1570, 2235	3362, 3364, 3377, 3379, 3705, 3706, 3708,
<code>\@@_w: . . . . . 1383, 1452</code>	3719, 3725, 3827, 3838, 3844, 3869, 4137, 4763
<code>\g_@@_width_first_col_dim . . . . .</code>	<code>\arraystretch</code> . . . . . 1029, 4407, 4490, 4503
. . . . . 277, 1111, 1305, 1842, 2001, 2002	<code>\AtBeginDocument</code> . . . . .
<code>\g_@@_width_last_col_dim . . . . .</code>	. . . . . 23, 27, 67, 83, 158, 182, 310, 605,
. . . . . 276, 1110, 1361, 1951, 2045, 2046	621, 2293, 2960, 3132, 3208, 3286, 3319, 4034
<code>\l_@@_x_final_dim . . . . . 270,</code>	<code>\AutoNiceMatrix</code> . . . . . 4871
2476, 2533, 2534, 2573, 2574, 2622, 2644,	<code>\AutoNiceMatrixWithDelims</code> . . . 4831, 4863, 4875
2652, 2656, 2660, 2662, 2667, 2669, 2702,	
2711, 2719, 2753, 2762, 2770, 2808, 2822,	
2831, 2867, 2919, 2935, 3335, 4060, 4071, 4097	
<code>\l_@@_x_initial_dim . . . . .</code>	<b>B</b>
. . . . . 268, 2471, 2524, 2525, 2566, 2567,	<code>\baselineskip</code> . . . . . 92, 99
2622, 2643, 2644, 2651, 2656, 2660, 2662,	<code>\bgroup</code> . . . . . 1106
2664, 2667, 2669, 2694, 2711, 2719, 2745,	<code>\Block</code> . . . . . 1077, 5126, 5259
2762, 2770, 2799, 2821, 2831, 2867, 2919,	<code>\BNiceMatrix</code> . . . . . 4823
2933, 2935, 2953, 2955, 3332, 4053, 4068, 4096	<code>\bNiceMatrix</code> . . . . . 4820
<code>\l_@@_xdots_color_tl 454, 468, 2514, 2556,</code>	bool commands:
2608, 2609, 2682, 2733, 2789, 3202, 3277, 3294	<code>\bool_do_until:Nn</code> . . . . . 2326, 2394
<code>\l_@@_xdots_down_tl . . . 472, 2805, 2815, 2850</code>	<code>\bool_gset_false:N</code> . . . . .
<code>\l_@@_xdots_line_style_tl . . . . .</code>	. . . . . 841, 886, 887, 978, 1093,
. . . . . 429, 431, 464, 2781, 2789, 4072, 4102	1112, 2012, 2058, 3885, 3896, 3907, 3918, 4433
<code>\l_@@_xdots_shorten_dim . . . . . 425,</code>	<code>\bool_gset_true:N</code> . . . . .
426, 470, 2219, 2796, 2797, 2893, 2904, 2912	. . . . . 1847, 1976, 2020, 2111, 2977,
<code>\l_@@_xdots_up_tl . . . . 473, 2801, 2814, 2840</code>	2992, 3007, 3030, 3053, 3064, 3283, 3628, 3750
<code>\l_@@_y_final_dim . . . . . 271,</code>	<code>\bool_if:NTF</code> . . . . .
2477, 2536, 2540, 2585, 2589, 2591, 2633,	. . . . . 150, 166, 607, 612, 623, 628, 772,
2700, 2713, 2716, 2751, 2764, 2767, 2808,	775, 868, 952, 979, 982, 1017, 1027, 1084,
2822, 2830, 2869, 2924, 2943, 3336, 4051, 4101	1085, 1105, 1119, 1129, 1147, 1182, 1196,
<code>\l_@@_y_initial_dim . . . . .</code>	1198, 1264, 1284, 1359, 1364, 1375, 1399,
. . . . . 269, 2472, 2527, 2539, 2584, 2585, 2589,	1543, 1570, 1667, 1833, 1850, 1868, 1886,
2591, 2627, 2692, 2713, 2718, 2743, 2764,	1899, 1925, 1946, 1952, 1980, 1998, 2025,
2769, 2799, 2821, 2830, 2869, 2924, 2941,	2042, 2132, 2134, 2153, 2156, 2175, 2200,
2943, 2953, 2956, 3333, 4049, 4050, 4051, 4099	2215, 2234, 2235, 2238, 2588, 2590, 2705,
	2756, 2976, 2991, 3006, 3029, 3052, 3955,
	3973, 3991, 4120, 4130, 4152, 4410, 4428,
	4493, 4506, 4617, 4675, 4882, 5006, 5016, 5042

`\bool_if:nTF` ..... 184,  
 312, 339, 915, 1294, 3308, 3550, 4146, 4725  
`\bool_lazy_all:nTF` .....  
 ... 1409, 1421, 2224, 3878, 3889, 3900, 3911  
`\bool_lazy_and:nnTF` .....  
 ..... 1889, 1981, 2187, 2577,  
 2813, 3116, 3358, 3465, 3493, 3699, 3820, 4775  
`\bool_lazy_or:nnTF` .....  
 . 461, 880, 1617, 1638, 1716, 2028, 2617,  
 2873, 3542, 3947, 3965, 3983, 4376, 4557, 4580  
`\bool_lazy_or_p:nn` ..... 1984  
`\bool_not_p:n` .....  
