

# The package `nicematrix`\*

F. Pantigny  
fpantigny@wanadoo.fr

July 31, 2020

## 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.

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

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).

This package requires and **loads** the packages `l3keys2e`, `xparse`, `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 5.1 of `nicematrix`, at the date of 2020/07/31.

# 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

### 4 The blocks

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` don't create space by itself.

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. The second argument is the content of the block.

In `{NiceTabular}` the content of the block is composed in text mode. In the other environments, it is composed in math mode.

```

\begin{NiceTabular}{cccc}
rose      & tulipe & marguerite & dahlia \\
violette  & \Block{2-2}{\LARGE\color{blue} fleurs} & & souci \\
pervenche & & lys \\
arum      & iris & jacinthe & muguet
\end{NiceTabular}

```

rose	tulipe	marguerite	dahlia
violette			souci
pervenche		<b>fleurs</b>	lys
arum	iris	jacinthe	muguet

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

It’s also possible to use 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.

```

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

```

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

## 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.

### 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>2</sup>

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

First	Second
Peter	
Mary	George

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`.

```
\newcolumntype{I}{!{\vrule}}
```

However, in this case, it is probably more clever to add a command `\OnlyMainNiceMatrix` (cf. p. 29):

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

#### 5.1.2 The command `\cline`

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

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

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

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

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

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

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

<sup>2</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 29.

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

The key `hlines` draws all the horizontal rules and the key `vlines` draws all the vertical rules. In fact, in the environments with delimiters (as `{pNiceMatrix}` or `{bNiceArray}`) the exterior 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*.<sup>3</sup>

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

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

<sup>3</sup>In fact, when the key `hvlines` (or the key `hvlines-except-corners` described just after) is in force, the rules are also not drawn in the virtual blocks delimited by cells relied par dotted lines (cf. p. 18).

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

The key `hvlines-except-corners` draws all the horizontal and vertical rules, *excepted in the blocks* 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.

```
\begin{NiceTabular}{*{6}{c}}%
[hvlines-except-corners,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>4</sup>

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

$\begin{smallmatrix} y \\ x \end{smallmatrix}$	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>4</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>5</sup>. In `nicematrix`, the dotted lines drawn by `\hdottedline` and “:” do likewise.

## 6 The color of the rows and columns

### 6.1 Utilisation 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 and on the other side, some thin white lines appear. The package `nicematrix` provides some tools without those drawbacks.

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

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

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

<sup>5</sup>In fact, this is true only for `\hline` and “|” but not for `\cline`: cf p. 5



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

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

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

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

a	b	c
e	f	g
h	i	j

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

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

a	b	c
e	f	g
h	i	j

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

```
\begin{NiceArray}{l11}[hvlines, code-before = \rowcolor{red!15}{1,3-5,8-}]
a_1 & b_1 & c_1 \\
a_2 & b_2 & c_2 \\
a_3 & b_3 & c_3 \\
a_4 & b_4 & c_4 \\
a_5 & b_5 & c_5 \\
a_6 & b_6 & c_6 \\
a_7 & b_7 & c_7 \\
a_8 & b_8 & c_8 \\
a_9 & b_9 & c_9 \\
a_{10} & b_{10} & c_{10} \\
\end{NiceArray}
```

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

- The command `\columncolor` takes its name from the command `\columncolor` of `colortbl`. Its syntax is similar to the syntax of `\rowcolor`.
- The command `\rowcolors` (with a  $s$ ) takes its name from the command `\rowcolors` of `xcolor`<sup>6</sup>. The  $s$  emphasizes the fact that there is *two* colors. This command colors alternately the rows of the tabular, beginning with the row whose number is given in first (mandatory) argument. The two other (mandatory) arguments are the colors.

---

<sup>6</sup>The command `\rowcolors` of `xcolor` is available when `xcolor` is loaded with the option `table`.

```

\begin{NiceTabular}{lr}[hlines,code-before = \rowcolors{1}{blue!10}{}]
John & 12 \\
Stephen & 8 \\
Sarah & 18 \\
Ashley & 20 \\
Henry & 14 \\
Madison & 15
\end{NiceTabular}

```

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

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

```

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

```

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

We have used the key `r` which aligns all the columns rightwards (cf. p. 23).

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>7</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`<sup>8</sup>.

<sup>7</sup>As of now, this key is not available in `\NiceMatrixOptions`.

<sup>8</sup>Unlike the command `\columncolor` of `colortbl`, this command `\columncolor` can appear within another command, itself used in the preamble.

```

\begin{NiceTabular}[colortbl-like]{>{\columncolor{blue!15}}ccc}
\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).

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>9</sup>

```

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

```

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

---

<sup>9</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).

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>10</sup>. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 3).

```
\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}
```

$$\begin{bmatrix} 9 & 17 \\ -2 & 5 \end{bmatrix} \quad \begin{bmatrix} 1 & 1245345 \\ 345 & 2 \end{bmatrix}$$

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

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

The dotted lines have been drawn with the tools presented p. 14.

<sup>10</sup>At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

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. 25) 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}}
$\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col,nullify-dots]
    & C_1 & & \Cdots & & C_4 & & \\
L_1 & & a_{11} & & a_{12} & & a_{13} & & a_{14} & & L_1 \\
\vdots & & a_{21} & & a_{22} & & a_{23} & & a_{24} & & \vdots \\
\hline
    & & a_{31} & & a_{32} & & a_{33} & & a_{34} & & \\
L_4 & & a_{41} & & a_{42} & & a_{43} & & a_{44} & & L_4 \\
    & & C_1 & & \Cdots & & C_4 & & \\
\end{pNiceArray}$
```

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

#### Remarks

- As shown in the previous example, the horizontal and vertical rules doesn’t extend in the exterior rows and columns.  
However, if one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 29.
- 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.

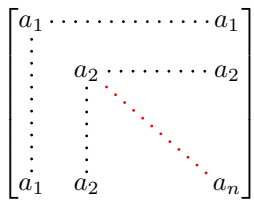
- Logically, the potential option `columns-width` (described p. 11) doesn't apply to the “first column” and “last column”.
- For technical reasons, it's not possible to use the option of the command `\` after the “first row” or before the “last row” (the placement of the delimiters would be wrong).

## 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>11</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>12</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>13</sup>

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



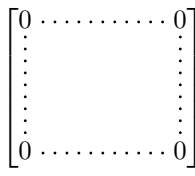
In order to represent the null matrix, one can use the following code:

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



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

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



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

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

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



<sup>11</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>12</sup>The precise definition of a “non-empty cell” is given below (cf. p. 31).

<sup>13</sup>It's also possible to change the color of all these dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 17.

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\l` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.<sup>14</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}
```

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

## 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.

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

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

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).

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

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

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

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

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

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

<sup>14</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. 11

```

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

```

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

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

```

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

```

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

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

The package `nicematrix` also provides a command `\Vdotsfor` similar to `\Hdotsfor` but for the vertical dotted lines. The following example uses both `\Hdotsfor` and `\Vdotsfor`:

```

 $\begin{bNiceMatrix}$ 
C[a_1,a_1] & \Cdots & C[a_1,a_n] \\
& \hspace*{20mm} & C[a_1,a_1^{(p)}] & \Cdots & C[a_1,a_n^{(p)}] \\
\Vdots & \Ddots & \Vdots \\
& \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}

```

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

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

The package `nicematrix` provides an option called `transparent` for using existing code transparently in the environments of the `amsmath` : `{matrix}`, `{pmatrix}`, `{bmatrix}`, etc. In fact, this option is an alias for the conjunction of two options: `renew-dots` and `renew-matrix`.<sup>15</sup>

<sup>15</sup>The options `renew-dots`, `renew-matrix` and `transparent` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage` (it's an exception for these three specific options.)



- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>11</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 option `transparent`, a classical code gives directly the output of `nicematrix`.

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

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

## 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. 19) 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. 19) may be customized by three options (specified between square brackets after the command):

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

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

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

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

### The option `xdots/color`

The option `xdots/color` fixes the color of the dotted line. However, one should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. p. 12.

### 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>16</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 & \\
\Vdots & & & & & & & 0      & \\
0      & \Cdots & & & & & & b      & \\
\end{pNiceMatrix}
```

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

## 9.6 The dotted lines and the key `hvlines`

We have said (cf. p. 6) that the key `hvlines` draws all the horizontal and vertical rules, excepted in the blocks. In fact, when this key is in force, the rules are also not drawn in the virtual blocks delimited by cells relied par dotted lines.

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

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

## 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>17</sup>

<sup>16</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.

<sup>17</sup>There is also a key `code-before` described p. 8.

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}
```

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

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`<sup>18</sup>. For an example, cf. p. 36.

## 11 The notes in the tabulars

### 11.1 The footnotes

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

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

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

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

### 11.2 The notes of tabular

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

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

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

<sup>18</sup>In some circonsstancies, one must put `\omit \CodeAfter`. `\omit` is a keyword of TeX which cancels the pattern of the current cell.

```
\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`).
- 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. 21. This table has been composed with the following code.

```
\begin{table}
\setlength{\belowcaptionskip}{1ex}
\centering
\caption{Utilisation of \texttt{\textbackslash tabularnote}\tabularnote{It's possible
to put a note in the caption.}}
\label{t:tabularnote}
\begin{NiceTabular}{@{}llc@{}}[notes/bottomrule]
\toprule
Last name & First name & Length of life \\
\midrule
Churchill & Wiston & 91\\
Nightingale\tabularnote{Considered as the first nurse of
history.}\tabularnote{Nicknamed ``the Lady with the Lamp''.}
& Florence & 90 \\
Schoelcher & Victor & 89\tabularnote{The label of the note is overlapping.}\\
Touchet & Marie & 89 \\
Wallis & John & 87 \\
\bottomrule
\end{NiceTabular}
\end{table}
```

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

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

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

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

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

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

### 11.3 Customisation of the tabular notes

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

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

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

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

We detail these keys.

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

Initial value: `false`

That key is also available within a given environment.

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

Initial value: `false`

That key is also available within a given environment.

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

Initial value: `\emph{\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. 21).

- 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. 31.

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

If you wish to use the environment `{NiceTabular}` in an environment `{threeparttable}` of the eponymous package, you have to patch the environment `{threeparttable}` with the following code:

```
\makeatletter
\AtBeginEnvironment{threeparttable}{\TPT@hookin{NiceTabular}}
\makeatother
```

The command `\AtBeginEnvironment` is a command of the package `etoolbox` which must have been loaded previously.

## 12 Other features

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

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

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

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

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

### 12.2 Alignment option in `{NiceMatrix}`

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

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

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

There is also a key `S` which sets all the columns all type `S` of `siunitx` (if this package is loaded).<sup>19</sup>

### 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° in the direct sens.

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

<sup>19</sup>This is a part of the functionality provided by the environments `{pmatrix*}`, `{bmatrix*}`, etc. of `mathtools`.

```

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

```

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

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`).

```

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

```

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

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>20</sup>. Of course, the user must not

<sup>20</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.



change the value of these counters which are used internally by `nicematrix`.

In the `code-before` (cf. p. 8) and in the `code-after` (cf. p. 18), `iRow` represents the total number of rows (excepted the potential exterior rows) and `jCol` represents the total number of columns (excepted the potential exterior columns).

```

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

```

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

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  $n-p$  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).

```

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

```

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

## 12.6 The option `light-syntax`

The option `light-syntax` (inpired 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.

The following example has been composed with XeLaTeX with `unicode-math`, which allows the use of greek letters directly in the TeX source.

```

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

```

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

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>21</sup>

<sup>21</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.

## 12.7 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.

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

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

## 13 Utilisation 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).

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.

```
$\begin{pNiceMatrix}[name=mymatrix]
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{pNiceMatrix}$
\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`, and if Tikz is loaded, the things are easier. One may design the nodes with the form  $i-j$ : there is no need 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. 36).

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

## 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>22</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.

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

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>23</sup>

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

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>24</sup>

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

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.

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

**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 \\[1ex]
noix & noisette & brugnon
\end{NiceTabular}
```

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

<sup>22</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>23</sup>There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 12).

<sup>24</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.

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 utilisation 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`.

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 Tools for the developers

As its name implies, the token list `\g_nicematrix_code_after_tl` is public (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).

This variable contains the code of what we have called the “code-after”. The developer can use it to add code from a cell of the array (the affectation must be global, allowing to exit the cell, which is a TeX group).

*Example :* We want to write a command `\crossbox` to draw a cross in the current cell. This command will take in an optional argument between square brackets for a list of pairs *key-value* which will be given to Tikz before the drawing.

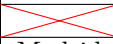
It’s possible to program such command `\crossbox` as follows, explicitly using the public variable `\g_nicematrix_code_after_tl`.

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

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

Here is an example of utilisation:

```
\begin{NiceTabular}{ccc}[hvlines]
Tokyo & Paris & London \\
Cap Town & \crossbox[red] & Miami \\
Los Angeles & Madrid & Roma
\end{NiceTabular}
```

Tokyo	Paris	London
Cap Town		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>25</sup>:

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

The heavy vertical rule won't extend in the exterior rows.<sup>26</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}$
```

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

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

## 15.2 Diagonal lines

By default, all the diagonal lines<sup>27</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}$
```

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

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

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

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

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

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

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

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

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

The same example without parallelization:

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

<sup>25</sup>The command \vrule is a TeX (and not LaTeX) command.

<sup>26</sup>Of course, such rule is defined by the classical technics of nicematrix and, for this reason, won't cross the double rules of \hline\hline.

<sup>27</sup>We speak of the lines created by \Ddots and not the lines created by a command \line in code-after.

### 15.3 The “empty” cells

An instruction like `\Ldots`, `\Cdots`, etc. tries to determine the first non-empty cells 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 `&`). Indeed, a cell which only contains `\hspace*{1cm}` may be considered as empty.

For `nicematrix`, 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 with a command `\Hspace` (or `\Hspace*`) is empty. This command `\Hspace` is a command defined by the package `nicematrix` with the same meaning as `\hspace` except that the cell where it is used is considered as empty. This command can be used to fix the width of some columns of the matrix without interfering with `nicematrix`.

### 15.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea<sup>28</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>29</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. 19.

---

<sup>28</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>29</sup>And not by inserting `@{}` on both sides of the preamble of the array. As a consequence, the length of the `\hline` is not modified and may appear too long, in particular when using square brackets

Let's consider that we wish to number the notes of a tabular with stars.<sup>30</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>31</sup>

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

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

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

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

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

\*Achard is an old family of the Poitou.

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

<sup>30</sup>Of course, it's realistic only when there is very few notes in the tabular.

<sup>31</sup>In fact: the value of its argument.



## 16.2 Dotted lines

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

```

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

```

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

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 \\
 $\end{pNiceMatrix}$ 

```

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

An example with `\multicolumn`:

```

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

```

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

An example with `\Hdotsfor`:

```

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

```

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

An example for the resultant of two polynoms:

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

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

An example for a linear system:

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

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

### 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{pNiceMatrix}[first-row,first-col]
& & \Ldots[line-style={solid,<->},shorten=0pt]^{n \text{ columns}} \\
& 1 & 1 & 1 & \Ldots & 1 \\
& 1 & 1 & 1 & & 1
\end{pNiceMatrix}
```

```

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

```

$$\begin{array}{c} \text{\textcolor{blue}{\scriptsize $n$ rows}} \end{array} \left( \begin{array}{c} \text{\textcolor{blue}{\scriptsize $n$ columns}} \\ \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix} \end{array} \right)$$

## 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}

```

$$\begin{array}{c} \begin{pmatrix} 1 & 1 & 1 & 1 & \vdots & 1 \\ 2 & 4 & 8 & 16 & \vdots & 9 \\ 3 & 9 & 27 & 81 & \vdots & 36 \\ 4 & 16 & 64 & 256 & \vdots & 100 \end{pmatrix} \end{array} \quad \left| \quad \begin{array}{c} \begin{pmatrix} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 1 & 3 & 7 & \vdots & \frac{7}{2} \\ 0 & 0 & 3 & 18 & \vdots & 6 \\ 0 & 0 & -2 & -14 & \vdots & -\frac{9}{2} \end{pmatrix} \begin{array}{l} L_3 \leftarrow -3L_2 + L_3 \\ L_4 \leftarrow L_2 - L_4 \end{array} \end{array}$$

$$\begin{array}{c} \begin{pmatrix} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 2 & 6 & 14 & \vdots & 7 \\ 0 & 6 & 24 & 78 & \vdots & 33 \\ 0 & 12 & 60 & 252 & \vdots & 96 \end{pmatrix} \begin{array}{l} L_2 \leftarrow -2L_1 + L_2 \\ L_3 \leftarrow -3L_1 + L_3 \\ L_4 \leftarrow -4L_1 + L_4 \end{array} \end{array} \quad \left| \quad \begin{array}{c} \begin{pmatrix} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 1 & 3 & 7 & \vdots & \frac{7}{2} \\ 0 & 0 & 1 & 6 & \vdots & 2 \\ 0 & 0 & -2 & -14 & \vdots & -\frac{9}{2} \end{pmatrix} \begin{array}{l} L_3 \leftarrow \frac{1}{3}L_3 \end{array} \end{array}$$

$$\begin{array}{c} \begin{pmatrix} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 1 & 3 & 7 & \vdots & \frac{7}{2} \\ 0 & 3 & 12 & 39 & \vdots & \frac{33}{2} \\ 0 & 1 & 5 & 21 & \vdots & 8 \end{pmatrix} \begin{array}{l} L_2 \leftarrow \frac{1}{2}L_2 \\ L_3 \leftarrow \frac{1}{2}L_3 \\ L_4 \leftarrow \frac{1}{12}L_4 \end{array} \end{array} \quad \left| \quad \begin{array}{c} \begin{pmatrix} 1 & 1 & 1 & 1 & \vdots & 1 \\ 0 & 1 & 3 & 7 & \vdots & \frac{7}{2} \\ 0 & 0 & 1 & 6 & \vdots & 2 \\ 0 & 0 & 0 & -2 & \vdots & -\frac{1}{2} \end{pmatrix} \begin{array}{l} L_4 \leftarrow L_3 + L_4 \end{array} \end{array}$$

## 16.5 How to highlight cells of the matrix

The following examples 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}
```

In order to highlight a cell of a matrix, it's possible to “draw” one of the correspondant nodes (the “normal node”, the “medium node” or the “large node”). In the following example, we use the “large nodes” of the diagonal of the matrix (with the Tikz key “`name suffix`”, it's easy to use the “large nodes”).

We redraw the nodes with other nodes by using the Tikz library `fit`. Since we want to redraw the nodes exactly, we have to set `inner sep = 0 pt` (if we don't do that, the new nodes will be larger than the nodes created by `nicematrix`).

```
$\begin{pNiceArray}{>{\strut}cccc}[create-large-nodes,margin,extra-margin = 2pt]
  a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{21} & a_{22} & a_{23} & a_{24} \\
  a_{31} & a_{32} & a_{33} & a_{34} \\
  a_{41} & a_{42} & a_{43} & a_{44} \\
\CodeAfter
  \begin{tikzpicture}[name suffix = -large,
    every node/.style = {draw,inner sep = 0 pt}]
    \node [fit = (1-1)] {};
    \node [fit = (2-2)] {};
    \node [fit = (3-3)] {};
    \node [fit = (4-4)] {};
  \end{tikzpicture}
\end{pNiceArray}$
```

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

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the command `\hline`, the specifier “|” and the options `hlines`, `vlines` and `hvlines` spread the cells.<sup>32</sup>

It's possible to color a row with `\rowcolor` in the `code-before` (or with `\rowcolor` of `colortbl` in the first cell of the row). However, it's not possible to do a fine tuning. That's why we describe now 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`.

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

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

<sup>32</sup>For the command `\cline`, see the remark p. 5.