 1412, 1413, 1414, 1424, 1425, 1426, 1891, 2189  
`\bool_set:Nn` ..... 2621, 3487  
`\c_false_bool` ..... 2972, 2987, 3002  
`\g_tmpa_bool` .....  
 .... 3628, 3637, 3671, 3679, 3684, 3750,  
 3758, 3792, 3800, 3805, 3885, 3896, 3907, 3918  
`\l_tmpb_bool` ... 3952, 3966, 3984, 4006, 4019  
 box commands:  
`\box_clear_new:N` ..... 1019, 1263  
`\box_dp:N` ..... 797, 817, 854,  
 873, 1045, 1054, 1209, 1586, 1746, 1759, 4463  
`\box_gclear_new:N` ..... 4398  
`\box_grotate:Nn` ..... 4430  
`\box_ht:N` ... 798, 819, 825, 837, 860, 871,  
 1047, 1049, 1052, 1207, 1585, 1746, 1759, 4454  
`\box_move_up:nn` . 58, 60, 62, 1629, 1694, 1733  
`\box_rotate:Nn` ..... 831  
`\box_set_dp:Nn` ..... 853, 872, 1586  
`\box_set_ht:Nn` ..... 859, 870, 1585  
`\box_set_wd:Nn` ..... 847  
`\box_use:N` ..... 363, 838, 1521, 1524  
`\box_use_drop:N` .... 878, 884, 902, 1545,  
 1588, 1629, 1630, 1635, 2007, 4473, 4720, 4754  
`\box_wd:N` .....  
 ... 364, 848, 875, 882, 1259, 1261, 1634,  
 1753, 1765, 2002, 2005, 2046, 2050, 4442, 4889  
`\l_tmpa_box` .....  
 .. 346, 363, 364, 1258, 1259, 1260, 1261,  
 1329, 1585, 1586, 1588, 1629, 1630, 1746, 1759  
`\l_tmpb_box` ..... 1739, 1753, 1754, 1765

## C

`\c` ..... 203, 1403  
`\Cdots` ..... 1067, 2982, 2985  
`\cdots` ..... 1002, 1059  
`\cellcolor` ..... 1011, 1190, 3575  
`\chessboardcolors` ..... 1195  
`\cline` ..... 142, 1064, 1065  
 clist commands:  
`\clist_if_empty:NnTF` ..... 3635, 3757  
`\clist_map_inline:Nn` ..... 3346, 3365, 3923  
`\clist_map_inline:nn` ..... 2099, 3443, 3478  
`\clist_new:N` ..... 440  
`\clist_set:Nn` ..... 543  
`\CodeAfter` ..... 766, 1081, 1791, 1794, 2251  
`\color` ..... 93, 100, 154, 156,  
 1333, 2508, 2511, 2514, 2550, 2553, 2556,  
 2602, 2605, 2609, 2682, 2733, 3196, 3199,  
 3202, 3271, 3274, 3277, 3294, 3393, 3406,  
 3419, 3442, 3511, 3512, 3522, 3523, 4405, 4527  
`\colorlet` ..... 248, 249, 782, 789, 1990, 2033  
`\columncolor` ..... 1013, 1194, 2258, 3591

`\cr` ..... 126, 144, 1970  
`\crrcr` ..... 1827  
 cs commands:  
`\cs_generate_variant:Nn` ... 146, 4142, 4143  
`\cs_gset:Npn` .....  
 ..... 93, 100, 2140, 2147, 2161, 2168, 4135  
`\cs_gset_eq:NN` ... 179, 213, 1038, 1121, 2271  
`\cs_if_exist:NnTF` .....  
 1020, 1023, 1122, 1125, 1217, 1224, 1235,  
 1242, 2315, 2316, 2361, 2374, 2429, 2442,  
 3164, 3182, 3239, 3257, 4122, 4175, 4637, 4655  
`\cs_if_exist_p:N` .....  
 ..... 462, 3468, 3496, 3949, 3968, 3986  
`\cs_if_free:NnTF` .....  
 ..... 209, 2503, 2545, 2597, 2677, 2728  
`\cs_if_free_p:N` ..... 3310, 3312  
`\cs_new_protected:Npx` .... 2295, 3321, 4036  
`\cs_set:Nn` ..... 593, 596, 599  
`\cs_set:Npn` ..... 89,  
 90, 96, 97, 102, 114, 115, 129, 131, 132,  
 154, 156, 309, 994, 2320, 2382, 2450, 3206,  
 3281, 3857, 3858, 3864, 3865, 4407, 4490, 4503  
`\cs_set_nopar:Npn` .....  
 ..... 929, 941, 1029, 1032, 4317, 4318  
`\cs_set_nopar:Npx` ..... 942  
`\cs_set_protected:Npn` ..... 4860  
`\cs_set_protected_nopar:Npn` .... 4386, 4604

## D

`\Ddots` ..... 1069, 3013, 3014, 3019, 3020  
`\ddots` ..... 1004, 1061  
`\diagbox` ..... 1082, 4386, 4604  
 dim commands:  
`\dim_add:Nn` ..... 4286  
`\dim_compare:nNnTF` .....  
 ..... 92, 99, 845, 851, 857, 1430,  
 1887, 2050, 2662, 4199, 4209, 4647, 4667, 4889  
`\dim_gzero:N` ..... 849, 855, 861  
`\dim_max:nn` ..... 4188, 4192  
`\dim_min:nn` ..... 4180, 4184  
`\dim_ratio:nn` ..... 2720, 2771,  
 2887, 2892, 2903, 2911, 2920, 2925, 2936, 2944  
`\dim_set:Nn` .... 4161, 4168, 4179, 4183,  
 4187, 4191, 4203, 4204, 4213, 4214, 4258, 4271  
`\dim_set_eq:NN` ..... 4159, 4166, 4266, 4280  
`\dim_sub:Nn` ..... 4283  
`\dim_use:N` .....  
 .... 4200, 4210, 4261, 4262, 4274, 4275,  
 4298, 4299, 4300, 4301, 4325, 4326, 4327, 4328  
`\dim_zero_new:N` .... 4158, 4160, 4165, 4167  
`\c_max_dim` ..... 4159, 4161,  
 4166, 4168, 4200, 4210, 4634, 4647, 4652, 4667  
`\l_tmpc_dim` .....  
 ..... 272, 3361, 3362, 3381, 3698, 3706,  
 3711, 3716, 3722, 3819, 3830, 3835, 3841,  
 4626, 4632, 4674, 4782, 4793, 4921, 4924, 4932  
`\l_tmpd_dim` .... 273, 3379, 3381, 3707,  
 3711, 3826, 3830, 4628, 4632, 4652, 4663,  
 4667, 4670, 4674, 4791, 4794, 4923, 4924, 4936  
`\dotfill` ..... 1080, 4878  
`\dots` ..... 1006  
`\doublerulesep` 1501, 3708, 3720, 3827, 3839, 3870  
`\doublerulesepcolor` ..... 96  
`\draw` ..... 2793

<b>E</b>	
<code>\egroup</code> .....	1374
else commands:	
<code>\else:</code> .....	244
<code>\endarray</code> .....	1779, 1807
<code>\endBNiceMatrix</code> .....	4824
<code>\endbNiceMatrix</code> .....	4821
<code>\endNiceArray</code> .....	2119, 2128
<code>\endNiceArrayWithDelims</code> .....	2068, 2078
<code>\endpgfscope</code> .....	2855, 4935
<code>\endpNiceMatrix</code> .....	4812
<code>\endsavenotes</code> .....	1375
<code>\endtabularnotes</code> .....	1662
<code>\endVNiceMatrix</code> .....	4818
<code>\endvNiceMatrix</code> .....	4815
<code>\everycr</code> .....	125, 144, 1042
exp commands:	
<code>\exp_after:wN</code> .....	191, 1146, 1396
<code>\exp_args:Ne</code> .....	3077
<code>\exp_args:NNc</code> .....	4124
<code>\exp_args:NNe</code> .....	3073
<code>\exp_args:Nne</code> .....	2106
<code>\exp_args:NNV</code> .....	1799,
2964, 2979, 2994, 3009, 3032, 3136, 3212, 3290	
<code>\exp_args:NNv</code> .....	1138
<code>\exp_args:Nnx</code> .....	4413, 4420, 4495, 4509
<code>\exp_args:No</code> .....	2788
<code>\exp_args:NV</code> .....	1775, 1802, 1804
<code>\exp_args:Nxx</code> .....	4379, 4380
<code>\exp_not:N</code> .....	38,
39, 42, 43, 1494, 3581, 3591, 4038, 4039, 4598	
<code>\exp_not:n</code> .....	
..... 921, 2264, 3146, 3222, 3583, 4395,	
4471, 4483, 4484, 4587, 4599, 4613, 4899, 4900	
<code>\ExplSyntaxOff</code> ..	211, 2151, 2172, 2198, 2266, 4139
<code>\ExplSyntaxOn</code> ..	208, 2137, 2158, 2177, 2259, 4132
<code>\extrarowheight</code> .....	4408, 4491, 4504
<b>F</b>	
<code>\fi</code> .....	104, 1387, 3857, 3874
fi commands:	
<code>\fi:</code> .....	246
<code>\fontdimen</code> .....	1625
fp commands:	
<code>\fp_eval:n</code> .....	2826
<code>\fp_to_dim:n</code> .....	2863
<code>\futurelet</code> .....	109
<b>G</b>	
<code>\globaldefs</code> .....	1370, 4567
group commands:	
<code>\group_insert_after:N</code> ..	4883, 4884, 4886, 4887
<b>H</b>	
<code>\halign</code> .....	1056
<code>\hbox</code> .....	950,
1339, 1694, 1733, 1852, 1870, 1897, 1901, 1927	
hbox commands:	
<code>\hbox:n</code> .....	58, 60, 63
<code>\hbox_gset:Nn</code> .....	4400
<code>\hbox_overlap_left:n</code> .....	1831, 2003
<code>\hbox_overlap_right:n</code> .....	363, 1948, 2048
<code>\hbox_set:Nn</code> .....	