```

1 & \Cdots & 1 \\
0 & \Cdots & 0
\end{bNiceMatrix}$

```

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

This code fails with `latex-dvips-ps2pdf` because Tikz for `dvips`, as for now, doesn't support blend modes. However, the following code, in the preamble, should activate blend modes in this way of compilation.

```

\ExplSyntaxOn
\makeatletter
\tl_set:Nn \l_tmpa_tl {pgfsys-dvips.def}
\tl_if_eq:NNT \l_tmpa_tl \pgfsysdriver
{ \cs_set:Npn \pgfsys@blend@mode#1 { \special{ps:~/\tl_upper_case:n #1~.setblendmode}}}
\makeatother
\ExplSyntaxOff

```

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`.

```

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

```

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

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

```

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

```

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

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

```

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

```

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

We obtain the following matrix.

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

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} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44} \\
\CodeAfter
\tikz \path [name suffix = -large,fill = red!15,blend mode = multiply]
(1-1.north west)
|- (2-2.north west)
|- (3-3.north west)
|- (4-4.north west)
|- (4-4.south east)
|- (1-1.north west) ;
\end{pNiceArray}
```

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

## 16.6 Direct use of the Tikz nodes

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

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

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

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

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

```

 $\begin{array}{cc}$ 
&

```

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

```

\begin{bNiceArray}{c>\strut}cccc[name=B,first-row]
& & C_j & & \\
b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\
\vdots & & \vdots & & \vdots \\
& & b_{kj} & & \\
& & \vdots & & \\
b_{n1} & \cdots & b_{nj} & \cdots & b_{nn} \\
\end{bNiceArray} \quad \quad

```

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

```

\begin{bNiceArray}{cc>\strut}ccc[name=A,first-col]
& a_{11} & \cdots & & a_{1n} \\
& \vdots & & & \vdots \\
L_i & a_{i1} & \cdots & a_{ik} & \cdots & a_{in} \\
& \vdots & & & \vdots \\
& a_{n1} & \cdots & & a_{nn} \\
\end{bNiceArray}
&

```

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

```

\begin{bNiceArray}{cc>\strut}ccc
& & & \\
& & \vdots & \\
\cdots & & c_{ij} & \\
\\
\\
\end{bNiceArray}
\end{array}$

```

```
\end{NiceMatrixBlock}
```

```

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

```

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

## 17 Implementation

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

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

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

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

### Declaration of the package and packages loaded

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

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

`<@@=nicematrix>`

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

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

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

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

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

We load some packages.

```
9 \RequirePackage { array }
10 \RequirePackage { amsmath }
11 \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` of `xparse`, 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.

```

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:nNnT \baselineskip = \c_zero_dim \noalign
93     { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }
94   }
95   \cs_set:Npn \hline
96   {
97     \noalign { \ifnum 0 = ` } \fi
98     \cs_set_eq:NN \hskip \vskip
99     \cs_set_eq:NN \vrule \hrule
100    \cs_set_eq:NN \@width \@height
101    { \CT@arc@ \vline }
102    \futurelet \reserved@a
103    \@xhline
104  }
105 }
106 }

```

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.

```

107 \cs_set:Npn \@_standard_cline #1 { \@_standard_cline:w #1 \q_stop }
108 \cs_set:Npn \@_standard_cline:w #1-#2 \q_stop
109 {
110   \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
111   \int_compare:nNnT { #1 } > 1 { \multispan { \@_pred:n { #1 } } & }
112   \multispan { \int_eval:n { #2 - #1 + 1 } }

```

```

113 { \CT@arc@ \leaders \hrule \@height \arrayrulewidth \hfill }

```

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`.

```

114 \everycr { }
115 \cr
116 \noalign { \skip_vertical:N -\arrayrulewidth }
117 }

```

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

```

118 \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`.

```

119 { \@@_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*.

```

120 \cs_set:Npn \@@_cline_i:nn #1 #2 { \@@_cline_i:w #1-#2 \q_stop }
121 \cs_set:Npn \@@_cline_i:w #1-#2-#3 \q_stop
122 {

```

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).

```

123 \int_compare:nNnT { #1 } < { #2 }
124 { \multispan { \int_eval:n { #2 - #1 } } & }
125 \multispan { \int_eval:n { #3 - #2 + 1 } }
126 { \CT@arc@ \leaders \hrule \@height \arrayrulewidth \hfill }

```

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

```

127 \peek_meaning_remove_ignore_spaces:NTF \cline
128 { & \@@_cline_i:en { \@@_succ:n { #3 } } }
129 { \everycr { } \cr }
130 }
131 \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.

```

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

```

The following command is a small shortcut.

```

134 \cs_new:Npn \@@_math_toggle_token:
135 { \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token }

```

```

136 \cs_new_protected:Npn \@@_set_CT@arc@:
137 { \peek_meaning:NTF [ \@@_set_CT@arc@_i: \@@_set_CT@arc@_ii: }
138 \cs_new_protected:Npn \@@_set_CT@arc@_i: [ #1 ] #2 \q_stop
139 { \cs_set:Npn \CT@arc@ { \color [ #1 ] { #2 } } }
140 \cs_new_protected:Npn \@@_set_CT@arc@_ii: #1 \q_stop
141 { \cs_set:Npn \CT@arc@ { \color { #1 } } }

```

```

142 \cs_new:Npn \@@_tab_or_array_colsep:
143 { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }

```

## 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`.

```

144 \bool_new:N \c_@@_siunitx_loaded_bool
145 \AtBeginDocument
146 {

```

```

147 \ifpackageloaded { siunitx }
148   { \bool_set_true:N \c_@@_siunitx_loaded_bool }
149   { }
150 }

```

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:`.

```

151 \cs_set_protected:Npn \@@_adapt_S_column:
152 {
153   \bool_if:NT \c_@@_siunitx_loaded_bool
154   {
155     \group_begin:
156     \@temptokena = { }

```

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

```

157   \cs_set_eq:NN \NC@find \prg_do_nothing:
158   \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`).

```

159   \tl_gset:NV \g_tmpa_tl \@temptokena
160   \group_end:
161   \tl_new:N \c_@@_table_collect_begin_tl
162   \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
163   \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
164   \tl_new:N \c_@@_table_print_tl
165   \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).

```

166     \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
167   }
168 }

```

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

```

169 \AtBeginDocument
170 {
171   \bool_if:nTF { ! \c_@@_siunitx_loaded_bool }
172   { \cs_set_eq:NN \@@_renew_NC@rewrite@S: \prg_do_nothing: }
173   {
174     \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
175     {
176       \renewcommand*{\NC@rewrite@S}[1] []
177       {
178         \@temptokena \exp_after:wN
179         {
180           \tex_the:D \@temptokena
181           > { \@@_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.

```

182         \@@_true_c:
183         < { \c_@@_table_print_tl \@@_end_Cell: }
184       }
185     \NC@find
186   }
187 }
188 }
189 }

```

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

```

190 \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.

```

191 \cs_new_protected:Npn \@@_provide_pgfsyspdfmark:
192 {
193   \iow_now:Nn \@mainaux
194   {
195     \ExplSyntaxOn
196     \cs_if_free:NT \pgfsyspdfmark
197     { \cs_set_eq:NN \pgfsyspdfmark \use:nnn }
198     \ExplSyntaxOff
199   }
200   \cs_gset_eq:NN \@@_provide_pgfsyspdfmark: \prg_do_nothing:
201 }

```

## 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.

```

202 \bool_new:N \c_@@_define_L_C_R_bool

```

```

203 \cs_new_protected:Npn \@@_define_L_C_R:
204 {
205     \newcolumntype L l
206     \newcolumntype C c
207     \newcolumntype R r
208 }

```

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.

```

209 \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).

```

210 \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.

```

211 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
212 { \int_use:N \g_@@_env_int }

```

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

```

213 \cs_new_protected:Npn \@@_qpoint:n #1
214 { \pgfpointanchor { \@@_env: - #1 } { center } }

```

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

```

215 \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).

```

216 \dim_new:N \l_@@_columns_width_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`.

```

217 \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.

```

218 \bool_new:N \l_@@_in_env_bool

```

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

```

219 \bool_new:N \l_@@_NiceArray_bool

```

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

```

220 \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.

```

221 \dim_new:N \l_@@_tabular_width_dim

```

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

```

222 \bool_new:N \l_@@_Matrix_bool

223 \cs_new_protected:Npn \@@_test_if_math_mode:
224 {
225   \if_mode_math: \else:
226     \@@_fatal:n { Outside~math~mode }
227   \fi:
228 }
```

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

```

229 \colorlet { nicematrix-last-col } { . }
230 \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*).

```

231 \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*.

```

232 \str_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.

```

233 \cs_new:Npn \@@_full_name_env:
234 {
235   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
236     { command \space \c_backslash_str \g_@@_name_env_str }
237     { environment \space \{ \g_@@_name_env_str \} }
238 }
```

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

```

239 \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.

```

240 \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.

```

241 \int_new:N \l_@@_old_iRow_int
242 \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.

```

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

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).

```
244 \bool_new:N \g_@@_row_of_col_done_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 explicetely add material in `\l_@@_code_before_tl` by using the key `code-before`.

```
245 \tl_new:N \l_@@_code_before_tl
246 \bool_new:N \l_@@_code_before_bool
```

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

```
247 \dim_new:N \l_@@_x_initial_dim
248 \dim_new:N \l_@@_y_initial_dim
249 \dim_new:N \l_@@_x_final_dim
250 \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`).

```
251 \dim_zero_new:N \l_tmpc_dim
252 \dim_zero_new:N \l_tmpd_dim
```

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

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

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

```
254 \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”.

```
255 \dim_new:N \g_@@_width_last_col_dim
256 \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.

```
257 \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}`.

```
258 \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 required by the keys `hvlines` or `hvlines-except-corners`.

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}`.

```
259 \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).

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).

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

Used for the color of the blocks.

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

## 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.

```
262 \int_new:N \l_@@_first_row_int
263 \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.

```
264 \int_new:N \l_@@_first_col_int
265 \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).

```
266 \int_new:N \l_@@_last_row_int
267 \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>33</sup>

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

Idem for `\l_@@_last_col_without_value_bool`

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

---

<sup>33</sup>We can't use `\l_@@_last_row_int` for this usage because, if `nicematrix` has read its value from the `aux` file, the value of the counter won't be `-1` any longer.

- **Last column**

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

```
270 \int_new:N \l_@@_last_col_int
271 \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.

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

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

## The command `\tablarnote`

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

```
273 \newcounter { tablarnote }
```

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

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

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

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

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

```
276 \cs_new:Npn \@@_notes_style:n #1 { \emph { \alph { #1 } } }
```

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

```
277 \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`.

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

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

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

The tabular notes will be available for the final user only when `enumitem` is loaded. Indeed, the tabular notes will be composed at the end of the array with a list customized by `enumitem` (a list `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`).

```

280 \AtBeginDocument
281 {
282   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
283   {
284     \NewDocumentCommand \tabularnote { m }
285     { \@@_error:n { enumitem~not~loaded } }
286   }
287   {

```

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.

```

288   \newlist { tabularnotes } { enumerate } { 1 }
289   \setlist [ tabularnotes ]
290   {
291     noitemsep , leftmargin = * , align = left , labelsep = 0pt ,
292     label =
293     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
294   }
295   \newlist { tabularnotes* } { enumerate* } { 1 }
296   \setlist [ tabularnotes* ]
297   {
298     afterlabel = \nobreak ,
299     itemjoin = \quad ,
300     label =
301     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
302   }

```

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>34</sup>

```

303   \NewDocumentCommand \tabularnote { m }
304   {
305     \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
306     { \@@_error:n { tabularnote~forbidden } }
307     {

```

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

```

308     \int_incr:N \l_@@_number_of_notes_int

```

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

```

309     \seq_gput_right:Nx \g_@@_tabularnotes_seq { #1 }
310     \peek_meaning:NF \tabularnote
311     {

```

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.

```

312     \hbox_set:Nn \l_tmpa_box
313     {

```

---

<sup>34</sup>We should try to find a solution to that problem.

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

```

314         \@@_notes_label_in_tabular:n
315         {
316             \stepcounter { tabularnote }
317             \@@_notes_style:n { tabularnote }
318             \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
319             {
320                 ,
321                 \stepcounter { tabularnote }
322                 \@@_notes_style:n { tabularnote }
323             }
324         }
325     }

```

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.

```

326         \addtocounter { tabularnote } { -1 }
327         \refstepcounter { tabularnote }
328         \int_zero:N \l_@@_number_of_notes_int
329         \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).

```

330         \skip_horizontal:n { \box_wd:N \l_tmpa_box }
331     }
332 }
333 }
334 }
335 }

```

## 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.

```

336 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
337 {
338     \begin { pgfscope }
339     \pgfset
340     {
341         outer~sep = \c_zero_dim ,
342         inner~sep = \c_zero_dim ,
343         minimum~size = \c_zero_dim
344     }
345     \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
346     \pgfnode
347     { rectangle }
348     { center }
349     {
350         \vbox_to_ht:nn
351         { \dim_abs:n { #5 - #3 } }
352         {
353             \vfill
354             \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
355         }
356     }
357     { #1 }
358     { }

```

```

359   \end { pgfscope }
360 }

```

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

```

361 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
362 {
363   \begin { pgfscope }
364   \pgfset
365   {
366     outer~sep = \c_zero_dim ,
367     inner~sep = \c_zero_dim ,
368     minimum~size = \c_zero_dim
369   }
370   \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
371   \pgfpointdiff { #3 } { #2 }
372   \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
373   \pgfnode
374   { rectangle }
375   { center }
376   {
377     \vbox_to_ht:nn
378     { \dim_abs:n \l_tmpb_dim }
379     { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
380   }
381   { #1 }
382   { }
383   \end { pgfscope }
384 }

```

## 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.

```

385 \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`.

```

386 \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`).

```

387 \dim_new:N \l_@@_cell_space_top_limit_dim
388 \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.

```

389 \dim_new:N \l_@@_inter_dots_dim
390 \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).

```

391 \dim_new:N \l_@@_xdots_shorten_dim
392 \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.

```
393 \dim_new:N \l_@@_radius_dim
394 \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.

```
395 \tl_new:N \l_@@_xdots_line_style_tl
396 \tl_const:Nn \c_@@_standard_tl { standard }
397 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

```
398 \bool_new:N \l_@@_light_syntax_bool
```

The string `\l_@@_baseline_str` 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).

```
399 \str_new:N \l_@@_baseline_str
400 \str_set:Nn \l_@@_baseline_str 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`).

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

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

```
402 \bool_new:N \l_@@_parallelize_diags_bool
403 \bool_set_true:N \l_@@_parallelize_diags_bool
```

The flag `\l_@@_hlines_bool` corresponds to the key `hlines`, the flag `\l_@@_vlines_bool` to the key `vlines` and the flag `hvlines` to the key `hvlines`. The key `hvlines` is not a mere alias for the conjunction of `hlines` and `vlines`. Indeed, with `hvlines`, the vertical and horizontal rules are *not* drawn within the blocks (created by `\Block`).

```
404 \bool_new:N \l_@@_hlines_bool
405 \bool_new:N \l_@@_vlines_bool
406 \bool_new:N \l_@@_hvlines_bool
```

The flag `\l_@@_hlines_except_corners_bool` corresponds to the key `hlines-except-corners`.

```
407 \bool_new:N \l_@@_hvlines_except_corners_bool
408 \dim_new:N \l_@@_notes_above_space_dim
409 \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.).

```
410 \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).

```
411 \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.

```
412 \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”.

```
413 \bool_new:N \l_@@_medium_nodes_bool
```

```
414 \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`).

```
415 \dim_new:N \l_@@_left_margin_dim
```

```
416 \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`.

```
417 \dim_new:N \l_@@_extra_left_margin_dim
```

```
418 \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.

```
419 \tl_new:N \l_@@_end_of_row_tl
```

```
420 \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 “:”.

```
421 \tl_new:N \l_@@_xdots_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.

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

```
423 \keys_define:nn { NiceMatrix / xdots }
```

```
424 {
```

```
425   line-style .code:n =
```

```
426   {
```

```
427     \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.

```
428   { \cs_if_exist_p:N \tikzpicture }
```

```
429   { \str_if_eq_p:nn { #1 } { standard } }
```

```
430   { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
```

```
431   { \@@_error:n { bad-option-for-line-style } }
```

```
432   } ,
```

```
433   line-style .value_required:n = true ,
```

```
434   color .tl_set:N = \l_@@_xdots_color_tl ,
```

```
435   color .value_required:n = true ,
```

```
436   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
```

```
437   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 `_`.

```

438   down .tl_set:N = \l_@@_xdots_down_tl ,
439   up .tl_set:N = \l_@@_xdots_up_tl ,
440   unknown .code:n = \@@_error:n { Unknown-option-for-~xdots }
441 }

```

The following keys are for the tabular notes (specified by the command `\tabularnote` inside `{NiceTabular}` and `{NiceArray}`).

```

442 \keys_define:nn { NiceMatrix / rules }
443 {
444   color .tl_set:N = \l_@@_rules_color_tl ,
445   color .value_required:n = true ,
446   width .dim_set:N = \arrayrulewidth ,
447   width .value_required:n = true
448 }

```

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

```

449 \keys_define:nn { NiceMatrix / Global }
450 {
451   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
452   standard-cline .default:n = true ,
453   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
454   cell-space-top-limit .value_required:n = true ,
455   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
456   cell-space-bottom-limit .value_required:n = true ,
457   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
458   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
459   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
460   light-syntax .default:n = true ,
461   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
462   end-of-row .value_required:n = true ,
463   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
464   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
465   last-row .int_set:N = \l_@@_last_row_int ,
466   last-row .default:n = -1 ,
467   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
468   code-for-first-col .value_required:n = true ,
469   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
470   code-for-last-col .value_required:n = true ,
471   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
472   code-for-first-row .value_required:n = true ,
473   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
474   code-for-last-row .value_required:n = true ,
475   hlines .bool_set:N = \l_@@_hlines_bool ,
476   vlines .bool_set:N = \l_@@_vlines_bool ,
477   hvlines .code:n =
478   {
479     \bool_set_true:N \l_@@_hvlines_bool
480     \bool_set_true:N \l_@@_vlines_bool
481     \bool_set_true:N \l_@@_hlines_bool
482   } ,
483   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.

```

484   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
485   renew-dots .value_forbidden:n = true ,

```



```

486 nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
487 create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
488 create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
489 create-extra-nodes .meta:n =
490   { create-medium-nodes , create-large-nodes } ,
491 left-margin .dim_set:N = \l_@@_left_margin_dim ,
492 left-margin .default:n = \arraycolsep ,
493 right-margin .dim_set:N = \l_@@_right_margin_dim ,
494 right-margin .default:n = \arraycolsep ,
495 margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
496 margin .default:n = \arraycolsep ,
497 extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
498 extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
499 extra-margin .meta:n =
500   { extra-left-margin = #1 , extra-right-margin = #1 } ,
501 extra-margin .value_required:n = true ,
502 }

```

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

```

503 \keys_define:nn { NiceMatrix / Env }
504 {
505   hvlines-except-corners .bool_set:N = \l_@@_hvlines_except_corners_bool ,
506   hvlines-except-corners .default:n = true ,
507   code-before .code:n =
508     {
509       \tl_if_empty:nF { #1 }
510       {
511         \tl_put_right:Nn \l_@@_code_before_tl { #1 }
512         \bool_set_true:N \l_@@_code_before_bool
513       }
514     } ,

```

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

```

515 c .code:n = \str_set:Nn \l_@@_baseline_str c ,
516 t .code:n = \str_set:Nn \l_@@_baseline_str t ,
517 b .code:n = \str_set:Nn \l_@@_baseline_str b ,
518 baseline .tl_set:N = \l_@@_baseline_str ,
519 baseline .value_required:n = true ,
520 columns-width .code:n =
521   \str_if_eq:nnTF { #1 } { auto }
522     { \bool_set_true:N \l_@@_auto_columns_width_bool }
523     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
524 columns-width .value_required:n = true ,
525 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.