..... 346, 1258, 1260, 1329, 1739, 1754, 4616	

<code>\hbox_set:Nw</code> ....	771, 1266, 1534, 1978, 2023
<code>\hbox_set_end:</code> ..	867, 1281, 1542, 1997, 2041
<code>\hbox_to_wd:nn</code> .....	388, 413
<code>\Hdotsfor</code> .....	1074, 5051, 5171
<code>\hdotsfor</code> .....	1007
<code>\hdottedline</code> .....	1071
<code>\heavyrulewidth</code> .....	1672
<code>\hfil</code> .....	1453
<code>\hfill</code> .....	122, 139
<code>\Hline</code> .....	1072, 3860
<code>\hline</code> .....	102
<code>\hrule</code> .....	106, 122, 139, 987, 1672
<code>\hskip</code> .....	105
<code>\Hspace</code> .....	1073
<code>\hspace</code> .....	3065
<code>\hss</code> .....	1453
<b>I</b>	
<code>\ialign</code> .....	947, 1032, 1055, 4541
<code>\Iddots</code> .....	1070, 3036, 3037, 3042, 3043
<code>\iddots</code> .....	53, 1005, 1062
if commands:	
<code>\if_mode_math:</code> .....	244
<code>\ifnum</code> .....	104, 3857, 3874
<code>\ifstandalone</code> .....	1125
int commands:	
<code>\int_case:nnTF</code> .....	3011, 3017, 3034, 3040
<code>\int_compare:nNnTF</code> .....	117,
118, 134, 768, 769, 779, 786, 822, 832, 984,	
986, 1157, 1159, 1202, 1212, 1231, 1282,	
1286, 1297, 1301, 1302, 1366, 1650, 1689,	
1728, 1800, 1828, 2026, 2336, 2343, 2347,	
2349, 2404, 2411, 2415, 2417, 2510, 2552,	
2604, 2645, 2647, 3096, 3110, 3198, 3273,	
3355, 3374, 3537, 3588, 3602, 3603, 3610,	
3611, 3615, 4012, 4014, 4016, 4018, 4404,	
4435, 4447, 4694, 4723, 4727, 4784, 4786,	
4838, 4840, 4842, 4846, 4848, 4850, 4852, 4854	
<code>\int_compare_p:n</code> ....	3552, 3554, 4776, 4777
<code>\int_do_until:nNnn</code> .....	3490
<code>\int_gadd:Nn</code> .....	3109
<code>\int_gincr:N</code> ..	767, 795, 1128, 1474, 1526,
1548, 1554, 1923, 2021, 2707, 2758, 4117, 4385	
<code>\int_if_even:nTF</code> .....	3565
<code>\int_if_odd_p:n</code> .....	3487
<code>\int_incr:N</code> .....	342, 1484
<code>\int_step_inline:nn</code> .....	3561, 3563
<code>\int_step_inline:nnnn</code> ..	3944, 3962, 3977, 3980
<code>\l_tmpc_int</code> .....	3488, 3489, 3490
<code>\c_zero_int</code> .....	3543
iow commands:	
<code>\iow_now:Nn</code> .....	70,
206, 2177, 2178, 2180, 2198, 2259, 2260, 2266	
<code>\iow_shipout:Nn</code> .....	2137, 2138, 2145,
2151, 2158, 2159, 2166, 2172, 4132, 4133, 4139	
<code>\item</code> .....	1655, 1661
<b>K</b>	
<code>\kern</code> .....	63
keys commands:	
<code>\keys_define:nn</code> .....	457,
477, 484, 537, 583, 635, 671, 707, 723, 740,	
751, 3056, 3449, 4106, 4336, 4518, 4801, 4973	



\l_keys_key_str .....	4960, 5115, 5121, 5253, 5258, 5268, 5286, 5329, 5380, 5431
\keys_set:nn .....	492, 699, 706, 1142, 1143, 2105, 2115, 2124, 2513, 2555, 2607, 2681, 2732, 3201, 3276, 3293, 3463, 4119, 4579
\keys_set_known:nn ..	3023, 3046, 4362, 4764
\l_keys_value_tl .....	5186, 5193, 5480
<b>L</b>	
\Ldots .....	1066, 2967, 2970
\ldots .....	1001, 1058
\leaders .....	122, 139
\left .....	1334, 1742, 1757
legacy commands:	
\legacy_if:nTF .....	567
\line .....	2250, 5165
<b>M</b>	
\makebox .....	1545
\mathinner .....	55
mode commands:	
\mode_leave_vertical: .....	1117, 1519
msg commands:	
\msg_error:nn .....	12
\msg_error:nnn .....	13
\msg_error:nnnn .....	14, 4564, 4571
\msg_fatal:nn .....	15
\msg_fatal:nnn .....	16
\msg_info:nn .....	5009
\msg_new:nnn .....	17
\msg_new:nnnn .....	18
\msg_redirect_name:nnn .....	20
\multicolumn ..	1076, 3067, 3071, 3122, 3128, 3149
\multispan .....	118, 119, 135, 136
\myfiledate .....	6
\myfileversion .....	7
<b>N</b>	
\newcolumntype .....	218, 219, 220
\newcounter .....	302
\NewDocumentCommand .....	
..	314, 337, 705, 2964, 2979, 2994, 3009, 3032, 3136, 3212, 3290, 3387, 3400, 3413, 3436, 3458, 3559, 3572, 3577, 3586, 4831, 4871
\NewDocumentEnvironment .....	1102, 1772, 1781, 2061, 2071, 2101, 2112, 2120, 4115
\NewExpandableDocumentCommand .....	224, 4347
\newlist .....	318, 329
\NiceArray .....	2117, 2126
\NiceArrayWithDelims .....	2066, 2076
nicematrix commands:	
\g_nicematrix_code_after_tl .....	