```

526 \legacy_if:nF { measuring@ }
527 {
528   \str_set:Nn \l_tmpa_str { #1 }
529   \seq_if_in:NVTf \g_@@_names_seq \l_tmpa_str
530     { \@@_error:nn { Duplicate-name } { #1 } }
531     { \seq_gput_left:Nv \g_@@_names_seq \l_tmpa_str }
532   \str_set_eq:NN \l_@@_name_str \l_tmpa_str
533 } ,
534 name .value_required:n = true ,
535 code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
536 code-after .value_required:n = true ,
537 colortbl-like .code:n =
538   \bool_set_true:N \l_@@_colortbl_like_bool

```

```

539     \bool_set_true:N \l_@@_code_before_bool ,
540     colortbl-like .value_forbidden:n = true
541   }
542 \keys_define:nn { NiceMatrix / notes }
543 {
544   para .bool_set:N = \l_@@_notes_para_bool ,
545   para .default:n = true ,
546   code-before .tl_set:N = \l_@@_notes_code_before_tl ,
547   code-before .value_required:n = true ,
548   code-after .tl_set:N = \l_@@_notes_code_after_tl ,
549   code-after .value_required:n = true ,
550   bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
551   bottomrule .default:n = true ,
552   style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
553   style .value_required:n = true ,
554   label-in-tabular .code:n =
555     \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
556   label-in-tabular .value_required:n = true ,
557   label-in-list .code:n =
558     \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
559   label-in-list .value_required:n = true ,
560   enumitem-keys .code:n =
561     {
562       \bool_if:NTF \c_@@_in_preamble_bool
563       {
564         \AtBeginDocument
565         {
566           \bool_if:NT \c_@@_enumitem_loaded_bool
567             { \setlist* [ tabularnotes ] { #1 } }
568         }
569       }
570       {
571         \bool_if:NT \c_@@_enumitem_loaded_bool
572           { \setlist* [ tabularnotes ] { #1 } }
573       }
574     } ,
575   enumitem-keys .value_required:n = true ,
576   enumitem-keys-para .code:n =
577     {
578       \bool_if:NTF \c_@@_in_preamble_bool
579       {
580         \AtBeginDocument
581         {
582           \bool_if:NT \c_@@_enumitem_loaded_bool
583             { \setlist* [ tabularnotes* ] { #1 } }
584         }
585       }
586       {
587         \bool_if:NT \c_@@_enumitem_loaded_bool
588           { \setlist* [ tabularnotes* ] { #1 } }
589       }
590     } ,
591   enumitem-keys-para .value_required:n = true ,
592   unknown .code:n = \@@_error:n { Unknown~key~for~notes }
593 }

```

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

```

594 \keys_define:nn { NiceMatrix }
595 {
596   NiceMatrixOptions .inherit:n =
597     { NiceMatrix / Global } ,

```

```

598 NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
599 NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
600 NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
601 NiceMatrix .inherit:n =
602 {
603     NiceMatrix / Global ,
604     NiceMatrix / Env ,
605 } ,
606 NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
607 NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
608 NiceTabular .inherit:n =
609 {
610     NiceMatrix / Global ,
611     NiceMatrix / Env
612 } ,
613 NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
614 NiceTabular / rules .inherit:n = NiceMatrix / rules ,
615 NiceArray .inherit:n =
616 {
617     NiceMatrix / Global ,
618     NiceMatrix / Env ,
619 } ,
620 NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
621 NiceArray / rules .inherit:n = NiceMatrix / rules ,
622 pNiceArray .inherit:n =
623 {
624     NiceMatrix / Global ,
625     NiceMatrix / Env ,
626 } ,
627 pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
628 pNiceArray / rules .inherit:n = NiceMatrix / rules ,
629 }

```

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

```

630 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
631 {
632     last-col .code:n = \tl_if_empty:nF { #1 }
633                     { \@@_error:n { last-col-non-empty-for-NiceMatrixOptions } }
634                     \int_zero:N \l_@@_last_col_int ,
635     small .bool_set:N = \l_@@_small_bool ,
636     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.

```

637     renew-matrix .code:n = \@@_renew_matrix: ,
638     renew-matrix .value_forbidden:n = true ,
639     transparent .meta:n = { renew-dots , renew-matrix } ,
640     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}`.

```

641     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.

```

642     columns-width .code:n =
643     \str_if_eq:nnTF { #1 } { auto }
644     { \@@_error:n { Option~auto~for~columns-width } }
645     { \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.

```

646   allow-duplicate-names .code:n =
647     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
648   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`).

```

649   letter-for-dotted-lines .code:n =
650     {
651       \int_compare:nTF { \tl_count:n { #1 } = 1 }
652         { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
653         { \@@_error:n { Bad-value-for-letter-for-dotted-lines } }
654     } ,
655   letter-for-dotted-lines .value_required:n = true ,
656   notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
657   notes .value_required:n = true ,
658   unknown .code:n = \@@_error:n { Unknown-key-for-NiceMatrixOptions }
659 }

660 \str_new:N \l_@@_letter_for_dotted_lines_str
661 \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.

```

662 \NewDocumentCommand \NiceMatrixOptions { m }
663 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

664 \keys_define:nn { NiceMatrix / NiceMatrix }
665 {
666   last-col .code:n = \tl_if_empty:nTF {#1}
667     {
668       \bool_set_true:N \l_@@_last_col_without_value_bool
669       \int_set:Nn \l_@@_last_col_int { -1 }
670     }
671     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
672   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
673   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
674   S .code:n = \bool_if:NTF \c_@@_siunitx_loaded_bool
675     { \tl_set:Nn \l_@@_type_of_col_tl S }
676     { \@@_error:n { option-S-without-siunitx } } ,
677   small .bool_set:N = \l_@@_small_bool ,
678   small .value_forbidden:n = true ,
679   unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
680 }

```

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

```

681 \keys_define:nn { NiceMatrix / NiceArray }
682 {

```

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.

```

683   small .bool_set:N = \l_@@_small_bool ,
684   small .value_forbidden:n = true ,

```

```

685   last-col .code:n = \tl_if_empty:nF { #1 }
686           { \@@_error:n { last-col~non-empty~for~NiceArray } }
687           \int_zero:N \l_@@_last_col_int ,
688   notes / para .bool_set:N = \l_@@_notes_para_bool ,
689   notes / para .default:n = true ,
690   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
691   notes / bottomrule .default:n = true ,
692   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
693 }
694 \keys_define:nn { NiceMatrix / pNiceArray }
695 {
696   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
697   last-col .code:n = \tl_if_empty:nF {#1}
698           { \@@_error:n { last-col~non-empty~for~NiceArray } }
699           \int_zero:N \l_@@_last_col_int ,
700   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
701   small .bool_set:N = \l_@@_small_bool ,
702   small .value_forbidden:n = true ,
703   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
704 }

```

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

```

705 \keys_define:nn { NiceMatrix / NiceTabular }
706 {
707   notes / para .bool_set:N = \l_@@_notes_para_bool ,
708   notes / para .default:n = true ,
709   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
710   notes / bottomrule .default:n = true ,
711   last-col .code:n = \tl_if_empty:nF {#1}
712           { \@@_error:n { last-col~non-empty~for~NiceArray } }
713           \int_zero:N \l_@@_last_col_int ,
714   unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
715 }

```

## 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}`).

```

716 \cs_new_protected:Npn \@@_Cell:
717 {

```

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

```

718   \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.

```

719   \int_compare:nNnT \c@jCol = 1
720   { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
721   \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).

```

722   \hbox_set:Nw \l_@@_cell_box
723   \bool_if:NF \l_@@_NiceTabular_bool
724   {
725     \c_math_toggle_token

```

```

726     \bool_if:NT \l_@@_small_bool \scriptstyle
727 }

```

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.

```

728 \int_compare:nNnTF \c@iRow = 0
729 {
730     \int_compare:nNnT \c@jCol > 0
731     {
732         \l_@@_code_for_first_row_tl
733         \xglobal \colorlet { nicematrix-first-row } { . }
734     }
735 }
736 {
737     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
738     {
739         \l_@@_code_for_last_row_tl
740         \xglobal \colorlet { nicematrix-last-row } { . }
741     }
742 }
743 }

```

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.

```

744 \cs_new_protected:Npn \@@_begin_of_row:
745 {
746     \int_gincr:N \c@iRow
747     \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
748     \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
749     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
750     \pgfpicture
751     \pgfrememberpicturepositiononpagetrue
752     \pgfcoordinate
753     { \@@_env: - row - \int_use:N \c@iRow - base }
754     { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
755     \str_if_empty:NF \l_@@_name_str
756     {
757         \pgfnodealias
758         { \l_@@_name_str - row - \int_use:N \c@iRow - base }
759         { \@@_env: - row - \int_use:N \c@iRow - base }
760     }
761     \endpgfpicture
762 }

```

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.

```

763 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
764 {
765     \int_compare:nNnTF \c@iRow = 0
766     {
767         \dim_gset:Nn \g_@@_dp_row_zero_dim
768         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
769         \dim_gset:Nn \g_@@_ht_row_zero_dim
770         { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
771     }
772     {
773         \int_compare:nNnT \c@iRow = 1
774         {
775             \dim_gset:Nn \g_@@_ht_row_one_dim

```

```

776         { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
777     }
778 }
779 }

```

```

780 \cs_new_protected:Npn \@@_end_Cell:
781 {
782     \@@_math_toggle_token:
783     \hbox_set_end:
784     \box_set_ht:Nn \l_@@_cell_box
785     { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
786     \box_set_dp:Nn \l_@@_cell_box
787     { \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”).

```

788     \dim_gset:Nn \g_@@_max_cell_width_dim
789     { \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”.

```

790     \@@_update_for_first_and_last_row:

```

If the cell is empty, or may be considered as if, we must not create the PGF node, for two reasons:

- it’s a waste of time since such a node would be rather pointless;
- we test the existence of these nodes in order to determine whether a cell is empty when we search the extremities of a dotted line.

However, it’s very difficult to determine whether a cell is empty. As of now, we use the following technic:

- if the width of the box `\l_@@_cell_box` (created with the content of the cell) is equal to zero, we consider the cell as empty (however, this is not perfect since the user may have used a `\rlap`, a `\llap` or a `\mathclap` of `mathtools`).
- the cells with a command `\Ldots` or `\Cdots`, `\Vdots`, etc., should also be considered as empty; if `nullify-dots` is in force, there would be nothing to do (in this case the previous commands only write an instruction in a kind of `code-after`); however, if `nullify-dots` is not in force, a phantom of `\ldots`, `\cdots`, `\vdots` is inserted and its width is not equal to zero; that’s why these commands raise a boolean `\g_@@_empty_cell_bool` and we begin by testing this boolean.

```

791     \bool_if:NTF \g_@@_empty_cell_bool
792     { \box_use_drop:N \l_@@_cell_box }
793     {
794         \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
795         \@@_node_for_the_cell:
796         { \box_use_drop:N \l_@@_cell_box }
797     }
798     \bool_gset_false:N \g_@@_empty_cell_bool
799 }

```

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

```

800 \cs_new_protected:Npn \@@_node_for_the_cell:
801 {
802     \pgfpicture
803     \pgfsetbaseline \c_zero_dim
804     \pgfrememberpicturepositiononpagetrue
805     \pgfset
806     {
807         inner-sep = \c_zero_dim ,
808         minimum-width = \c_zero_dim
809     }
810     \pgfnode

```

```

811 { rectangle }
812 { base }
813 { \box_use_drop:N \l_@@_cell_box }
814 { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
815 { }
816 \str_if_empty:NF \l_@@_name_str
817 {
818   \pgfnodealias
819   { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
820   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
821 }
822 \endpgfpicture
823 }

```

The first argument of the following command `\@@_instruction_of_type:nn` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). The second 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}

```

```

824 \cs_new_protected:Npn \@@_instruction_of_type:nn #1 #2
825 {

```

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).

```

826 \tl_gput_right:cx
827 { g_@@_ #1 _ lines _ tl }
828 {
829   \use:c { @@ _ draw _ #1 : nnn }
830   { \int_use:N \c@iRow }
831   { \int_use:N \c@jCol }
832   { \exp_not:n { #2 } }
833 }
834 }

```

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`.

```

835 \cs_new_protected:Npn \@@_revtex_array:
836 {
837   \cs_set_eq:NN \@acoll \@arrayacol
838   \cs_set_eq:NN \@acolr \@arrayacol
839   \cs_set_eq:NN \@acol \@arrayacol
840   \cs_set:Npn \@halignto { }
841   \@array@array
842 }
843 \cs_new_protected:Npn \@@_array:
844 {
845   \bool_if:NTF \c_@@_revtex_bool
846   \@@_revtex_array:
847   {

```



```

848 \bool_if:NTF \l_@@_NiceTabular_bool
849 { \dim_set_eq:NN \col@sep \tabcolsep }
850 { \dim_set_eq:NN \col@sep \arraycolsep }
851 \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim
852 { \cs_set:Npn \@halignto { } }
853 { \cs_set:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }

```

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

```

854 \tabarray
855 }

```

`\l_@@_baseline_str` 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.

```

856 [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
857 }

```

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.

```

858 \cs_set_eq:NN \@_old_ialign: \ialign

```

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

```

859 \cs_new_protected:Npn \@_create_row_node:
860 {

```

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

```

861 \hbox
862 {
863 \bool_if:NT \l_@@_code_before_bool
864 {
865 \vtop
866 {
867 \skip_vertical:N 0.5\arrayrulewidth
868 \pgfsys@markposition { \@_env: - row - \@_succ:n \c@iRow }
869 \skip_vertical:N -0.5\arrayrulewidth
870 }
871 }
872 \pgfpicture
873 \pgfrememberpicturerepositiononpagetrue
874 \pgfcoordinate { \@_env: - row - \@_succ:n \c@iRow }
875 { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
876 \str_if_empty:NF \l_@@_name_str
877 {
878 \pgfnodealias
879 { \l_@@_name_str - row - \int_use:N \c@iRow }
880 { \@_env: - row - \int_use:N \c@iRow }
881 }
882 \endpgfpicture
883 }
884 }

```

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

```

885 \cs_new:Npn \@_everycr: { \noalign { \@_everycr_i: } }
886 \cs_new_protected:Npn \@_everycr_i:
887 {
888 \int_gzero:N \c@jCol
889 \bool_if:NF \g_@@_row_of_col_done_bool
890 {
891 \@_create_row_node:

```

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

```

892 \bool_if:NT \l_@@_hlines_bool
893 {

```

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.

```

894         \int_compare:nNnT \c@iRow > { -1 }
895         {
896             \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@`.

```

897             { \hrule height \arrayrulewidth width \c_zero_dim }
898         }
899     }
900 }
901 }

```

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`).

```

902 \cs_set_protected:Npn \@@_newcolumnntype #1
903 {
904     \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
905     \peek_meaning:NTF [
906         { \newcol@ #1 }
907         { \newcol@ #1 [ 0 ] }
908     }

```

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

```

909 \cs_set_protected:Npn \@@_renew_dots:
910 {
911     \cs_set_eq:NN \ldots \@@_Ldots
912     \cs_set_eq:NN \cdots \@@_Cdots
913     \cs_set_eq:NN \vdots \@@_Vdots
914     \cs_set_eq:NN \ddots \@@_Ddots
915     \cs_set_eq:NN \iddots \@@_Iddots
916     \cs_set_eq:NN \dots \@@_Ldots
917     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
918 }

```

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

```

919 \cs_new_protected:Npn \@@_colortbl_like:
920 {
921     \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
922     \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
923     \cs_set_eq:NN \columncolor \@@_columncolor_preamble
924 }

```

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

```

925 \cs_new_protected:Npn \@@_pre_array:
926 {

```

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>35</sup>.

```

927     \bool_if:NT \c_@@_booktabs_loaded_bool

```

---

<sup>35</sup>cf. `\nicematrix@redefine@check@rerun`

```

928     { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
929     \box_clear_new:N \l_@@_cell_box
930     \cs_if_exist:NT \theiRow
931     { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
932     \int_gzero_new:N \c@iRow
933     \cs_if_exist:NT \thejCol
934     { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
935     \int_gzero_new:N \c@jCol
936     \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}`).

```

937     \bool_if:NT \l_@@_small_bool
938     {
939         \cs_set:Npn \arraystretch { 0.47 }
940         \dim_set:Nn \arraycolsep { 1.45 pt }
941     }

```

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`.

```

942     \cs_set:Npn \ialign
943     {
944         \bool_if:NTF \c_@@_colortbl_loaded_bool
945         {
946             \CT@everycr
947             {
948                 \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
949                 \@@_everycr:
950             }
951         }
952         { \everycr { \@@_everycr: } }
953         \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 values of `\arraystretch`<sup>36</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`.

```

954         \dim_gzero_new:N \g_@@_dp_row_zero_dim
955         \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
956         \dim_gzero_new:N \g_@@_ht_row_zero_dim
957         \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
958         \dim_gzero_new:N \g_@@_ht_row_one_dim
959         \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
960         \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
961         \dim_gzero_new:N \g_@@_ht_last_row_dim
962         \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
963         \dim_gzero_new:N \g_@@_dp_last_row_dim
964         \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`.

```

965         \cs_set_eq:NN \ialign \@@_old_ialign:
966         \halign
967     }

```

---

<sup>36</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

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).

```

968 \cs_set_eq:NN \@@_old_ldots \ldots
969 \cs_set_eq:NN \@@_old_cdots \cdots
970 \cs_set_eq:NN \@@_old_vdots \vdots
971 \cs_set_eq:NN \@@_old_ddots \ddots
972 \cs_set_eq:NN \@@_old_iddots \iddots
973 \bool_if:NTF \l_@@_standard_cline_bool
974 { \cs_set_eq:NN \cline \@@_standard_cline }
975 { \cs_set_eq:NN \cline \@@_cline }
976 \cs_set_eq:NN \Ldots \@@_Ldots
977 \cs_set_eq:NN \Cdots \@@_Cdots
978 \cs_set_eq:NN \Vdots \@@_Vdots
979 \cs_set_eq:NN \Ddots \@@_Ddots
980 \cs_set_eq:NN \Iddots \@@_Iddots
981 \cs_set_eq:NN \hdottedline \@@_hdottedline:
982 \cs_set_eq:NN \Hspace \@@_Hspace:
983 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
984 \cs_set_eq:NN \Vdotsfor \@@_Vdotsfor:
985 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
986 \cs_set_eq:NN \Block \@@_Block:
987 \cs_set_eq:NN \rotate \@@_rotate:
988 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
989 \cs_set_eq:NN \dotfill \@@_dotfill:
990 \cs_set_eq:NN \CodeAfter \@@_CodeAfter:n
991 \cs_set_eq:NN \diagbox \@@_diagbox:nn
992 \bool_if:NT \l_@@_colortbl_like_bool \@@_colortbl_like:
993 \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).