	258, 576, 1796, 2252, 2253, 4584, 4944, 4952
\g_nicematrix_code_before_tl .....	
	1100, 2255, 2264, 3574, 3579, 3590, 4596
\NiceMatrixLastEnv .....	224
\NiceMatrixOptions .....	705, 5287, 5485
\NiceMatrixoptions .....	5190
\noalign ..	92, 99, 104, 127, 974, 1038, 3857, 4026
\nobreak .....	332
\normalbaselines .....	1026
\NotEmpty .....	1083
\nulldelimiterspace .....	1753, 1765
\numexpr .....	147, 148
<b>O</b>	
\omit .....	117, 1830, 1846, 1922, 4941
\OnlyMainNiceMatrix .....	1079, 3594
<b>P</b>	
\par .....	1649, 1657
peek commands:	
\peek_meaning:NTF .....	152, 344, 995
\peek_meaning_ignore_spaces:NTF .....	1774, 3860
\peek_meaning_remove_ignore_spaces:NTF .....	142
\peek_remove_spaces:n .....	3094
\pgfextracty .....	4697
\pgfgetlastxy .....	406
\pgfpathcircle .....	2952
\pgfpathlineto ....	3716, 3722, 3835, 3841, 4924
\pgfpathmoveto ....	3715, 3721, 3834, 3840, 4919
\pgfpathrectanglecorners .....	3380, 3709, 3828, 4792
\pgfpointhead .....	404
\pgfpointheadanchor .....	227, 2481, 2492, 4178, 4186, 4642, 4660, 4679, 4681, 4698, 4732
\pgfpointdiff .....	405, 1170, 1178
\pgfpointlineatime .....	2820
\pgfpointorigin .....	1837, 1960
\pgfpointscale .....	404
\pgfpointshapeborder .....	3331, 3334
\pgfrememberpicturepositiononpagetrue .....	800, 893, 962, 1836, 1860, 1878, 1909, 1935, 1958, 2304, 2779, 2857, 3330, 3691, 3812, 4047, 4094, 4221, 4231, 4242, 4619, 4766, 4914
\pgfscope .....	2817, 4931
\pgfset .....	373, 398, 894, 4709, 4743, 4930
\pgfsetbaseline .....	892
\pgfsetlinewidth .....	3725, 3844, 4795
\pgfsetrectcap .....	3726, 3845
\pgfsetroundcap .....	4927
\pgfsetstrokecolor .....	4772
\pgfsyspdfmark .....	209, 210
\pgftransformrotate .....	2824
\pgftransformshift .....	
....	379, 404, 2818, 4708, 4730, 4932, 4936
\pgfusepath .....	2844, 2854
\pgfusepathqfill ..	2958, 3385, 3445, 3712, 3831
\pgfusepathqstroke .....	3727, 3846, 4796, 4928
\phantom .....	2976, 2991, 3006, 3029, 3052
\pNiceMatrix .....	4811
prg commands:	
\prg_do_nothing: .....	
	170, 179, 185, 213, 474, 1038, 2251
\prg_new_conditional:Nnn .....	3540, 3548
\prg_replicate:nn .....	352, 1918, 1919, 3149, 3717, 3836, 4841, 4844, 4847, 4853
\prg_return_false: .....	3545, 3557
\prg_return_true: .....	3546, 3556
\ProcessKeysOptions .....	4987
\ProvideDocumentCommand .....	53
\ProvidesExplPackage .....	4
<b>Q</b>	
\quad .....	333
quark commands:	
\q_stop .....	
	114, 115, 131, 132, 153, 155, 1146, 1396, 1458, 1791, 1794, 3284, 3298, 3299, 3339,

3350, 3351, 3369, 3370, 3426, 3431, 3482, 4315, 4322, 4349, 4352, 4774, 4783, 4826, 4835	
<b>R</b>	
\rectanglecolor . . . . .	1191, 2257, 3581, 4598
\refstepcounter . . . . .	361
regex commands:	
\regex_const:Nn . . . . .	203
\regex_replace_all:NnN . . . . .	1401
\relax . . . . .	147, 148
\renewcommand . . . . .	189
\RenewDocumentEnvironment . . . . .	4810, 4813, 4816, 4819, 4822
\RequirePackage . . . . .	1, 3, 9, 10, 11
\right . . . . .	1352, 1749, 1761
\rotate . . . . .	1078
\rowcolor . . . . .	1012, 1192
\rowcolors . . . . .	1193
<b>S</b>	
\savenotes . . . . .	1105
\scriptstyle . . . . .	775, 1980, 2025, 2801, 2805, 2840, 2850
seq commands:	
\seq_clear_new:N . . . . .	2179, 3922
\seq_count:N . . . . .	1801
\seq_gclear:N . . . . .	1131, 1132, 1133, 1134, 1677
\seq_gclear_new:N . . . . .	1086, 1087, 1797, 1813
\seq_gpop_left:NN . . . . .	1803, 1815
\seq_gput_left:Nn . . . . .	572, 3098, 3100, 4577
\seq_gput_right:Nn . . . . .	343, 2461, 3101, 4468, 4480, 4591, 4902
\seq_gset_from_clist:Nn . . . . .	2182, 2194
\seq_gset_map_x:NNn . . . . .	2275
\seq_gset_split:Nnn . . . . .	1799, 1814
\seq_if_empty:NTF . . . . .	1637
\seq_if_empty_p:N . . . . .	2226, 2227, 2228
\seq_if_exist:NTF . . . . .	1149
\seq_if_in:NnTF . . . . .	570, 3668, 3674, 3681, 3789, 3795, 3802, 4181, 4189, 4640, 4658, 5037
\seq_item:Nn . . . . .	1153, 1156, 1165, 1166, 1173, 1174
\seq_map_function:NN . . . . .	1805
\seq_map_inline:Nn . . . . .	1655, 1661, 1817, 3502, 3629, 3631, 3633, 3751, 3753, 3755, 4007, 4542
\seq_mapthread_function:NNN . . . . .	4310
\seq_new:N . . . . .	235, 278, 279, 280, 281, 303, 5027
\seq_put_right:Nn . . . . .	3994
\seq_set_eq:NN . . . . .	3471
\seq_set_filter:NNn . . . . .	3473, 3500
\seq_set_from_clist:Nn . . . . .	5028
\seq_set_map_x:NNn . . . . .	5033
\seq_use:Nnnn . . . . .	2196, 5490
\l_tmpa_seq . . . . .	3473, 3500
\l_tmpb_seq . . . . .	3471, 3473, 3500, 3502
\setlist . . . . .	319, 330, 608, 613, 624, 629
skip commands:	
\skip_gadd:Nn . . . . .	1894
\skip_gset:Nn . . . . .	1885
\skip_gset_eq:NN . . . . .	1892, 1893
\skip_horizontal:N . . . . .	123, 140, 1267, 1268, 1279, 1280, 1304, 1305, 1342, 1343, 1346, 1347, 1361, 1362, 1392, 1560, 1573, 1766, 1767, 1769, 1770, 1841, 1842, 1854, 1856, 1872, 1874, 1896,
1903, 1905, 1924, 1929, 1931, 1950, 1951, 2008, 2009, 2010, 2013, 2047, 2052, 2053, 2054	
\skip_horizontal:n . . . . .	364, 1496
\skip_vertical:N . . . . .	127, 956, 958, 1337, 1338, 1349, 1350, 1646, 1671, 4026
\skip_vertical:n . . . . .	837, 3867
\g_tmpa_skip . . . . .	1885, 1892, 1893, 1894, 1896, 1924
\c_zero_skip . . . . .	1043
\space . . . . .	255, 256
\stepcounter . . . . .	350, 355
str commands:	
\c_backslash_str . . . . .	255
\c_colon_str . . . . .	704
\str_case:nn . . . . .	3077, 4701, 4713, 4735, 4747
\str_case:nnTF . . . . .	1309, 1440, 1611, 3925
\str_foldcase:n . . . . .	3077
\str_gclear:N . . . . .	2269
\str_gset:Nn . . . . .	1114, 2065, 2074, 2103, 2114, 2122, 4862
\str_if_empty:NTF . . . . .	804, 905, 965, 1113, 1215, 1233, 1838, 1863, 1881, 1912, 1938, 1961, 2064, 2073, 2143, 2164, 4302, 4329
\str_if_eq:nnTF . . . . .	78, 254, 945, 1461, 1489, 1567, 1821, 4770, 4949
\str_if_eq_p:nn . . . . .	463, 3359
\str_if_in:NnTF . . . . .	1599, 1703
\str_new:N . . . . .	250, 433, 445, 703
\str_range:Nnn . . . . .	1603, 1707
\str_set:Nn . . . . .	569, 695
\str_set_eq:NN . . . . .	573, 704
\l_tmpa_str . . . . .	569, 570, 572, 573
\strut . . . . .	1655, 1661
<b>T</b>	
\tabcolsep . . . . .	938, 1342, 1346
\tabskip . . . . .	1043
\tabularnote . . . . .	314, 337, 344, 5218, 5231
\tabularnotes . . . . .	1660
TeX and L <sup>A</sup> T <sub>E</sub> X commands:	
\@BTnormal . . . . .	1018
\@acol . . . . .	928
\@acoll . . . . .	926
\@acolr . . . . .	927
\@array@array . . . . .	930
\@arrayacol . . . . .	926, 927, 928
\@arstrutbox . . . . .	797, 798, 837, 1045, 1047, 1049, 1052, 1054, 1521, 1524
\@currenvir . . . . .	4949