```

994 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
995 \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).

```

996 \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 of rows excepted the last row (if `\l_@@_last_row_bool` has been raised with the option `last-row`).

```

997 \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.

```

998 \int_gzero_new:N \g_@@_col_total_int
999 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1000 \@@_renew_NC@rewrite@S:
1001 \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.

```

1002 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1003 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1004 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1005 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1006 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1007 \tl_gclear_new:N \g_@@_HVdotsfor_lines_tl

```

```

1008 \tl_gclear_new:N \g_@@_code_before_tl
1009 }

```

This is the end of \@@\_pre\_array:.

## The environment {NiceArrayWithDelims}

```

1010 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
1011 {
1012 \@@_provide_pgfsyspdfmark:
1013 \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.

```

1014 \bgroup
1015 \tl_set:Nn \l_@@_left_delim_tl { #1 }
1016 \tl_set:Nn \l_@@_right_delim_tl { #2 }
1017 \bool_gset_false:N \g_@@_row_of_col_done_bool
1018 \str_if_empty:NT \g_@@_name_env_str
1019 { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1020 \@@_adapt_S_column:
1021 \bool_if:NTF \l_@@_NiceTabular_bool
1022 \mode_leave_vertical:
1023 \@@_test_if_math_mode:
1024 \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
1025 \bool_set_true:N \l_@@_in_env_bool

```

The command \CT@arc@ contains the instruction of color for the rules of the array<sup>37</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.

```

1026 \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).

```

1027 \cs_if_exist:NT \tikz@library@external@loaded
1028 {
1029 \tikzset { external / export = false }
1030 \cs_if_exist:NT \ifstandalone
1031 { \tikzset { external / optimize = false } }
1032 }

```

We increment the counter \g\_@@\_env\_int which counts the environments of the package.

```

1033 \int_gincr:N \g_@@_env_int
1034 \bool_if:NF \l_@@_block_auto_columns_width_bool
1035 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

We do a redefinition of \@arrayrule because we want that the vertical rules drawn by | in the preamble of the array don't extend in the potential exterior rows.

```

1036 \cs_set_protected:Npn \@arrayrule { \@addtopreamble \@@_vline: }

```

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.

```

1037 \seq_clear:N \g_@@_blocks_seq
1038 \seq_clear: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.

```

1039 \tl_if_exist:cT { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1040 {
1041 \bool_set_true:N \l_@@_code_before_bool
1042 \exp_args:NNv \tl_put_right:Nn \l_@@_code_before_tl

```

---

<sup>37</sup>e.g. \color[rgb]{0.5,0.5,0}

```

1043     { g_@@_code_before_ \int_use:N \g_@@_env_int _ t1 }
1044   }

```

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`.

```

1045   \bool_if:NTF \l_@@_NiceArray_bool
1046     { \keys_set:nn { NiceMatrix / NiceArray } }
1047     { \keys_set:nn { NiceMatrix / pNiceArray } }
1048   { #3 , #5 }

1049   \tl_if_empty:NF \l_@@_rules_color_tl
1050     { \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.

```

1051   \bool_if:NT \l_@@_code_before_bool
1052   {
1053     \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1054     {

```

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).

```

1055       \int_zero_new:N \c@iRow
1056       \int_set:Nn \c@iRow
1057         { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1058       \int_zero_new:N \c@jCol
1059       \int_set:Nn \c@jCol
1060         { \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.

```

1061       \int_compare:nNnF \l_@@_last_row_int = { -2 }
1062       { \int_decr:N \c@iRow }
1063       \int_compare:nNnF \l_@@_last_col_int = { -2 }
1064       { \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.

```

1065       \pgfsys@markposition { \@@_env: - position }
1066       \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1067       \pgfpicture

```

First, the creation of the `row` nodes.

```

1068       \int_step_inline:nnn
1069       { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1070       { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1071       {
1072         \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1073         \pgfcoordinate { \@@_env: - row - ##1 }
1074         { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1075       }

```

Now, the creation of the `col` nodes.

```

1076       \int_step_inline:nnn
1077       { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1078       { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1079       {
1080         \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1081         \pgfcoordinate { \@@_env: - col - ##1 }
1082         { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1083       }
1084       \endpgfpicture

```

```

1085 \group_begin:
1086 \bool_if:NT \c_@@_tikz_loaded_bool
1087 {
1088   \tikzset
1089   {
1090     every~picture / .style =
1091     { overlay , name~prefix = \@@_env: - }
1092   }
1093 }
1094 \cs_set_eq:NN \cellcolor \@@_cellcolor
1095 \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
1096 \cs_set_eq:NN \rowcolor \@@_rowcolor
1097 \cs_set_eq:NN \rowcolors \@@_rowcolors
1098 \cs_set_eq:NN \columncolor \@@_columncolor
1099 \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`.

```

1100 \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1101 \l_@@_code_before_tl
1102 \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1103 \group_end:
1104 }
1105 }

```

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).

```

1106 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1107 {
1108   \tl_put_right:Nn \@@_update_for_first_and_last_row:
1109   {
1110     \dim_gset:Nn \g_@@_ht_last_row_dim
1111     { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1112     \dim_gset:Nn \g_@@_dp_last_row_dim
1113     { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1114   }
1115 }
1116 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1117 {
1118   \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.

```

1119 \str_if_empty:NTF \l_@@_name_str
1120 {
1121   \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1122   {
1123     \int_set:Nn \l_@@_last_row_int
1124     { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1125   }
1126 }
1127 {
1128   \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1129   {
1130     \int_set:Nn \l_@@_last_row_int
1131     { \use:c { @@_last_row_ \l_@@_name_str } }
1132   }
1133 }
1134 }

```

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).

```

1135 \int_compare:nNnT \l_@@_last_col_int = { -1 }
1136 {
1137   \str_if_empty:NTF \l_@@_name_str

```

```

1138     {
1139         \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1140         {
1141             \int_set:Nn \l_@@_last_col_int
1142             { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1143         }
1144     }
1145     {
1146         \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1147         {
1148             \int_set:Nn \l_@@_last_col_int
1149             { \use:c { @@_last_col_ \l_@@_name_str } }
1150         }
1151     }
1152 }

```

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

```

1153 \@@_pre_array:

```

We compute the width of the two delimiters.

```

1154 \dim_zero_new:N \l_@@_left_delim_dim
1155 \dim_zero_new:N \l_@@_right_delim_dim
1156 \bool_if:NTF \l_@@_NiceArray_bool
1157 {
1158     \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1159     \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1160 }
1161 {

```

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

```

1162 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
1163 \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1164 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
1165 \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1166 }

```

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

```

1167 \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).

```

1168 \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:

```

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

```

1169 \@@_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).

```

1170 \hbox_set:Nw \l_@@_the_array_box
1171 \skip_horizontal:N \l_@@_left_margin_dim
1172 \skip_horizontal:N \l_@@_extra_left_margin_dim
1173 \c_math_toggle_token
1174 \bool_if:NTF \l_@@_light_syntax_bool
1175 { \use:c { @@-light-syntax } }
1176 { \use:c { @@-normal-syntax } }
1177 }
1178 {

```



```

1179 \bool_if:NTF \l_@@_light_syntax_bool
1180   { \use:c { end @@-light-syntax } }
1181   { \use:c { end @@-normal-syntax } }
1182 \c_math_toggle_token
1183 \skip_horizontal:N \l_@@_right_margin_dim
1184 \skip_horizontal:N \l_@@_extra_right_margin_dim
1185 \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).

```

1186 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1187 {
1188   \bool_if:NF \l_@@_last_row_without_value_bool
1189   {
1190     \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1191     {
1192       \@@_error:n { Wrong~last~row }
1193       \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1194     }
1195   }
1196 }

```

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>38</sup>

```

1197 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1198 \bool_if:NTF \g_@@_last_col_found_bool
1199   { \int_gdecr:N \c@jCol }
1200   {
1201     \int_compare:nNnT \l_@@_last_col_int > { -1 }
1202     { \@@_error:n { last~col~not~used } }
1203   }
1204 \bool_if:NF \l_@@_Matrix_bool
1205 {
1206   \int_compare:nNnT \c@jCol < \g_@@_static_num_of_col_int
1207   { \@@_error:n { columns~not~used } }
1208 }

```

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

```

1209 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1210 \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. 87).

```

1211 \int_compare:nNnT \l_@@_first_col_int = 0
1212 {
1213   \skip_horizontal:N \arraycolsep
1214   \skip_horizontal:N \g_@@_width_first_col_dim
1215 }

```

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. We begin with this case.

```

1216 \bool_if:NTF \l_@@_NiceArray_bool
1217 {
1218   \str_case:VnF \l_@@_baseline_str
1219   {
1220     b \@@_use_arraybox_with_notes_b:
1221     c \@@_use_arraybox_with_notes_c:
1222   }

```

---

<sup>38</sup>We remind that the potential “first column” (exterior) has the number 0.

```

1223 \@@_use_arraybox_with_notes:
1224 }

```

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).

```

1225 {
1226   \int_compare:nNnTF \l_@@_first_row_int = 0
1227   {
1228     \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1229     \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1230   }
1231   { \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>39</sup>

```

1232   \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1233   {
1234     \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1235     \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1236   }
1237   { \dim_zero:N \l_tmpb_dim }
1238   \hbox_set:Nn \l_tmpa_box
1239   {
1240     \c_math_toggle_token
1241     \left #1
1242     \vcenter
1243     {

```

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.

```

1244       \skip_vertical:N -\l_tmpa_dim
1245       \skip_vertical:N -\arrayrulewidth
1246       \hbox
1247       {
1248         \bool_if:NTF \l_@@_NiceTabular_bool
1249         { \skip_horizontal:N -\tabcolsep }
1250         { \skip_horizontal:N -\arraycolsep }
1251         \@@_use_arraybox_with_notes_c:
1252         \bool_if:NTF \l_@@_NiceTabular_bool
1253         { \skip_horizontal:N -\tabcolsep }
1254         { \skip_horizontal:N -\arraycolsep }
1255       }

```

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

```

1256       \skip_vertical:N -\l_tmpb_dim
1257       \skip_vertical:N \arrayrulewidth
1258     }
1259     \right #2
1260     \c_math_toggle_token
1261   }

```

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.

```

1262   \bool_if:NTF \l_@@_max_delimiter_width_bool
1263   { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1264   \@@_put_box_in_flow:
1265 }

```

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. 88).

```

1266   \bool_if:NT \g_@@_last_col_found_bool
1267   {
1268     \skip_horizontal:N \g_@@_width_last_col_dim

```

---

<sup>39</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).

```

1269     \skip_horizontal:N \arraycolsep
1270   }
1271   \@@_after_array:
1272   \egroup
1273   \bool_if:NT \c_@@_footnote_bool \endsavenotes
1274 }

```

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`.

```

1275 \cs_new_protected:Npn \@@_construct_preamble:n #1
1276 {

```

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.

```

1277   \group_begin:

```

If we are in an environment without explicit preamble, we have nothing to do (excepted the treatment on both sides of the preamble which will be done at the end).

```

1278   \bool_if:NTF \l_@@_Matrix_bool
1279   { \tl_gset:Nn \g_@@_preamble_tl { #1 } }
1280   {
1281     \@@_newcolumntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1282     \@@_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`).

```

1283     \@temptokena { #1 }

```

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

```

1284     \@tempswatrue

```

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

```

1285     \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`.

```

1286     \int_gzero_new:N \c@jCol
1287     \bool_if:NTF \l_@@_vlines_bool
1288     {
1289       \tl_gset:Nn \g_@@_preamble_tl
1290       { ! { \skip_horizontal:N \arrayrulewidth } }
1291     }
1292     { \tl_gclear:N \g_@@_preamble_tl }

```

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

```

1293     \int_zero:N \l_tmpa_int

```

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

```

1294     \exp_after:wN \@@_patch_preamble:n \the \temptokena \q_stop
1295     \int_gset_eq:NN \g_@@_static_num_of_col_int \c_jCol
1296 }

```

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

```

1297 \bool_if:NT \l_@@_colortbl_like_bool
1298 {
1299     \regex_replace_all:NnN
1300     \c_@@_columncolor_regex
1301     { \c { \@@_columncolor_preamble } }
1302     \g_@@_preamble_tl
1303 }

```

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

```

1304 \int_compare:nNnTF \l_@@_first_col_int = 0
1305 { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1306 {
1307     \bool_lazy_all:nT
1308     {
1309         \l_@@_NiceArray_bool
1310         { \bool_not_p:n \l_@@_NiceTabular_bool }
1311         { \bool_not_p:n \l_@@_vlines_bool }
1312         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1313     }
1314     { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1315 }
1316 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1317 { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }
1318 {
1319     \bool_lazy_all:nT
1320     {
1321         \l_@@_NiceArray_bool
1322         { \bool_not_p:n \l_@@_NiceTabular_bool }
1323         { \bool_not_p:n \l_@@_vlines_bool }
1324         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1325     }
1326     { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1327 }

```

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`).

```

1328 \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1329 {
1330     \tl_gput_right:Nn
1331     \g_@@_preamble_tl
1332     { > { \@@_error_too_much_cols: } 1 }
1333 }

```

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

```

1334 \group_end:
1335 }

```

```

1336 \cs_new_protected:Npn \@@_patch_preamble:n #1
1337 {
1338     \str_case:nnF { #1 }
1339     {
1340         c { \@@_patch_preamble_i:n #1 }
1341         l { \@@_patch_preamble_i:n #1 }
1342         r { \@@_patch_preamble_i:n #1 }

```

```

1343 > { \@@_patch_preamble_ii:nn #1 }
1344 < { \@@_patch_preamble_ii:nn #1 }
1345 ! { \@@_patch_preamble_ii:nn #1 }
1346 @ { \@@_patch_preamble_ii:nn #1 }
1347 | { \@@_patch_preamble_iii:n #1 }
1348 p { \@@_patch_preamble_iv:nnn t #1 }
1349 m { \@@_patch_preamble_iv:nnn c #1 }
1350 b { \@@_patch_preamble_iv:nnn b #1 }
1351 \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1352 \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1353 \@@_true_c: { \@@_patch_preamble_vi:n #1 }
1354 \q_stop { }
1355 }
1356 {
1357   \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1358   { \@@_patch_preamble_vii:n #1 }
1359   { \@@_fatal:nn { unknown~column~type } { #1 } }
1360 }
1361 }

```

For c, l and r

```

1362 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1363 {
1364   \tl_gput_right:Nn \g_@@_preamble_tl { > \@@_Cell: #1 < \@@_end_Cell: }

```

We increment the counter of columns.

```

1365   \int_gincr:N \c@jCol
1366   \bool_if:NT \l_@@_vlines_bool
1367   {
1368     \tl_gput_right:Nn \g_@@_preamble_tl
1369     { ! { \skip_horizontal:N \arrayrulewidth } }
1370   }
1371   \@@_patch_preamble:n
1372 }

```

For >, <, ! and @

```

1373 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1374 {
1375   \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
1376   \@@_patch_preamble:n
1377 }

```

For |

```

1378 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1379 {

```

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

```

1380   \int_incr:N \l_tmpa_int
1381   \@@_patch_preamble_iii_i:n
1382 }
1383 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1384 {
1385   \str_if_eq:nnTF { #1 } |
1386   { \@@_patch_preamble_iii:n | }
1387   {
1388     \tl_gput_right:Nx \g_@@_preamble_tl
1389     {
1390       \exp_not:N !
1391       {
1392         \skip_horizontal:n
1393         {
1394           \dim_eval:n
1395           {

```

```

1396             \arrayrulewidth * \l_tmpa_int
1397             + \doublerulesep * ( \l_tmpa_int - 1)
1398         }
1399     }
1400 }
1401 }
1402 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1403 { \@@_vline:nn { \int_use:N \c@jCol } { \int_use:N \l_tmpa_int } }
1404 \int_zero:N \l_tmpa_int
1405 \@@_patch_preamble:n #1
1406 }
1407 }

```

For p, m and b

```

1408 \cs_new_protected:Npn \@@_patch_preamble_iv:nnn #1 #2 #3
1409 {
1410     \tl_gput_right:Nn \g_@@_preamble_tl
1411     {
1412         > {
1413             \@@_Cell:
1414             \begin { minipage } [ #1 ] { #3 }
1415             \mode_leave_vertical:
1416             \box_use:N \@arstrutbox
1417         }
1418         c
1419         < { \box_use:N \@arstrutbox \end { minipage } \@@_end_Cell: }
1420     }

```

We increment the counter of columns.

```

1421     \int_gincr:N \c@jCol
1422     \@@_patch_preamble:n
1423 }

```

For w and W

```

1424 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1425 {
1426     \tl_gput_right:Nn \g_@@_preamble_tl
1427     {
1428         > {
1429             \hbox_set:Nw \l_@@_cell_box
1430             \@@_Cell:
1431         }
1432         c
1433         < {
1434             \@@_end_Cell:
1435             #1
1436             \hbox_set_end:
1437             \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
1438         }
1439     }

```

We increment the counter of columns.

```

1440     \int_gincr:N \c@jCol
1441     \@@_patch_preamble:n
1442 }

```

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

```

1443 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1444 {
1445     \tl_gput_right:Nn \g_@@_preamble_tl { c }

```

We increment the counter of columns.

```

1446     \int_gincr:N \c@jCol
1447     \bool_if:NT \l_@@_vlines_bool

```

```

1448 {
1449   \tl_gput_right:Nn \g_@@_preamble_tl
1450   { ! { \skip_horizontal:N \arrayrulewidth } }
1451 }
1452 \@@_patch_preamble:n
1453 }
1454 \cs_new_protected:Npn \@@_patch_preamble_vii:n #1
1455 {

```

Here, we have a problem in the cases of the use in the first column or the “last one”.

```

1456   \tl_gput_right:Nn \g_@@_preamble_tl
1457   { ! { \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`.

```

1458   \tl_gput_right:Nx \g_@@_internal_code_after_tl
1459   { \@@_vdottedline:n { \int_use:N \c@jCol } }
1460   \@@_patch_preamble:n
1461 }

```

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`).

```

1462 \cs_new_protected:Npn \@@_put_box_in_flow:
1463 {
1464   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1465   \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1466   \str_if_eq:VnTF \l_@@_baseline_str { c }
1467   { \box_use_drop:N \l_tmpa_box }
1468   \@@_put_box_in_flow_i:
1469 }

```

The command `\@@_put_box_in_flow_i:` is used when the value of `\l_@@_baseline_str` is different of `c` (which is the initial value and the most used).

```

1470 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1471 {
1472   \str_case:VnF \l_@@_baseline_str
1473   {
1474     { t } { \int_set:Nn \l_tmpa_int 1 }
1475     { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1476   }
1477   { \int_set:Nn \l_tmpa_int \l_@@_baseline_str }
1478   \bool_if:nTF
1479   {
1480     \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
1481     || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
1482   }
1483   {
1484     \@@_error:n { bad~value~for~baseline }
1485     \int_set:Nn \l_tmpa_int 1
1486   }
1487   \pgfpicture
1488   \@@_qpoint:n { row - 1 }
1489   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1490   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1491   \dim_gadd:Nn \g_tmpa_dim \pgf@y
1492   \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).

```

1493   \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1494   \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

We take into account the position of the mathematical axis.

```
1495 \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
```

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

```
1496 \endpgfpicture
1497 \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1498 \box_use_drop:N \l_tmpa_box
1499 }
```

```
1500 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
1501 {
1502   \int_compare:nNnTF \c@tabularnote = 0
1503   { \box_use_drop:N \l_@@_the_array_box }
1504   {
1505     \begin { minipage } { \box_wd:N \l_@@_the_array_box }
1506     \box_use_drop:N \l_@@_the_array_box
1507     \skip_vertical:N 0.65ex
```

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

```
1508 \group_begin:
1509 \l_@@_notes_code_before_tl
```

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.

```
1510 \bool_if:NTF \l_@@_notes_para_bool
1511 {
1512   \begin { tabularnotes* }
1513   \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1514   \end { tabularnotes* }
```

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

```
1515 \par
1516 }
1517 {
1518   \tabularnotes
1519   \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1520   \endtabularnotes
1521 }
1522 \unskip
1523 \group_end:
1524 \bool_if:NT \l_@@_notes_bottomrule_bool
1525 {
1526   \bool_if:NTF \c_@@_booktabs_loaded_bool
1527   {
```

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

```
1528 \skip_vertical:N \aboverulesep
```

`\CT@arc@` is the specification of color defined by `colortbl` but you use it even if `colortbl` is not loaded.

```
1529 { \CT@arc@ \hrule height \heavyrulewidth }
1530 }
1531 { \@@_error:n { bottomule~without~booktabs } }
1532 }
1533 \l_@@_notes_code_after_tl
1534 \end { minipage }
1535 \seq_gclear:N \g_@@_tabularnotes_seq
1536 \int_gzero:N \c@tabularnote
1537 }
1538 }
```



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`.

```

1539 \cs_new_protected:Npn \@@_use_arraybox_with_notes_b:
1540 {
1541   \pgfpicture
1542     \@@_qpoint:n { row - 1 }
1543     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1544     \@@_qpoint:n { row - \int_use:N \c@iRow - base }
1545     \dim_gsub:Nn \g_tmpa_dim \pgf@y
1546   \endpgfpicture
1547   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1548   \int_compare:nNnT \l_@@_first_row_int = 0
1549   {
1550     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1551     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1552   }
1553   \box_move_up:nn \g_tmpa_dim { \@@_use_arraybox_with_notes_c: }
1554 }

```

Now, the general case (hence the `g` in the name).

```

1555 \cs_new_protected:Npn \@@_use_arraybox_with_notes:
1556 {

```

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

```

1557   \str_if_eq:VnT \l_@@_baseline_str { t }
1558   { \str_set:Nn \l_@@_baseline_str { 1 } }

```

Now, we convert the value of `\l_@@_baseline_str` (which should represent an integer) to an integer stored in `\l_tmpa_int`.

```

1559   \int_set:Nn \l_tmpa_int \l_@@_baseline_str
1560   \bool_lazy_or:nnT
1561     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1562     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1563     {
1564       \@@_error:n { bad~value~for~baseline }
1565       \int_set:Nn \l_tmpa_int 1
1566     }
1567   \pgfpicture
1568     \@@_qpoint:n { row - 1 }
1569     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1570     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1571     \dim_gsub:Nn \g_tmpa_dim \pgf@y
1572   \endpgfpicture
1573   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1574   \int_compare:nNnT \l_@@_first_row_int = 0
1575   {
1576     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1577     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1578   }
1579   \box_move_up:nn \g_tmpa_dim { \@@_use_arraybox_with_notes_c: }
1580 }

```

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.

```

1581 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1582 {

```

We will compute the real width of both delimiters used.

```

1583   \dim_zero_new:N \l_@@_real_left_delim_dim
1584   \dim_zero_new:N \l_@@_real_right_delim_dim
1585   \hbox_set:Nn \l_tmpb_box

```

```

1586 {
1587   \c_math_toggle_token
1588   \left #1
1589   \vcenter
1590   {
1591     \vbox_to_ht:nn
1592     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1593     { }
1594   }
1595   \right .
1596   \c_math_toggle_token
1597 }
1598 \dim_set:Nn \l_@@_real_left_delim_dim
1599 { \box_wd:N \l_tmpb_box - \nullldelimiterspace }
1600 \hbox_set:Nn \l_tmpb_box
1601 {
1602   \c_math_toggle_token
1603   \left .
1604   \vbox_to_ht:nn
1605   { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1606   { }
1607   \right #2
1608   \c_math_toggle_token
1609 }
1610 \dim_set:Nn \l_@@_real_right_delim_dim
1611 { \box_wd:N \l_tmpb_box - \nullldelimiterspace }

```

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

```

1612 \skip_horizontal:N \l_@@_left_delim_dim
1613 \skip_horizontal:N -\l_@@_real_left_delim_dim
1614 \@@_put_box_in_flow:
1615 \skip_horizontal:N \l_@@_right_delim_dim
1616 \skip_horizontal:N -\l_@@_real_right_delim_dim
1617 }

```

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).

```

1618 \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).

```

1619 {
1620   \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`).

```

1621   { \exp_args:NV \@@_array: \g_@@_preamble_tl }
1622 }
1623 {
1624   \@@_create_col_nodes:
1625   \endarray
1626 }

```

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`).

```

1627 \NewDocumentEnvironment { @@-light-syntax } { b }
1628 {

```

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.

```

1629 \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1630 \tl_map_inline:nn { #1 }
1631 {
1632   \tl_if_eq:nnT { ##1 } { & }
1633   { \@@_fatal:n { ampersand-in-light-syntax } }
1634   \tl_if_eq:nnT { ##1 } { \ }
1635   { \@@_fatal:n { double-backslash-in-light-syntax } }
1636 }

```

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`.

```

1637 \@@_light_syntax_i #1 \CodeAfter \q_stop
1638 }

```

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`.

```

1639 { }

1640 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1641 {
1642   \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).

```

1643 \seq_gclear_new:N \g_@@_rows_seq
1644 \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1645 \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.

```

1646 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1647 { \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`).

```

1648 \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.

```

1649 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1650 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1651 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1652 \@@_create_col_nodes:
1653 \endarray
1654 }

1655 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1656 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }

1657 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1658 {
1659   \seq_gclear_new:N \g_@@_cells_seq
1660   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1661   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1662   \l_tmpa_tl
1663   \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1664 }

```

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).

```

1665 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1666 {
1667   \str_if_eq:VnT \g_@@_name_env_str { #2 }
1668   { \@@_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.

```

1669   \end { #2 }
1670 }

```

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).

```

1671 \cs_new:Npn \@@_create_col_nodes:
1672 {
1673   \crrc
1674   \int_compare:nNnT \l_@@_first_col_int = 0
1675   {
1676     \omit
1677     \skip_horizontal:N -2\col@sep
1678     \bool_if:NT \l_@@_code_before_bool
1679     { \pgfsys@markposition { \@@_env: - col - 0 } }
1680     \pgfpicture
1681     \pgfrememberpicturepositiononpagetrue
1682     \pgfcoordinate { \@@_env: - col - 0 } \pgfpintorigin
1683     \str_if_empty:NF \l_@@_name_str
1684     { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1685     \endpgfpicture
1686     &
1687   }
1688   \omit

```

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

```

1689   \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`).

```

1690   \int_compare:nNnTF \l_@@_first_col_int = 0
1691   {
1692     \bool_if:NT \l_@@_code_before_bool
1693     {
1694       \hbox
1695       {
1696         \skip_horizontal:N -0.5\arrayrulewidth
1697         \pgfsys@markposition { \@@_env: - col - 1 }
1698         \skip_horizontal:N 0.5\arrayrulewidth
1699       }
1700     }
1701     \pgfpicture
1702     \pgfrememberpicturepositiononpagetrue
1703     \pgfcoordinate { \@@_env: - col - 1 }
1704     { \pgfpint { - 0.5 \arrayrulewidth } \c_zero_dim }
1705     \str_if_empty:NF \l_@@_name_str
1706     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1707     \endpgfpicture
1708   }
1709   {
1710     \bool_if:NT \l_@@_code_before_bool
1711     {
1712       \hbox
1713       {

```

```

1714         \skip_horizontal:N 0.5 \arrayrulewidth
1715         \pgfsys@markposition { \@@_env: - col - 1 }
1716         \skip_horizontal:N -0.5\arrayrulewidth
1717     }
1718 }
1719 \pgfpicture
1720 \pgfrememberpicturepositiononpagetrue
1721 \pgfcoordinate { \@@_env: - col - 1 }
1722 { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
1723 \str_if_empty:NF \l_@@_name_str
1724 { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1725 \endpgfpicture
1726 }

```

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.

```

1727 \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1728 \bool_if:NF \l_@@_auto_columns_width_bool
1729 { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
1730 {
1731     \bool_lazy_and:nnTF
1732     \l_@@_auto_columns_width_bool
1733     { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1734     { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1735     { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1736     \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
1737 }
1738 \skip_horizontal:N \g_tmpa_skip
1739 \hbox
1740 {
1741     \bool_if:NT \l_@@_code_before_bool
1742     {
1743         \hbox
1744         {
1745             \skip_horizontal:N -0.5\arrayrulewidth
1746             \pgfsys@markposition { \@@_env: - col - 2 }
1747             \skip_horizontal:N 0.5\arrayrulewidth
1748         }
1749     }
1750     \pgfpicture
1751     \pgfrememberpicturepositiononpagetrue
1752     \pgfcoordinate { \@@_env: - col - 2 }
1753     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1754     \str_if_empty:NF \l_@@_name_str
1755     { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
1756     \endpgfpicture
1757 }

```

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.

```

1758 \int_gset:Nn \g_tmpa_int 1
1759 \bool_if:NNTF \g_@@_last_col_found_bool
1760 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1761 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1762 {
1763     &
1764     \omit

```

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

```

1765     \int_gincr:N \g_tmpa_int

```

```

1766 \skip_horizontal:N \g_tmpa_skip
1767 \bool_if:NT \l_@@_code_before_bool
1768 {
1769     \hbox
1770     {
1771         \skip_horizontal:N -0.5\arrayrulewidth
1772         \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1773         \skip_horizontal:N 0.5\arrayrulewidth
1774     }
1775 }

```

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

```

1776 \pgfpicture
1777 \pgfrememberpicturepositiononpagetrue
1778 \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1779 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1780 \str_if_empty:NF \l_@@_name_str
1781 {
1782     \pgfnodealias
1783     { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
1784     { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1785 }
1786 \endpgfpicture
1787 }
1788 \bool_if:NT \g_@@_last_col_found_bool
1789 {
1790     \bool_if:NT \l_@@_code_before_bool
1791     {
1792         \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1793     }
1794     \skip_horizontal:N 2\col@sep
1795     \pgfpicture
1796     \pgfrememberpicturepositiononpagetrue
1797     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1798     \pgfpointorigin
1799     \str_if_empty:NF \l_@@_name_str
1800     {
1801         \pgfnodealias
1802         { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
1803         { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1804     }
1805     \endpgfpicture
1806     \skip_horizontal:N -2\col@sep
1807 }
1808 \cr
1809 }

```

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

```

1810 \tl_const:Nn \c_@@_preamble_first_col_tl
1811 {
1812     >
1813     {
1814         \@@_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.

```

1815 \hbox_set:Nw \l_@@_cell_box
1816 \@@_math_toggle_token:
1817 \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”.

```

1818 \bool_lazy_and:nnT

```

```

1819 { \int_compare_p:nNn \c@iRow > 0 }
1820 {
1821   \bool_lazy_or_p:nn
1822     { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1823     { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1824   }
1825   {
1826     \l_@@_code_for_first_col_tl
1827     \xglobal \colorlet { nicematrix-first-col } { . }
1828   }
1829 }

```

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`.

```

1830   l
1831   <
1832   {
1833     \@@_math_toggle_token:
1834     \hbox_set_end:
1835     \@@_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.

```

1836   \dim_gset:Nn \g_@@_width_first_col_dim
1837   { \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.

```

1838   \hbox_overlap_left:n
1839   {
1840     \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1841       \@@_node_for_the_cell:
1842       { \box_use_drop:N \l_@@_cell_box }
1843       \skip_horizontal:N \l_@@_left_delim_dim
1844       \skip_horizontal:N \l_@@_left_margin_dim
1845       \skip_horizontal:N \l_@@_extra_left_margin_dim
1846     }
1847     \skip_horizontal:N -2\col@sep
1848   }
1849 }

```

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

```

1850 \tl_const:Nn \c_@@_preamble_last_col_tl
1851 {
1852   >
1853   {

```

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

```

1854   \bool_gset_true:N \g_@@_last_col_found_bool
1855   \int_gincr:N \c@jCol
1856   \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.

```

1857   \hbox_set:Nw \l_@@_cell_box
1858   \@@_math_toggle_token:
1859   \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”.

```

1860   \int_compare:nNnT \c@iRow > 0
1861   {
1862     \bool_lazy_or:nnT
1863       { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1864       { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1865     {
1866       \l_@@_code_for_last_col_tl

```

```

1867         \xglobal \colorlet { nicematrix-last-col } { . }
1868     }
1869 }
1870 }
1871 1
1872 <
1873 {
1874     \@@_math_toggle_token:
1875     \hbox_set_end:
1876     \@@_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.

```

1877     \dim_gset:Nn \g_@@_width_last_col_dim
1878     { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
1879     \skip_horizontal:N -2\col@sep

```

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

```

1880     \hbox_overlap_right:n
1881     {
1882         \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1883         {
1884             \skip_horizontal:N \l_@@_right_delim_dim
1885             \skip_horizontal:N \l_@@_right_margin_dim
1886             \skip_horizontal:N \l_@@_extra_right_margin_dim
1887             \@@_node_for_the_cell:
1888         }
1889     }
1890 }
1891 }

```

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.

```

1892 \NewDocumentEnvironment { NiceArray } { }
1893 {
1894     \bool_set_true:N \l_@@_NiceArray_bool
1895     \str_if_empty:NT \g_@@_name_env_str
1896     { \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).

```

1897     \NiceArrayWithDelims . .
1898 }
1899 { \endNiceArrayWithDelims }

```

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

```

1900 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
1901 {
1902     \NewDocumentEnvironment { #1 NiceArray } { }
1903     {
1904         \str_if_empty:NT \g_@@_name_env_str
1905         { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
1906         \@@_test_if_math_mode:
1907         \NiceArrayWithDelims #2 #3
1908     }
1909     { \endNiceArrayWithDelims }
1910 }
1911 \@@_def_env:nnn p ( )
1912 \@@_def_env:nnn b [ ]
1913 \@@_def_env:nnn B {\ }
1914 \@@_def_env:nnn v | |
1915 \@@_def_env:nnn V \l \r

```



## The environment `{NiceMatrix}` and its variants

```

1916 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
1917 {
1918   \bool_set_true:N \l_@@_Matrix_bool
1919   \use:c { #1 NiceArray }
1920   {
1921     *
1922     {
1923       \int_compare:nNnTF \l_@@_last_col_int < 0
1924         \c@MaxMatrixCols
1925         { \@@_pred:n \l_@@_last_col_int }
1926     }
1927     { > \@@_Cell: #2 < \@@_end_Cell: }
1928   }
1929 }
1930 \clist_map_inline:nn { { } , p , b , B , v , V }
1931 {
1932   \NewDocumentEnvironment { #1 NiceMatrix } { ! O { } }
1933   {
1934     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
1935     \tl_set:Nn \l_@@_type_of_col_tl c
1936     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1937     \exp_args:Nnx \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
1938   }
1939   { \use:c { end #1 NiceArray } }
1940 }

```

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

```

1941 \NewDocumentEnvironment { NiceTabular } { O { } m ! O { } }
1942 {
1943   \str_gset:Nn \g_@@_name_env_str { NiceTabular }
1944   \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
1945   \bool_set_true:N \l_@@_NiceTabular_bool
1946   \NiceArray { #2 }
1947 }
1948 { \endNiceArray }

1949 \NewDocumentEnvironment { NiceTabular* } { m O { } m ! O { } }
1950 {
1951   \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
1952   \dim_set:Nn \l_@@_tabular_width_dim { #1 }
1953   \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
1954   \bool_set_true:N \l_@@_NiceTabular_bool
1955   \NiceArray { #3 }
1956 }
1957 { \endNiceArray }

```

## After the construction of the array

```

1958 \cs_new_protected:Npn \@@_after_array:
1959 {
1960   \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.

```

1961 \bool_if:NT \g_@@_last_col_found_bool
1962 { \int_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int }
1963 \bool_if:NT \l_@@_last_col_without_value_bool
1964 {
1965   \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
1966   \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1967   \iow_shipout:Nx \@mainaux
1968   {
1969     \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
1970     { \int_use:N \g_@@_col_total_int }
1971   }
1972   \str_if_empty:NF \l_@@_name_str
1973   {
1974     \iow_shipout:Nx \@mainaux
1975     {
1976       \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
1977       { \int_use:N \g_@@_col_total_int }
1978     }
1979   }
1980   \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1981 }

```

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.

```

1982 \bool_if:NT \l_@@_last_row_without_value_bool
1983 {
1984   \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.

```

1985 \bool_if:NF \l_@@_light_syntax_bool
1986 {
1987   \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1988   \iow_shipout:Nx \@mainaux
1989   {
1990     \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1991     { \int_use:N \g_@@_row_total_int }
1992   }

```

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.

```

1993 \str_if_empty:NF \l_@@_name_str
1994 {
1995   \iow_shipout:Nx \@mainaux
1996   {
1997     \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1998     { \int_use:N \g_@@_row_total_int }
1999   }
2000 }
2001 \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2002 }
2003 }

```

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

```

2004 \bool_if:NT \l_@@_code_before_bool
2005 {
2006   \iow_now:Nn \@mainaux \ExplSyntaxOn
2007   \iow_now:Nx \@mainaux
2008   { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
2009   \iow_now:Nx \@mainaux
2010   {
2011     \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
2012     {

```

```

2013         \int_use:N \l_@@_first_row_int ,
2014         \int_use:N \g_@@_row_total_int ,
2015         \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.

```

2016         \bool_lazy_and:nnTF
2017         { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
2018         { \bool_not_p:n \g_@@_last_col_found_bool }
2019         \@@_succ:n
2020         \int_use:N
2021         \g_@@_col_total_int
2022     }
2023 }
2024 \iow_now:Nn \@mainaux \ExplSyntaxOff
2025 }

```

By default, the diagonal lines will be parallelized<sup>40</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.