\@gobblethree . . . . .	210
\@halignto . . . . .	929, 941, 942
\@height . . . . .	107, 122, 139
\@ifclassloaded . . . . .	47, 50, 5008, 5018
\@ifnextchar . . . . .	1091
\@ifpackageloaded . . . . .	29, 32, 35, 69, 85, 160, 5011, 5021
\@mainaux . . . . .	70, 206, 2137, 2138, 2145, 2151, 2158, 2159, 2166, 2172, 2177, 2178, 2180, 2198, 2259, 2260, 2266, 4132, 4133, 4139
\@tabarray . . . . .	943
\@tempswafalse . . . . .	1387
\@tempswatrue . . . . .	1386
\@temptokena . . . . .	169, 172, 191, 193, 1385, 1396
\@whiles . . . . .	1387
\@width . . . . .	107

<code>\@xhline</code> .....	110	<code>\tl_if_empty:nTF</code> .....	550, 675, 709, 727, 743, 759, 1783, 1810, 2514, 2556, 2608, 2682, 2733, 3202, 3277, 3294, 3393, 3406, 3419, 3442, 3510, 3521, 4363, 4366, 5050
<code>\bBigg@</code> .....	1258, 1260	<code>\tl_if_empty_p:N</code> .....	2814, 2815, 4581
<code>\c@MaxMatrixCols</code> .....	2093, 5065	<code>\tl_if_empty_p:n</code> .....	1640
<code>\c@tabularnote</code> .....	1639, 1650, 1678	<code>\tl_if_eq:NNTF</code> .....	2781, 4067, 4070
<code>\col@sep</code> .....	938, 939, 1304, 1362, 1841, 1894, 1950, 2013, 2047, 2525, 2534, 2567, 2574	<code>\tl_if_eq:NnTF</code> .....	1587, 1698
<code>\CT@arc</code> .....	89, 90	<code>\tl_if_eq:nnTF</code> .....	562, 686, 1786, 1788
<code>\CT@arc@</code> ..	88, 93, 108, 121, 138, 154, 156, 1121, 1672, 2271, 3724, 3843, 4093, 4771, 4926	<code>\tl_if_exist:NNTF</code> .....	1135
<code>\CT@drsc</code> .....	96, 97	<code>\tl_if_in:NnTF</code> .....	3349, 3368, 3481
<code>\CT@drsc@</code> ...	95, 100, 3701, 3704, 3822, 3825	<code>\tl_if_single_token:nTF</code> .....	694
<code>\CT@everycr</code> .....	1036	<code>\tl_item:Nn</code> .....	175, 176, 178
<code>\CT@row@color</code> .....	1038	<code>\tl_map_inline:nn</code> .....	1784
<code>\if@tempswa</code> .....	1387	<code>\tl_new:N</code> ...	174, 177, 230, 258, 259, 262, 266, 283, 284, 285, 287, 304, 429, 452, 454, 455
<code>\NC@</code> .....	994	<code>\tl_put_left:Nn</code> .....	1018
<code>\NC@find</code> .....	170, 198	<code>\tl_put_right:Nn</code> .....	552, 1138, 1204
<code>\NC@list</code> .....	1387	<code>\tl_range:nnn</code> .....	78
<code>\NC@rewrite@S</code> .....	171, 189	<code>\tl_set:Nn</code> ..	175, 3354, 3356, 3373, 3375, 3432, 3433, 3483, 3504, 3624, 4369, 4785, 4787
<code>\new@ifnextchar</code> .....	1091	<code>\tl_set_eq:NN</code> .....	431, 3429, 3430, 3640, 3761, 4072, 4102, 4359
<code>\newcol@</code> .....	996, 997	<code>\tl_set_rescan:Nnn</code> .....	1798, 2963, 3135, 3211, 3289
<code>\nicematrix@redefine@check@rerun</code> ..	70, 73	<code>\tl_to_str:n</code> .....	5034
<code>\pgf@relevantforpicturesizefalse</code> ....	2305, 2780, 2858, 2949, 3392, 3405, 3418, 3441, 3477, 3692, 3813, 4222, 4232, 4243, 4620, 4767, 4913	<code>\g_tmpa_tl</code> .....	172, 175, 178
<code>\pgfsys@getposition</code> .....	1162, 1168, 1176	<code>\l_tmpb_tl</code> .....	3342, 3353, 3354, 3355, 3356, 3363, 3372, 3373, 3374, 3375, 3376, 3430, 3433, 3483, 3489, 3624, 3666, 3670, 3676, 3678, 3683, 3748, 3761, 3770, 3791, 3797, 3804, 3882, 3883, 3893, 3894, 3904, 3905, 3915, 3916, 4777, 4781, 4786, 4787, 4790
<code>\pgfsys@markposition</code> .....	957, 1161, 1834, 1855, 1873, 1904, 1930, 1954	<code>\l_tmpc_tl</code> .....	3427, 3429, 3432, 3625, 3639, 3640, 3643, 3648, 3650, 3654, 3659, 3661, 3747, 3760, 3761, 3764, 3769, 3771, 3775, 3780, 3782
<code>\pgfutil@check@rerun</code> .....	75, 76	<code>\l_tmpd_tl</code> .....	3428, 3430, 3433
<code>\reserved@a</code> .....	109	token commands:	
<code>\set@color</code> .....	4404	<code>\token_to_str</code> .....	5133
<code>\tikz@library@external@loaded</code> .....	1122	<code>\token_to_str:N</code> .....	5051, 5126, 5138, 5165, 5171, 5218, 5231, 5248, 5258, 5287, 5485
tex commands:			
<code>\tex_mkern:D</code> .....	57, 59, 61, 64		
<code>\tex_the:D</code> .....	193		
<code>\textfont</code> .....	1625		
<code>\textit</code> .....	306		
<code>\textsuperscript</code> .....	307, 308		
<code>\the</code> .....	147, 148, 1387, 1396		