```

2026     \bool_if:NT \l_@@_parallelize_diags_bool
2027     {
2028         \int_gzero_new:N \g_@@_ddots_int
2029         \int_gzero_new:N \g_@@_iddots_int

```

The dimensions `\g_@@_delta_x_one_dim` and `\g_@@_delta_y_one_dim` will contain the  $\Delta_x$  and  $\Delta_y$  of the first `\Ddots` diagonal. We have to store these values in order to draw the others `\Ddots` diagonals parallel to the first one. Similarly `\g_@@_delta_x_two_dim` and `\g_@@_delta_y_two_dim` are the  $\Delta_x$  and  $\Delta_y$  of the first `\Iddots` diagonal.

```

2030     \dim_gzero_new:N \g_@@_delta_x_one_dim
2031     \dim_gzero_new:N \g_@@_delta_y_one_dim
2032     \dim_gzero_new:N \g_@@_delta_x_two_dim
2033     \dim_gzero_new:N \g_@@_delta_y_two_dim
2034 }
2035 \bool_if:nTF \l_@@_medium_nodes_bool
2036 {
2037     \bool_if:NTF \l_@@_large_nodes_bool
2038     \@@_create_medium_and_large_nodes:
2039     \@@_create_medium_nodes:
2040 }
2041 { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
2042 \int_zero_new:N \l_@@_initial_i_int
2043 \int_zero_new:N \l_@@_initial_j_int
2044 \int_zero_new:N \l_@@_final_i_int
2045 \int_zero_new:N \l_@@_final_j_int
2046 \bool_set_false:N \l_@@_initial_open_bool
2047 \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.

```

2048     \bool_if:NT \l_@@_small_bool
2049     {
2050         \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2051         \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.

```

2052     \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2053 }

```

Now, we actually draw the dotted lines.

---

<sup>40</sup>It's possible to use the option `parallelize-diags` to disable this parallelization.

```

2054 \@@_draw_dotted_lines:
2055 \bool_if:NTF \l_@@_hvlines_bool
2056   \@@_draw_hvlines:
2057   {
2058     \bool_if:NT \l_@@_hlines_bool \@@_draw_hlines:
2059     \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
2060     \bool_if:NT \l_@@_hvlines_except_corners_bool
2061       \@@_draw_hvlines_except_corners:
2062   }

```

We have to revert to a clean version of `\ialign` because there may be tabulars in the `\Block` instructions that will be composed now.

```

2063 \cs_set_eq:NN \ialign \@@_old_ialign:
2064 \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
2065 \g_@@_internal_code_after_tl
2066 \tl_gclear:N \g_@@_internal_code_after_tl
2067 \bool_if:NT \c_@@_tikz_loaded_bool
2068   {
2069     \tikzset
2070     {
2071       every~picture / .style =
2072       {
2073         overlay ,
2074         remember~picture ,
2075         name~prefix = \@@_env: -
2076       }
2077     }
2078   }
2079 \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.

```

2080 \cs_set_eq:NN \CodeAfter \prg_do_nothing:

```

And here's the code-after:

```

2081 \g_nicematrix_code_after_tl
2082 \tl_gclear:N \g_nicematrix_code_after_tl
2083 \group_end:

```

`\g_@@_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.

```

2084 \tl_if_empty:NF \g_@@_code_before_tl
2085   {

```

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_@@_code_before_tl`. In order to avoid an error during the expansion, we define a protected version of `\rectanglecolor`.

```

2086 \cs_set_protected:Npn \rectanglecolor { }
2087 \cs_set_protected:Npn \columncolor { }
2088 \iow_now:Nn \@mainaux \ExplSyntaxOn
2089 \iow_now:Nx \@mainaux
2090   {
2091     \tl_gset:cn
2092       { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
2093     { \g_@@_code_before_tl }
2094   }
2095 \iow_now:Nn \@mainaux \ExplSyntaxOff
2096 \bool_set_true:N \l_@@_code_before_bool
2097 }

2098 \str_gclear:N \g_@@_name_env_str
2099 \@@_restore_iRow_jCol:

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>41</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`.

```
2100 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
2101 }
```

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).

```
2102 \AtBeginDocument
2103 {
2104 \cs_new_protected:Npx \@@_draw_dotted_lines:
2105 {
2106 \c_@@_pgfortikzpicture_tl
2107 \@@_draw_dotted_lines_i:
2108 \c_@@_endpgfortikzpicture_tl
2109 }
2110 }
```

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

```
2111 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2112 {
2113 \pgfrememberpicturepositiononpagetrue
2114 \pgf@relevantforpicturesizefalse
2115 \g_@@_HVdotsfor_lines_tl
2116 \g_@@_Vdots_lines_tl
2117 \g_@@_Ddots_lines_tl
2118 \g_@@_Iddots_lines_tl
2119 \g_@@_Cdots_lines_tl
2120 \g_@@_Ldots_lines_tl
2121 }

2122 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2123 {
2124 \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2125 \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2126 }
```

## We draw the dotted lines

A dotted line will be said *open* in one of its extremities when it stops on the edge of the matrix and *closed* otherwise. In the following matrix, the dotted line is closed on its left extremity and open on its right.

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

The command `\@@_find_extremities_of_line:nnnn` takes four arguments:

- the first argument is the row of the cell where the command was issued;
- the second argument is the column of the cell where the command was issued;
- the third argument is the  $x$ -value of the orientation vector of the line;
- the fourth argument is the  $y$ -value of the orientation vector of the line.

This command computes:

---

<sup>41</sup>e.g. `\color[rgb]{0.5,0.5,0}`

- `\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.

```
2127 \cs_new_protected:Npn \@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2128 {
```

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

```
2129 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
2130 \int_set:Nn \l_@@_initial_i_int { #1 }
2131 \int_set:Nn \l_@@_initial_j_int { #2 }
2132 \int_set:Nn \l_@@_final_i_int { #1 }
2133 \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.

```
2134 \bool_set_false:N \l_@@_stop_loop_bool
2135 \bool_do_until:Nn \l_@@_stop_loop_bool
2136 {
2137   \int_add:Nn \l_@@_final_i_int { #3 }
2138   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
2139   \bool_set_false:N \l_@@_final_open_bool
2140   \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
2141   {
2142     \int_compare:nNnTF { #3 } = 1
2143     { \bool_set_true:N \l_@@_final_open_bool }
2144     {
2145       \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2146       { \bool_set_true:N \l_@@_final_open_bool }
2147     }
2148   }
2149   {
2150     \int_compare:nNnTF \l_@@_final_j_int < 1
2151     {
2152       \int_compare:nNnT { #4 } = { -1 }
2153       { \bool_set_true:N \l_@@_final_open_bool }
2154     }
2155     {
2156       \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2157       {
2158         \int_compare:nNnT { #4 } = 1
2159         { \bool_set_true:N \l_@@_final_open_bool }
2160       }
2161     }
2162   }
2163   \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.

```
2164   {
```

We do a step backwards.

```
2165     \int_sub:Nn \l_@@_final_i_int { #3 }
2166     \int_sub:Nn \l_@@_final_j_int { #4 }
2167     \bool_set_true:N \l_@@_stop_loop_bool
2168   }
```

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`.

```

2169     {
2170         \cs_if_exist:cTF
2171         {
2172             @@ _ dotted _
2173             \int_use:N \l_@@_final_i_int -
2174             \int_use:N \l_@@_final_j_int
2175         }
2176         {
2177             \int_sub:Nn \l_@@_final_i_int { #3 }
2178             \int_sub:Nn \l_@@_final_j_int { #4 }
2179             \bool_set_true:N \l_@@_final_open_bool
2180             \bool_set_true:N \l_@@_stop_loop_bool
2181         }
2182     }
2183     \cs_if_exist:cTF
2184     {
2185         pgf @ sh @ ns @ \@@_env:
2186         - \int_use:N \l_@@_final_i_int
2187         - \int_use:N \l_@@_final_j_int
2188     }
2189     { \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.

```

2190     {
2191         \cs_set:cpn
2192         {
2193             @@ _ dotted _
2194             \int_use:N \l_@@_final_i_int -
2195             \int_use:N \l_@@_final_j_int
2196         }
2197         { }
2198     }
2199 }
2200 }
2201 }

```

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

```

2202 \bool_set_false:N \l_@@_stop_loop_bool
2203 \bool_do_until:Nn \l_@@_stop_loop_bool
2204 {
2205     \int_sub:Nn \l_@@_initial_i_int { #3 }
2206     \int_sub:Nn \l_@@_initial_j_int { #4 }
2207     \bool_set_false:N \l_@@_initial_open_bool
2208     \int_compare:nNnTF \l_@@_initial_i_int < 1
2209     {
2210         \int_compare:nNnTF { #3 } = 1
2211         { \bool_set_true:N \l_@@_initial_open_bool }
2212         {
2213             \int_compare:nNnT \l_@@_initial_j_int = 0
2214             { \bool_set_true:N \l_@@_initial_open_bool }
2215         }
2216     }
2217     {
2218         \int_compare:nNnTF \l_@@_initial_j_int < 1

```

```

2219     {
2220         \int_compare:nNt { #4 } = 1
2221         { \bool_set_true:N \l_@@_initial_open_bool }
2222     }
2223     {
2224         \int_compare:nNt \l_@@_initial_j_int > \c@jCol
2225         {
2226             \int_compare:nNt { #4 } = { -1 }
2227             { \bool_set_true:N \l_@@_initial_open_bool }
2228         }
2229     }
2230 }
2231 \bool_if:NTF \l_@@_initial_open_bool
2232 {
2233     \int_add:Nn \l_@@_initial_i_int { #3 }
2234     \int_add:Nn \l_@@_initial_j_int { #4 }
2235     \bool_set_true:N \l_@@_stop_loop_bool
2236 }
2237 {
2238     \cs_if_exist:cTF
2239     {
2240         @@ _ dotted _
2241         \int_use:N \l_@@_initial_i_int -
2242         \int_use:N \l_@@_initial_j_int
2243     }
2244     {
2245         \int_add:Nn \l_@@_initial_i_int { #3 }
2246         \int_add:Nn \l_@@_initial_j_int { #4 }
2247         \bool_set_true:N \l_@@_initial_open_bool
2248         \bool_set_true:N \l_@@_stop_loop_bool
2249     }
2250     {
2251         \cs_if_exist:cTF
2252         {
2253             pgf @ sh @ ns @ \@@_env:
2254             - \int_use:N \l_@@_initial_i_int
2255             - \int_use:N \l_@@_initial_j_int
2256         }
2257         { \bool_set_true:N \l_@@_stop_loop_bool }
2258         {
2259             \cs_set:cpn
2260             {
2261                 @@ _ dotted _
2262                 \int_use:N \l_@@_initial_i_int -
2263                 \int_use:N \l_@@_initial_j_int
2264             }
2265             { }
2266         }
2267     }
2268 }
2269 }

```

If the key `hvlines` is used, 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.

```

2270 \bool_if:NT \l_@@_hvlines_bool
2271 {
2272     \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2273     {
2274         { \int_use:N \l_@@_initial_i_int }
2275         { \int_use:N \l_@@_initial_j_int }
2276         { \int_use:N \l_@@_final_i_int }
2277         { \int_use:N \l_@@_final_j_int }
2278     }
2279 }

```



```

2280 }

2281 \cs_new_protected:Npn \@@_set_initial_coords:
2282 {
2283   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2284   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2285 }
2286 \cs_new_protected:Npn \@@_set_final_coords:
2287 {
2288   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2289   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2290 }
2291 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2292 {
2293   \pgfpointanchor
2294   {
2295     \@@_env:
2296     - \int_use:N \l_@@_initial_i_int
2297     - \int_use:N \l_@@_initial_j_int
2298   }
2299   { #1 }
2300   \@@_set_initial_coords:
2301 }
2302 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2303 {
2304   \pgfpointanchor
2305   {
2306     \@@_env:
2307     - \int_use:N \l_@@_final_i_int
2308     - \int_use:N \l_@@_final_j_int
2309   }
2310   { #1 }
2311   \@@_set_final_coords:
2312 }

```

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.

```

2313 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
2314 {
2315   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2316   {
2317     \@@_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.

```

2318   \group_begin:
2319   \int_compare:nNnTF { #1 } = 0
2320   { \color { nicematrix-first-row } }
2321   {

```

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.

```

2322     \int_compare:nNnT { #1 } = \l_@@_last_row_int
2323     { \color { nicematrix-last-row } }
2324   }
2325   \keys_set:nn { NiceMatrix / xdots } { #3 }
2326   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2327   \@@_actually_draw_Ldots:
2328   \group_end:
2329 }
2330 }

```

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.

```

2331 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2332 {
2333   \bool_if:NTF \l_@@_initial_open_bool
2334   {
2335     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2336     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2337     \dim_add:Nn \l_@@_x_initial_dim \@@_tab_or_array_colsep:
2338     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2339     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2340   }
2341   { \@@_set_initial_coords_from_anchor:n { base-east } }
2342   \bool_if:NTF \l_@@_final_open_bool
2343   {
2344     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2345     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2346     \dim_sub:Nn \l_@@_x_final_dim \@@_tab_or_array_colsep:
2347     \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2348     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2349   }
2350   { \@@_set_final_coords_from_anchor:n { base-west } }

```

We raise the line of a quantity equal to the radius of the dots because we want the dots really “on” the line of texte. Of course, maybe we should not do that when the option `line-style` is used (?).

```

2351   \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2352   \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2353   \@@_draw_line:
2354 }

```

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.

```

2355 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2356 {
2357   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2358   {
2359     \@@_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.

```

2360   \group_begin:
2361     \int_compare:nNnTF { #1 } = 0
2362     { \color { nicematrix-first-row } }
2363     {

```

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.

```

2364       \int_compare:nNnT { #1 } = \l_@@_last_row_int
2365       { \color { nicematrix-last-row } }
2366     }
2367     \keys_set:nn { NiceMatrix / xdots } { #3 }
2368     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2369     \@@_actually_draw_Cdots:

```

```

2370     \group_end:
2371   }
2372 }

```

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.`

```

2373 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2374 {
2375   \bool_if:NTF \l_@@_initial_open_bool
2376   {
2377     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2378     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2379     \dim_add:Nn \l_@@_x_initial_dim
2380     { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
2381   }
2382   { \@@_set_initial_coords_from_anchor:n { mid~east } }
2383   \bool_if:NTF \l_@@_final_open_bool
2384   {
2385     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2386     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2387     \dim_sub:Nn \l_@@_x_final_dim
2388     { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
2389   }
2390   { \@@_set_final_coords_from_anchor:n { mid~west } }
2391   \bool_lazy_and:nnTF
2392     \l_@@_initial_open_bool
2393     \l_@@_final_open_bool
2394   {
2395     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2396     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2397     \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2398     \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2399     \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2400   }
2401   {
2402     \bool_if:NT \l_@@_initial_open_bool
2403     { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2404     \bool_if:NT \l_@@_final_open_bool
2405     { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2406   }
2407   \@@_draw_line:
2408 }

```

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.

```

2409 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2410 {
2411   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2412   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2413   {
2414     \@@_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.

```

2415     \group_begin:
2416     \int_compare:nNnTF { #2 } = 0
2417     { \color { nicematrix-first-col } }
2418     {
2419         \int_compare:nNnT { #2 } = \l_@@_last_col_int
2420         { \color { nicematrix-last-col } }
2421     }
2422     \keys_set:nn { NiceMatrix / xdots } { #3 }
2423     \@@_actually_draw_Vdots:
2424 \group_end:
2425 }
2426 }
```

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`.

```

2427 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2428 {
```

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

```

2429     \bool_set_false:N \l_tmpa_bool
2430     \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2431     {
2432         \@@_set_initial_coords_from_anchor:n { south-west }
2433         \@@_set_final_coords_from_anchor:n { north-west }
2434         \bool_set:Nn \l_tmpa_bool
2435         { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2436     }
```

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

```

2437     \bool_if:NTF \l_@@_initial_open_bool
2438     {
2439         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2440         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2441     }
2442     { \@@_set_initial_coords_from_anchor:n { south } }
2443     \bool_if:NTF \l_@@_final_open_bool
2444     {
2445         \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2446         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2447     }
2448     { \@@_set_final_coords_from_anchor:n { north } }
2449     \bool_if:NTF \l_@@_initial_open_bool
2450     {
2451         \bool_if:NTF \l_@@_final_open_bool
2452         {
2453             \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2454             \dim_set_eq:NN \l_tmpa_dim \pgf@x
2455             \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2456             \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2457             \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.

```

2458     \int_compare:nNnT \l_@@_last_col_int > { -2 }
2459     {
2460         \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2461         {
2462             \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2463             \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2464             \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2465             \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2466         }
2467     }
2468 }
2469 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2470 }
2471 {
2472     \bool_if:NTF \l_@@_final_open_bool
2473     { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2474     {

```

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.

```

2475     \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2476     {
2477         \dim_set:Nn \l_@@_x_initial_dim
2478         {
2479             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2480             \l_@@_x_initial_dim \l_@@_x_final_dim
2481         }
2482         \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2483     }
2484 }
2485 }
2486 \@@_draw_line:
2487 }

```

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.

```

2488 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2489 {
2490     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2491     {
2492         \@@_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.

```

2493     \group_begin:
2494     \keys_set:nn { NiceMatrix / xdots } { #3 }
2495     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2496     \@@_actually_draw_Ddots:
2497     \group_end:
2498 }
2499 }

```

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`.

```

2500 \cs_new_protected:Npn \l_@@_actually_draw_Ddots:
2501 {
2502   \bool_if:NTF \l_@@_initial_open_bool
2503   {
2504     \l_@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2505     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2506     \l_@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2507     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2508   }
2509   { \l_@@_set_initial_coords_from_anchor:n { south-east } }
2510   \bool_if:NTF \l_@@_final_open_bool
2511   {
2512     \l_@@_qpoint:n { row - \l_@@_succ:n \l_@@_final_i_int }
2513     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2514     \l_@@_qpoint:n { col - \l_@@_succ:n \l_@@_final_j_int }
2515     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2516   }
2517   { \l_@@_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.

```

2518   \bool_if:NT \l_@@_parallelize_diags_bool
2519   {
2520     \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).

```

2521     \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.

```

2522     {
2523       \dim_gset:Nn \g_@@_delta_x_one_dim
2524       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2525       \dim_gset:Nn \g_@@_delta_y_one_dim
2526       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2527     }

```

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`.

```

2528     {
2529       \dim_set:Nn \l_@@_y_final_dim
2530       {
2531         \l_@@_y_initial_dim +
2532         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2533         \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
2534       }
2535     }
2536   }
2537   \l_@@_draw_line:
2538 }

```

We draw the `\l_@@_draw_line` 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.

```

2539 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2540 {
2541   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2542   {
2543     \@@_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.

```

2544     \group_begin:
2545     \keys_set:nn { NiceMatrix / xdots } { #3 }
2546     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2547     \@@_actually_draw_Iddots:
2548     \group_end:
2549   }
2550 }

```

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`.

```

2551 \cs_new_protected:Npn \@@_actually_draw_Iddots:
2552 {
2553   \bool_if:NTF \l_@@_initial_open_bool
2554   {
2555     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2556     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2557     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2558     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2559   }
2560   { \@@_set_initial_coords_from_anchor:n { south-west } }
2561   \bool_if:NTF \l_@@_final_open_bool
2562   {
2563     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2564     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2565     \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2566     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2567   }
2568   { \@@_set_final_coords_from_anchor:n { north-east } }
2569   \bool_if:NT \l_@@_parallelize_diags_bool
2570   {
2571     \int_gincr:N \g_@@_iddots_int
2572     \int_compare:nNnTF \g_@@_iddots_int = 1
2573     {
2574       \dim_gset:Nn \g_@@_delta_x_two_dim
2575       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2576       \dim_gset:Nn \g_@@_delta_y_two_dim
2577       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2578     }
2579     {
2580       \dim_set:Nn \l_@@_y_final_dim
2581       {
2582         \l_@@_y_initial_dim +
2583         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2584         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2585       }

```

```

2586     }
2587   }
2588   \@@_draw_line:
2589 }

```

## 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`

```

2590 \cs_new_protected:Npn \@@_draw_line:
2591 {
2592   \pgfrememberpicturepositiononpagetrue
2593   \pgf@relevantforpicturesizefalse
2594   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
2595     \@@_draw_standard_dotted_line:
2596     \@@_draw_non_standard_dotted_line:
2597 }

```

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.

```

2598 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2599 {
2600   \begin { scope }
2601   \exp_args:No \@@_draw_non_standard_dotted_line:n
2602     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2603 }

```

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.

```

2604 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2605 {
2606   \draw
2607   [
2608     #1 ,
2609     shorten-> = \l_@@_xdots_shorten_dim ,
2610     shorten-< = \l_@@_xdots_shorten_dim ,
2611   ]
2612     ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
2613     -- node [ sloped , above ]
2614       { \c_math_toggle_token \scriptstyle \l_@@_xdots_up_tl \c_math_toggle_token }
2615     node [ sloped , below ]
2616       {
2617         \c_math_toggle_token
2618         \scriptstyle \l_@@_xdots_down_tl
2619         \c_math_toggle_token
2620       }
2621     ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2622   \end { scope }
2623 }

```



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

```
2624 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2625 {
```

First, we put the labels.

```
2626   \bool_lazy_and:nnF
2627   { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2628   { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2629   {
2630     \pgfscope
2631     \pgftransformshift
2632     {
2633       \pgfpointlineatime { 0.5 }
2634       { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2635       { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2636     }
2637     \pgftransformrotate
2638     {
2639       \fp_eval:n
2640       {
2641         atand
2642         (
2643           \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2644           \l_@@_x_final_dim - \l_@@_x_initial_dim
2645         )
2646       }
2647     }
2648     \pgfnode
2649     { rectangle }
2650     { south }
2651     {
2652       \c_math_toggle_token
2653       \scriptstyle \l_@@_xdots_up_tl
2654       \c_math_toggle_token
2655     }
2656     { }
2657     { \pgfusepath { } }
2658     \pgfnode
2659     { rectangle }
2660     { north }
2661     {
2662       \c_math_toggle_token
2663       \scriptstyle \l_@@_xdots_down_tl
2664       \c_math_toggle_token
2665     }
2666     { }
2667     { \pgfusepath { } }
2668     \endpgfscope
2669   }
2670   \pgfrememberpicturepositiononpagetrue
2671   \pgf@relevantforpicturesizefalse
2672   \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.

```
2673   \dim_zero_new:N \l_@@_l_dim
2674   \dim_set:Nn \l_@@_l_dim
2675   {
2676     \fp_to_dim:n
2677     {
2678       sqrt
2679       (
2680         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
```

```

2681      +
2682      ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2683    )
2684  }
2685 }

```

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.

```

2686   \bool_lazy_or:nnF
2687   { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2688   { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2689   \@@_draw_standard_dotted_line_i:
2690 \group_end:
2691 }
2692 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2693 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2694 {

```

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

```

2695   \bool_if:NTF \l_@@_initial_open_bool
2696   {
2697     \bool_if:NTF \l_@@_final_open_bool
2698     {
2699       \int_set:Nn \l_tmpa_int
2700       { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2701     }
2702     {
2703       \int_set:Nn \l_tmpa_int
2704       {
2705         \dim_ratio:nn
2706         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2707         \l_@@_inter_dots_dim
2708       }
2709     }
2710   }
2711   {
2712     \bool_if:NTF \l_@@_final_open_bool
2713     {
2714       \int_set:Nn \l_tmpa_int
2715       {
2716         \dim_ratio:nn
2717         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2718         \l_@@_inter_dots_dim
2719       }
2720     }
2721     {
2722       \int_set:Nn \l_tmpa_int
2723       {
2724         \dim_ratio:nn
2725         { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2726         \l_@@_inter_dots_dim
2727       }
2728     }
2729   }

```

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

```

2730   \dim_set:Nn \l_tmpa_dim
2731   {
2732     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2733     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim

```

```

2734     }
2735     \dim_set:Nn \l_tmpb_dim
2736     {
2737         ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2738         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2739     }

```

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`.

```

2740     \int_set:Nn \l_tmpb_int
2741     {
2742         \bool_if:NTF \l_@@_initial_open_bool
2743         { \bool_if:NTF \l_@@_final_open_bool 1 0 }
2744         { \bool_if:NTF \l_@@_final_open_bool 2 1 }
2745     }

```

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.

```

2746     \dim_gadd:Nn \l_@@_x_initial_dim
2747     {
2748         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2749         \dim_ratio:nn
2750         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2751         { 2 \l_@@_l_dim }
2752         * \l_tmpb_int
2753     }
2754     \dim_gadd:Nn \l_@@_y_initial_dim
2755     {
2756         ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2757         \dim_ratio:nn
2758         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2759         { 2 \l_@@_l_dim }
2760         * \l_tmpb_int
2761     }
2762     \pgf@relevantforpicturesizefalse
2763     \int_step_inline:nnn 0 \l_tmpa_int
2764     {
2765         \pgfpathcircle
2766         { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2767         { \l_@@_radius_dim }
2768         \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2769         \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
2770     }
2771     \pgfusepathqfill
2772 }

```

## User commands available in the new environments

The commands `\@@_Ldots`, `\@@_Cdots`, `\@@_Vdots`, `\@@_Ddots` and `\@@_Iddots` will be linked to `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` and `\Iddots` in the environments `{NiceArray}` (the other environments of `nicematrix` rely upon `{NiceArray}`).

The starred versions of these commands are deprecated since version 3.1 but, as of now, they are still available with an error.

The syntax of these commands uses the character `_` as embellishment and that's why we have to insert a character `_` in the *arg spec* of these commands. However, we don't know the future catcode of `_` in the main document (maybe the user will use `\underscore`, and, in that case, the catcode is 13

because underscore activates `_`). That's why these commands will be defined in a `\AtBeginDocument` and the *arg spec* will be rescanned.

```

2773 \AtBeginDocument
2774 {
2775   \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
2776   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2777   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
2778     {
2779       \int_compare:nNnTF \c@jCol = 0
2780       { \@@_error:nn { in~first~col } \Ldots }
2781       {
2782         \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2783         { \@@_error:nn { in~last~col } \Ldots }
2784         {
2785           \@@_instruction_of_type:nn { Ldots }
2786           { #1 , down = #2 , up = #3 }
2787         }
2788       }
2789       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ldots }
2790       \bool_gset_true:N \g_@@_empty_cell_bool
2791     }

2792   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
2793     {
2794       \int_compare:nNnTF \c@jCol = 0
2795       { \@@_error:nn { in~first~col } \Cdots }
2796       {
2797         \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2798         { \@@_error:nn { in~last~col } \Cdots }
2799         {
2800           \@@_instruction_of_type:nn { Cdots }
2801           { #1 , down = #2 , up = #3 }
2802         }
2803       }
2804       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_cdots }
2805       \bool_gset_true:N \g_@@_empty_cell_bool
2806     }

2807   \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
2808     {
2809       \int_compare:nNnTF \c@iRow = 0
2810       { \@@_error:nn { in~first~row } \Vdots }
2811       {
2812         \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
2813         { \@@_error:nn { in~last~row } \Vdots }
2814         {
2815           \@@_instruction_of_type:nn { Vdots }
2816           { #1 , down = #2 , up = #3 }
2817         }
2818       }
2819       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_vdots }
2820       \bool_gset_true:N \g_@@_empty_cell_bool
2821     }

2822   \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
2823     {
2824       \int_case:nnF \c@iRow
2825       {
2826         0 { \@@_error:nn { in~first~row } \Ddots }
2827         \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }

```

```

2828     }
2829     {
2830         \int_case:nnF \c@jCol
2831         {
2832             0 { \@@_error:nn { in~first~col } \Ddots }
2833             \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
2834         }
2835         {
2836             \@@_instruction_of_type:nn { Ddots }
2837             { #1 , down = #2 , up = #3 }
2838         }
2839     }
2840 }
2841 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ddots }
2842 \bool_gset_true:N \g_@@_empty_cell_bool
2843 }

2844 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
2845 {
2846     \int_case:nnF \c@iRow
2847     {
2848         0 { \@@_error:nn { in~first~row } \Iddots }
2849         \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
2850     }
2851     {
2852         \int_case:nnF \c@jCol
2853         {
2854             0 { \@@_error:nn { in~first~col } \Iddots }
2855             \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
2856         }
2857         {
2858             \@@_instruction_of_type:nn { Iddots }
2859             { #1 , down = #2 , up = #3 }
2860         }
2861     }
2862 }
2863 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_iddots }
2864 \bool_gset_true:N \g_@@_empty_cell_bool
2865 }
2866 }

```

End of the \AtBeginDocument.

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

```

2867 \cs_new_protected:Npn \@@_Hspace:
2868 {
2869     \bool_gset_true:N \g_@@_empty_cell_bool
2870     \hspace
2871 }

```

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

```

2872 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
2873 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
2874 {
2875     % \begin{macrocode}
2876     % We have to act in expandable way since it will begin by a |\multicolumn|.
2877     % \end{macrocode}
2878     \exp_args:NNe
2879     \@@_old_multicolumn
2880     { #1 }

```

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

```

2881 {
2882   > \@@_Cell:
2883   \bool_if:NT \c_@@_define_L_C_R_bool \tl_lower_case:n
2884   #2
2885   < \@@_end_Cell:
2886 }
2887 { #3 }

```

The `\peek_remove_spaces:n` is mandatory.

```

2888 \peek_remove_spaces:n
2889 {
2890   \int_compare:nNnT #1 > 1
2891   {
2892     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
2893     { \int_use:N \c@iRow - \int_use:N \c@jCol }
2894     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
2895     \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
2896     {
2897       { \int_use:N \c@iRow }
2898       { \int_use:N \c@jCol }
2899       { \int_use:N \c@iRow }
2900       { \int_eval:n { \c@jCol + #1 - 1 } }
2901     }
2902   }
2903   \int_gadd:Nn \c@jCol { #1 - 1 }
2904 }
2905 }

```

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`.

```

2906 \cs_new:Npn \@@_Hdotsfor:
2907 {
2908   \multicolumn { 1 } { c } { }
2909   \@@_Hdotsfor_i
2910 }

```

The command `\@@_Hdotsfor_i` is defined with the tools of `xparse` 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:`).

```

2911 \AtBeginDocument
2912 {
2913   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2914   \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.

```

2915   \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
2916   {
2917     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
2918     {
2919       \@@_Hdotsfor:nnnn
2920       { \int_use:N \c@iRow }
2921       { \int_use:N \c@jCol }
2922       { #2 }
2923       {
2924         #1 , #3 ,
2925         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
2926       }
2927     }
2928     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }

```

```

2929     }
2930 }

```

Enf of \AtBeginDocument.

```

2931 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
2932 {
2933     \bool_set_false:N \l_@@_initial_open_bool
2934     \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

2935     \int_set:Nn \l_@@_initial_i_int { #1 }
2936     \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

2937     \int_compare:nNnTF #2 = 1
2938     {
2939         \int_set:Nn \l_@@_initial_j_int 1
2940         \bool_set_true:N \l_@@_initial_open_bool
2941     }
2942     {
2943         \cs_if_exist:cTF
2944         {
2945             pgf @ sh @ ns @ \@@_env:
2946             - \int_use:N \l_@@_initial_i_int
2947             - \int_eval:n { #2 - 1 }
2948         }
2949         { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
2950         {
2951             \int_set:Nn \l_@@_initial_j_int { #2 }
2952             \bool_set_true:N \l_@@_initial_open_bool
2953         }
2954     }
2955     \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
2956     {
2957         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2958         \bool_set_true:N \l_@@_final_open_bool
2959     }
2960     {
2961         \cs_if_exist:cTF
2962         {
2963             pgf @ sh @ ns @ \@@_env:
2964             - \int_use:N \l_@@_final_i_int
2965             - \int_eval:n { #2 + #3 }
2966         }
2967         { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
2968         {
2969             \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2970             \bool_set_true:N \l_@@_final_open_bool
2971         }
2972     }
2973     \group_begin:
2974     \int_compare:nNnTF { #1 } = 0
2975     { \color { nicematrix-first-row } }
2976     {
2977         \int_compare:nNnT { #1 } = \g_@@_row_total_int
2978         { \color { nicematrix-last-row } }
2979     }
2980     \keys_set:nn { NiceMatrix / xdots } { #4 }
2981     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2982     \@@_actually_draw_Ldots:
2983     \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).

```

2984   \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
2985   { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
2986 }

2987 \AtBeginDocument
2988 {
2989   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2990   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2991   \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
2992   {
2993     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
2994     {
2995       \@@_Vdotsfor:nnnn
2996       { \int_use:N \c@iRow }
2997       { \int_use:N \c@jCol }
2998       { #2 }
2999       {
3000         #1 , #3 ,
3001         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3002       }
3003     }
3004   }
3005 }

```

Enf of `\AtBeginDocument`.

```

3006 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
3007 {
3008   \bool_set_false:N \l_@@_initial_open_bool
3009   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it’s easy.

```

3010   \int_set:Nn \l_@@_initial_j_int { #2 }
3011   \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

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

```

3012   \int_compare:nNnTF #1 = 1
3013   {
3014     \int_set:Nn \l_@@_initial_i_int 1
3015     \bool_set_true:N \l_@@_initial_open_bool
3016   }
3017   {
3018     \cs_if_exist:cTF
3019     {
3020       pgf @ sh @ ns @ \@@_env:
3021       - \int_eval:n { #1 - 1 }
3022       - \int_use:N \l_@@_initial_j_int
3023     }
3024     { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
3025     {
3026       \int_set:Nn \l_@@_initial_i_int { #1 }
3027       \bool_set_true:N \l_@@_initial_open_bool
3028     }
3029   }
3030   \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
3031   {
3032     \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3033     \bool_set_true:N \l_@@_final_open_bool
3034   }
3035   {
3036     \cs_if_exist:cTF

```



```

3037     {
3038         pgf @ sh @ ns @ \@@_env:
3039         - \int_eval:n { #1 + #3 }
3040         - \int_use:N \l_@@_final_j_int
3041     }
3042     { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
3043     {
3044         \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3045         \bool_set_true:N \l_@@_final_open_bool
3046     }
3047 }
3048 \group_begin:
3049 \int_compare:nNnTF { #2 } = 0
3050 { \color { nicematrix-first-col } }
3051 {
3052     \int_compare:nNnT { #2 } = \g_@@_col_total_int
3053     { \color { nicematrix-last-col } }
3054 }
3055 \keys_set:nn { NiceMatrix / xdots } { #4 }
3056 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3057 \@@_actually_draw_Vdots:
3058 \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).

```

3059     \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
3060     { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
3061 }

```

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

The command will exit three levels of groups (only two in `{NiceTabular}` because there is not the group of the math mode to exit) in order to execute the command

“`\box_rotate:Nn \l_@@_cell_box { 90 }`”

just after the construction of the box `\l_@@_cell_box`.

```

3062 \cs_new_protected:Npn \@@_rotate:
3063 {
3064     \bool_if:NTF \l_@@_NiceTabular_bool
3065     { \group_insert_after:N \@@_rotate_ii: }
3066     { \group_insert_after:N \@@_rotate_i: }
3067 }
3068 \cs_new_protected:Npn \@@_rotate_i: { \group_insert_after:N \@@_rotate_ii: }
3069 \cs_new_protected:Npn \@@_rotate_ii: { \group_insert_after:N \@@_rotate_iii: }
3070 \cs_new_protected:Npn \@@_rotate_iii:
3071 {
3072     \box_rotate:Nn \l_@@_cell_box { 90 }

```

If we are in the last row, we want all the boxes composed with the command `\rotate` aligned upwards.

```

3073     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
3074     {
3075         \vbox_set_top:Nn \l_@@_cell_box
3076         {
3077             \vbox_to_zero:n { }

```

0.8 ex will be the distance between the principal part of the array and our element (which is composed with `\rotate`).

```

3078         \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
3079         \box_use:N \l_@@_cell_box
3080     }
3081 }
3082 }

```

## 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>42</sup>

```
3083 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
3084   { \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).

```
3085 \AtBeginDocument
3086 {
3087   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
3088   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3089   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
3090     {
3091       \group_begin:
3092       \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
3093       \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3094       \use:x
3095       {
3096         \@@_line_i:nn
3097         { \@@_double_int_eval:n #2 \q_stop }
3098         { \@@_double_int_eval:n #3 \q_stop }
3099       }
3100       \group_end:
3101     }
3102 }

3103 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3104 {
3105   \bool_set_false:N \l_@@_initial_open_bool
3106   \bool_set_false:N \l_@@_final_open_bool
3107   \bool_if:nTF
3108   {
3109     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3110     ||
3111     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3112   }
3113   {
3114     \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
3115   }
3116   { \@@_draw_line_ii:nn { #1 } { #2 } }
3117 }

3118 \AtBeginDocument
3119 {
3120   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3121   {
```

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:.`

```
3122   \c_@@_pgfortikzpicture_tl
3123   \@@_draw_line_iii:nn { #1 } { #2 }
3124   \c_@@_endpgfortikzpicture_tl
3125 }
3126 }
```

---

<sup>42</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`.

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

```

3127 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3128 {
3129   \pgfrememberpicturepositiononpagetrue
3130   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3131   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3132   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3133   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3134   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3135   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3136   \@@_draw_line:
3137 }

```

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`.

```

3138 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3139 {
3140   \tl_set:Nn \l_tmpa_tl { #1 }
3141   \tl_set:Nn \l_tmpb_tl { #2 }
3142 }

```

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

```

3143 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3144 {
3145   \tl_if_blank:nF { #2 }
3146   {
3147     \pgfpicture
3148     \pgf@relevantforpicturesizefalse
3149     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }

```

`\l_tmpa_dim` is the  $x$ -value of the right side of the rows.

```

3150     \@@_qpoint:n { col - 1 }
3151     \int_compare:nNnTF \l_@@_first_col_int = 0
3152     { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3153     { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3154     \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3155     \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }
3156     \clist_map_inline:nn { #3 }
3157     {
3158       \tl_set:Nn \l_tmpa_tl { ##1 }
3159       \tl_if_in:NnTF \l_tmpa_tl { - }
3160       { \@@_cut_on_hyphen:w ##1 \q_stop }
3161       { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3162       \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3163       \tl_if_empty:NT \l_tmpb_tl
3164       { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3165       \int_compare:nNnT \l_tmpb_tl > \c@iRow
3166       { \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`.