<code>\thetabularnote</code> .....	309		
<code>\tikzexternaldisable</code> .....	1124		
<code>\tikzset</code> .....	1126, 1184, 2240		
tl commands:			
<code>\tl_clear:N</code> ....	3650, 3661, 3771, 3782, 4762		
<code>\tl_clear_new:N</code>	3427, 3428, 3461, 3625, 3747		
<code>\tl_const:Nn</code> ..	38, 39, 42, 43, 430, 1972, 2016		
<code>\tl_count:n</code> .....	1606, 1710		
<code>\tl_gclear:N</code> .....	1394, 2237, 2253		
<code>\tl_gclear_new:N</code>	1094, 1095, 1096, 1097, 1098, 1099, 1100		
<code>\tl_gput_left:Nn</code> .....	915, 1407, 1416, 3590		
<code>\tl_gput_right:Nn</code> .....	915, 1419, 1428, 1432, 1468, 1479, 1492, 1506, 1514, 1531, 1553, 1559, 1561, 1572, 1580, 1796, 3138, 3214, 3574, 3579, 3872, 4031, 4388, 4584, 4596, 4606, 4892, 4944, 4952		
<code>\tl_gset:Nn</code> ..	172, 176, 178, 1381, 1391, 2262		
<code>\tl_if_blank:nTF</code> .....	3389, 3402, 3415, 3438, 3508, 3519, 4349		
<code>\tl_if_blank_p:n</code> .....	3701, 3822		
<code>\tl_if_empty:NNTF</code> .....	1145, 1332, 1649, 2255, 3352, 3353, 3371, 3372, 3639, 3643, 3654, 3760, 3764, 3775, 4357, 4403, 4594, 4768		
		U	
		<code>\unskip</code> .....	1665
		use commands:	
		<code>\use:N</code> .....	918, 1220, 1227, 1238, 1245, 1271, 1272, 1276, 1277, 2088, 2108
		<code>\use:n</code> .....	3295, 3594
		<code>\usepackage</code> .....	5013, 5023
		<code>\usepgfmodule</code> .....	2
		V	
		vbox commands:	
		<code>\vbox:n</code> .....	63
		<code>\vbox_set_top:Nn</code> .....	834
		<code>\vbox_to_ht:nn</code> .....	384, 411, 1745, 1758
		<code>\vbox_to_zero:n</code> .....	836
		<code>\vcenter</code> .....	1335, 1743, 5248
		<code>\Vdots</code> .....	1068, 2997, 3000
		<code>\vdots</code> .....	1003, 1060
		<code>\Vdotsfor</code> .....	1075
		<code>\vfill</code> .....	387, 413
		<code>\vline</code> .....	108



<code>\VNiceMatrix</code> .....	4817	<code>\vtop</code> .....	954
<code>\vNiceMatrix</code> .....	4814		
<code>\vrule</code> .....	106		X
<code>\vskip</code> .....	105	<code>\xglobal</code> .....	782, 789, 1990, 2033

## 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>17</b>
9.1	The option <code>nullify-dots</code> .....	18
9.2	The commands <code>\Hdotsfor</code> and <code>\Vdotsfor</code> .....	19
9.3	How to generate the continuous dotted lines transparently .....	20
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-after</b>	<b>22</b>
<b>11</b>	<b>The notes in the tabulars</b>	<b>22</b>
11.1	The footnotes .....	22
11.2	The notes of tabular .....	22
11.3	Customisation of the tabular notes .....	24
11.4	Use of <code>{NiceTabular}</code> with <code>threeparttable</code> .....	26

<b>12</b>	<b>Other features</b>	<b>26</b>
12.1	Use of the column type S of siunitx . . . . .	26
12.2	Alignment option in {NiceMatrix} . . . . .	27
12.3	The command \rotate . . . . .	27
12.4	The option small . . . . .	27
12.5	The counters iRow and jCol . . . . .	28
12.6	The option light-syntax . . . . .	29
12.7	Color of the delimiters . . . . .	29
12.8	The environment {NiceArrayWithDelims} . . . . .	29
<b>13</b>	<b>Use of Tikz with nicematrix</b>	<b>29</b>
13.1	The nodes corresponding to the contents of the cells . . . . .	29
13.2	The “medium nodes” and the “large nodes” . . . . .	30
13.3	The “row-nodes” and the “col-nodes” . . . . .	31
<b>14</b>	<b>API for the developpers</b>	<b>32</b>
<b>15</b>	<b>Technical remarks</b>	<b>33</b>
15.1	Definition of new column types . . . . .	33
15.2	Diagonal lines . . . . .	34
15.3	The “empty” cells . . . . .	34
15.4	The option exterior-arraycolsep . . . . .	35
15.5	Incompatibilities . . . . .	35
<b>16</b>	<b>Examples</b>	<b>35</b>
16.1	Notes in the tabulars . . . . .	35
16.2	Dotted lines . . . . .	36
16.3	Dotted lines which are no longer dotted . . . . .	38
16.4	Width of the columns . . . . .	38
16.5	How to highlight cells of a matrix . . . . .	39
16.6	Direct use of the Tikz nodes . . . . .	42
<b>17</b>	<b>Implementation</b>	<b>43</b>
<b>18</b>	<b>History</b>	<b>168</b>
	<b>Index</b>	<b>174</b>