```

3167     \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3168     \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3169     \@@_qpoint:n { row - \l_tmpa_tl }
3170     \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3171     \pgfpathrectanglecorners
3172     { \pgfpoint \l_tmpc_dim \l_tmpd_dim }

```

```

3173         { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3174     }
3175     \pgfusepathqfill
3176     \endpgfpicture
3177 }
3178 }

```

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

```

3179 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3180 {
3181     \tl_if_blank:nF { #2 }
3182     {
3183         \pgfpicture
3184         \pgf@relevantforpicturesizefalse
3185         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3186         \@@_qpoint:n { row - 1 }

```

`\l_tmpa_dim` is the  $y$ -value of the top of the columns et `\l_tmpb_dim` is the  $y$ -value of the bottom.

```

3187         \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3188         \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3189         \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3190         \clist_map_inline:nn { #3 }
3191         {
3192             \tl_set:Nn \l_tmpa_tl { ##1 }
3193             \tl_if_in:NnTF \l_tmpa_tl { - }
3194             { \@@_cut_on_hyphen:w ##1 \q_stop }
3195             { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3196             \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3197             \tl_if_empty:NT \l_tmpb_tl
3198             { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3199             \int_compare:nNnT \l_tmpb_tl > \c@jCol
3200             { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }

```

Now, the numbers of both columns are in `\l_tmpa_tl` and `\l_tmpb_tl`.

```

3201         \@@_qpoint:n { col - \l_tmpa_tl }
3202         \int_compare:nNnTF \l_@@_first_col_int = \l_tmpa_tl
3203         { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3204         { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3205         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3206         \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3207         \pgfpathrectanglecorners
3208         { \pgfpoint \l_tmpc_dim \l_tmpa_dim }
3209         { \pgfpoint \l_tmpd_dim \l_tmpb_dim }
3210     }
3211     \pgfusepathqfill
3212     \endpgfpicture
3213 }
3214 }

```

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

```

3215 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3216 {
3217     \tl_if_blank:nF { #2 }
3218     {
3219         \pgfpicture
3220         \pgf@relevantforpicturesizefalse
3221         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3222         \clist_map_inline:nn { #3 }
3223         {
3224             \@@_cut_on_hyphen:w ##1 \q_stop
3225             \@@_qpoint:n { row - \l_tmpa_tl }
3226             \bool_lazy_and:nnT
3227             { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }

```

```

3228 { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
3229 {
3230   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3231   \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3232   \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3233   \@@_qpoint:n { col - \l_tmpb_tl }
3234   \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
3235     { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3236     { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3237   \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3238   \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3239   \pgfpathrectanglecorners
3240     { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3241     { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3242   }
3243 }
3244 \pgfusepathqfill
3245 \endpgfpicture
3246 }
3247 }

```

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

```

3248 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3249 {
3250   \tl_if_blank:nF { #2 }
3251   {
3252     \pgfpicture
3253     \pgf@relevantforpicturesizefalse
3254     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3255     \@@_cut_on_hyphen:w #3 \q_stop
3256     \bool_lazy_and:nnT
3257       { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
3258       { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
3259     {
3260       \@@_qpoint:n { row - \l_tmpa_tl }
3261       \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3262       \@@_qpoint:n { col - \l_tmpb_tl }
3263       \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
3264         { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3265         { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3266       \@@_cut_on_hyphen:w #4 \q_stop
3267       \int_compare:nNnT \l_tmpa_tl > \c@iRow
3268         { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
3269       \int_compare:nNnT \l_tmpb_tl > \c@jCol
3270         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3271       \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3272       \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3273       \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3274       \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3275       \pgfpathrectanglecorners
3276         { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3277         { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3278       \pgfusepathqfill
3279     }
3280   \endpgfpicture
3281 }
3282 }

```

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`.

```

3283 \NewDocumentCommand \@@_rowcolors { 0 { } m m m }

```

```

3284 {
3285   \int_step_inline:nnn { #2 } { \int_use:N \c@iRow }
3286   {
3287     \int_if_odd:nTF { ##1 }
3288     { \@@_rowcolor [ #1 ] { #3 } }
3289     { \@@_rowcolor [ #1 ] { #4 } }
3290     { ##1 }
3291   }
3292 }

3293 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3294 {
3295   \int_step_inline:nn { \int_use:N \c@iRow }
3296   {
3297     \int_step_inline:nn { \int_use:N \c@jCol }
3298     {
3299       \int_if_even:nTF { ####1 + ##1 }
3300       { \@@_cellcolor [ #1 ] { #2 } }
3301       { \@@_cellcolor [ #1 ] { #3 } }
3302       { ##1 - ####1 }
3303     }
3304   }
3305 }

```

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

```

3306 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
3307 {
3308   \tl_gput_right:Nx \g_@@_code_before_tl
3309   { \cellcolor [ #1 ] { #2 } { \int_use:N \c@iRow - \int_use:N \c@jCol } }
3310 }

```

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

```

3311 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
3312 {
3313   \tl_gput_right:Nx \g_@@_code_before_tl
3314   {
3315     \exp_not:N \rectanglecolor [ #1 ] { #2 }
3316     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3317     { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
3318   }
3319 }

```

```

3320 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
3321 {
3322   \int_compare:nNnT \c@iRow = 1
3323   {

```

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).

```

3324   \tl_gput_left:Nx \g_@@_code_before_tl
3325   { \exp_not:N \columncolor [ #1 ] { #2 } { \int_use:N \c@jCol } }
3326 }
3327 }

```

## 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`.

```
3328 \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`.

```
3329 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
3330 {
3331   \int_compare:nNnTF \l_@@_first_col_int = 0
3332   { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3333   {
3334     \int_compare:nNnTF \c@jCol = 0
3335     {
3336       \int_compare:nNnF \c@iRow = { -1 }
3337       { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
3338     }
3339     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3340   }
3341 }
```

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.

```
3342 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
3343 {
3344   \int_compare:nNnF \c@iRow = 0
3345   { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
3346 }
```

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`. `#1` is the number the column where to draw the rule and `#2` is the number of consecutive occurrences of `|`.

```
3347 \cs_new_protected:Npn \@@_vline:nn #1 #2
3348 {
3349   \bool_if:NTF \c_@@_tikz_loaded_bool
3350   {
3351     \tikzpicture
3352     \@@_vline_i:nn { #1 } { #2 }
3353     \endtikzpicture
3354   }
3355   {
3356     \pgfpicture
3357     \@@_vline_i:nn { #1 } { #2 }
3358     \endpgfpicture
3359   }
3360 }
3361 \cs_new_protected:Npn \@@_vline_i:nn #1 #2
3362 {
3363   \CT@arc@
3364   \pgfrememberpicturepositiononpagetrue
3365   \pgf@relevantforpicturesizefalse
```

```

3366 \pgfsetlinewidth { 1.1 \arrayrulewidth }
3367 \pgfsetrectcap
3368 \@@_qpoint:n { row - 1 }
3369 \dim_set_eq:NN \l_tmpa_dim \pgf@y
3370 \@@_qpoint:n { col - \int_eval:n { #1 + 1 } }
3371 \dim_set_eq:NN \l_tmpb_dim \pgf@x
3372 \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3373 \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3374 \dim_set_eq:NN \l_tmpc_dim \pgf@y
3375 \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3376 \prg_replicate:nn { #2 - 1 }
3377 {
3378   \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3379   \dim_sub:Nn \l_tmpb_dim \doublerulesep
3380   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3381   \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3382 }
3383 \pgfusepathqstroke
3384 }

```

The command `\@@_draw_vlines` will be executed when the user uses the option `vlines` (which draws all the vlines of the array).

```

3385 \cs_new_protected:Npn \@@_draw_vlines:
3386 {
3387   \group_begin:

```

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 when `colortbl` is not loaded.

```

3388 \CT@arc@
3389 \pgfpicture
3390 \pgfrememberpicturepositiononpagetrue
3391 \pgf@relevantforpicturesizefalse
3392 \pgfsetlinewidth \arrayrulewidth
3393 \pgfsetrectcap
3394 \@@_qpoint:n { row - 1 }
3395 \dim_set_eq:NN \l_tmpa_dim \pgf@y
3396 \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3397 \dim_set_eq:NN \l_tmpb_dim \pgf@x

```

Now, we can draw the vertical rules with a loop.

```

3398 \int_step_inline:nnn
3399 { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3400 { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3401 {
3402   \@@_qpoint:n { col - ##1 }
3403   \dim_set_eq:NN \l_tmpc_dim \pgf@x
3404   \pgfpathmoveto { \pgfpoint \l_tmpc_dim \l_tmpa_dim }
3405   \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3406 }
3407 \pgfusepathqstroke
3408 \endpgfpicture
3409 \group_end:
3410 }

```

## The key hvlines

### The key hvlines

```

3411 \cs_new_protected:Npn \@@_draw_hlines:
3412 {
3413   \pgfpicture
3414   \CT@arc@
3415   \pgfrememberpicturepositiononpagetrue

```



```

3416 \pgf@relevantforpicturesizefalse
3417 \pgfsetlinewidth \arrayrulewidth
3418 \int_step_inline:nnn
3419 { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3420 { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3421 {
3422   \@@_qpoint:n { row - ##1 }
3423   \dim_set_eq:NN \l_tmpa_dim \pgf@y
3424   \pgfpathmoveto { \pgfpoint \pgf@x \pgf@y }
3425   \@@_qpoint:n { col - \@@_succ:n { \c@jCol } }
3426   \dim_set:Nn \l_tmpb_dim { \pgf@x + 0.5 \arrayrulewidth }
3427   \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3428 }
3429 \pgfusepathqstroke
3430 \endpgfpicture
3431 }

```

Since version 4.1, the key `hvlines` is no longer a mere alias for the conjunction of `hlines` and `vlines`. Indeed, with `hvlines`, the vertical and horizontal rules are *not* drawn within the blocks (created by `\Block`) nor within the “virtual blocks” (corresponding to the dotted lines drawn by `\Cdots`, `\Vdots`, etc.).

```

3432 \cs_new_protected:Npn \@@_draw_hvlines:
3433 {
3434   \bool_lazy_and:nnTF
3435   { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
3436   { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
3437   \@@_draw_hvlines_i:
3438   \@@_draw_hvlines_ii:
3439 }

```

This version is only for efficiency. The general case (in `\@@_draw_hvlines_ii:`) does the job in all case (but slower).

```

3440 \cs_new_protected:Npn \@@_draw_hvlines_i:
3441 {
3442   \@@_draw_hlines:
3443   \@@_draw_vlines:
3444 }

```

Now, the general case, where there are blocks or dots in the array.

```

3445 \cs_new_protected:Npn \@@_draw_hvlines_ii:
3446 {
3447   \group_begin:
3448   \CT@arc@

```

The horizontal rules.

```

3449 \int_step_variable:nnNn
3450 { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3451 { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3452 \l_tmpa_tl
3453 {
3454   \int_step_variable:nnNn \c@jCol \l_tmpb_tl
3455   {

```

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.

```

3456 \bool_gset_true:N \g_tmpa_bool
3457 \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3458 { \@@_test_if_hline_in_block:nnnn ##1 }
3459 \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3460 { \@@_test_if_hline_in_block:nnnn ##1 }
3461 \bool_if:NT \l_@@_hvlines_except_corners_bool

```

```

3462 {
3463   \int_compare:nNnTF \l_tmpa_tl = { \c@iRow + 1 }
3464   {
3465     \seq_if_in:NxT
3466     \l_@@_empty_corner_cells_seq
3467     { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3468     { \bool_set_false:N \g_tmpa_bool }
3469   }
3470   {
3471     \seq_if_in:NxT
3472     \l_@@_empty_corner_cells_seq
3473     { \l_tmpa_tl - \l_tmpb_tl }
3474     {
3475       \int_compare:nNnTF \l_tmpa_tl = 1
3476       { \bool_set_false:N \g_tmpa_bool }
3477       {
3478         \seq_if_in:NxT
3479         \l_@@_empty_corner_cells_seq
3480         { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3481         { \bool_set_false:N \g_tmpa_bool }
3482       }
3483     }
3484   }
3485 }
3486 \bool_if:NT \g_tmpa_bool
3487 {
3488   \pgfpicture
3489   \pgfrememberpicturepositiononpagetrue
3490   \pgf@relevantforpicturesizefalse
3491   \pgfsetlinewidth \arrayrulewidth
3492   \pgfsetrectcap
3493   \@@_qpoint:n { row - \l_tmpa_tl }
3494   \dim_set_eq:NN \l_tmpb_dim \pgf@y
3495   \@@_qpoint:n { col - \l_tmpb_tl }
3496   \dim_set_eq:NN \l_tmpa_dim \pgf@x
3497   \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3498   \dim_set_eq:NN \l_tmpc_dim \pgf@x
3499   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3500   \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3501   \pgfusepathqstroke
3502   \endpgfpicture
3503 }
3504 }
3505 }

```

Now, the vertical rules.

```

3506 \int_step_variable:nNn \c@iRow \l_tmpa_tl
3507 {
3508   \int_step_variable:nnNn
3509   { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3510   { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3511   \l_tmpb_tl
3512   {

```

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.

```

3513   \bool_gset_true:N \g_tmpa_bool
3514   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3515   { \@@_test_if_vline_in_block:nnnn ##1 }
3516   \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3517   { \@@_test_if_vline_in_block:nnnn ##1 }
3518   \bool_if:NT \l_@@_hvlines_except_corners_bool

```

```

3519 {
3520   \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
3521   {
3522     \seq_if_in:NxT
3523     \l_@@_empty_corner_cells_seq
3524     { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3525     { \bool_set_false:N \g_tmpa_bool }
3526   }
3527   {
3528     \seq_if_in:NxT
3529     \l_@@_empty_corner_cells_seq
3530     { \l_tmpa_tl - \l_tmpb_tl }
3531     {
3532       \int_compare:nNnTF \l_tmpb_tl = 1
3533       { \bool_set_false:N \g_tmpa_bool }
3534       {
3535         \seq_if_in:NxT
3536         \l_@@_empty_corner_cells_seq
3537         { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3538         { \bool_set_false:N \g_tmpa_bool }
3539       }
3540     }
3541   }
3542 }
3543 \bool_if:NT \g_tmpa_bool
3544 {
3545   \pgfpicture
3546   \pgfrememberpicturepositiononpagetrue
3547   \pgf@relevantforpicturesizefalse
3548   \pgfsetlinewidth \arrayrulewidth
3549   \pgfsetrectcap
3550   \@@_qpoint:n { row - \l_tmpa_tl }
3551   \dim_set_eq:NN \l_tmpb_dim \pgf@y
3552   \@@_qpoint:n { col - \l_tmpb_tl }
3553   \dim_set_eq:NN \l_tmpa_dim \pgf@x
3554   \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3555   \dim_set_eq:NN \l_tmpc_dim \pgf@y
3556   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3557   \pgfpathlineto { \pgfpoint \l_tmpa_dim \l_tmpc_dim }
3558   \pgfusepathqstroke
3559   \endpgfpicture
3560 }
3561 }
3562 }

```

The group was for the color of the rules.

```

3563   \group_end:
3564   \seq_gclear:N \g_@@_pos_of_xdots_seq
3565 }

```

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.

```

3566 \cs_new_protected:Npn \@@_test_if_hline_in_block:nnnn #1 #2 #3 #4
3567 {
3568   \bool_lazy_all:nT
3569   {
3570     { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
3571     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3572     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3573     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3574   }
3575   { \bool_gset_false:N \g_tmpa_bool }

```

```
3576 }
```

The same for vertical rules.

```
3577 \cs_new_protected:Npn \@@_test_if_vline_in_block:nnnn #1 #2 #3 #4
3578 {
3579   \bool_lazy_all:nT
3580   {
3581     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3582     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3583     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
3584     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3585   }
3586   { \bool_gset_false:N \g_tmpa_bool }
3587 }
```

## The key hvlines-except-corners

```
3588 \cs_new_protected:Npn \@@_draw_hvlines_except_corners:
3589 {
```

The sequence `\l_@@_empty_corner_cells_seq` will be the sequence of all the cells empty (and not in a block) considered in the corners of the array.

```
3590   \seq_clear_new:N \l_@@_empty_corner_cells_seq
3591   \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol
3592   \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1
3593   \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol
3594   \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1
3595   \@@_draw_hvlines_ii:
3596 }
```

“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 step 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.

```
3597 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
3598 {
```

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.

```
3599   \bool_set_false:N \l_tmpa_bool
3600   \int_zero_new:N \l_@@_last_empty_row_int
3601   \int_step_inline:nnnn { #1 } { #3 } { #5 }
3602   {
3603     \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
3604     \bool_if:nTF
3605     {
3606       \cs_if_exist_p:c
3607       { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
3608       ||
3609       \l_tmpb_bool
3610     }
3611     { \bool_set_true:N \l_tmpa_bool }
3612     {
3613       \bool_if:NF \l_tmpa_bool
```

```

3614         { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
3615     }
3616 }

```

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

```

3617 \bool_set_false:N \l_tmpa_bool
3618 \int_zero_new:N \l_@@_last_empty_column_int
3619 \int_step_inline:nnnn { #2 } { #4 } { #6 }
3620 {
3621     \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
3622     \bool_if:nTF
3623     {
3624         \cs_if_exist_p:c
3625         { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
3626         || \l_tmpb_bool
3627     }
3628     { \bool_set_true:N \l_tmpa_bool }
3629     {
3630         \bool_if:NF \l_tmpa_bool
3631         { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
3632     }
3633 }

```

Now, we loop over the rows.

```

3634 \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
3635 {

```

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

```

3636     \bool_set_false:N \l_tmpa_bool
3637     \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
3638     {
3639         \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
3640         \bool_if:nTF
3641         {
3642             \cs_if_exist_p:c
3643             { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
3644             || \l_tmpb_bool
3645         }
3646         { \bool_set_true:N \l_tmpa_bool }
3647         {
3648             \bool_if:NF \l_tmpa_bool
3649             {
3650                 \int_set:Nn \l_@@_last_empty_column_int { #####1 }
3651                 \seq_put_right:Nn
3652                 \l_@@_empty_corner_cells_seq
3653                 { ##1 - #####1 }
3654             }
3655         }
3656     }
3657 }
3658 }

```

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`).

```

3659 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
3660 {
3661     \int_set:Nn \l_tmpa_int { #1 }
3662     \int_set:Nn \l_tmpb_int { #2 }
3663     \bool_set_false:N \l_tmpb_bool
3664     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3665     { \@@_test_if_cell_in_block:nnnnnn { \l_tmpa_int } { \l_tmpb_int } ##1 }
3666 }

```

```

3667 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnnn #1 #2 #3 #4 #5 #6
3668 {
3669   \bool_lazy_all:nT
3670   {
3671     { \int_compare_p:n { #3 <= #1 } }
3672     { \int_compare_p:n { #1 <= #5 } }
3673     { \int_compare_p:n { #4 <= #2 } }
3674     { \int_compare_p:n { #2 <= #6 } }
3675   }
3676   { \bool_set_true:N \l_tmpb_bool }
3677 }

```

## 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`.

```

3678 \cs_new:Npn \@@_hdottedline:
3679 {
3680   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
3681   \@@_hdottedline_i:
3682 }

```

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

```

3683 \cs_new_protected:Npn \@@_hdottedline_i:
3684 {

```

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).

```

3685   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3686   { \@@_hdottedline:n { \int_use:N \c@iRow } }
3687 }

```

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.

```

3688 \AtBeginDocument
3689 {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”.

```

3690   \cs_new_protected:Npx \@@_hdottedline:n #1
3691   {
3692     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
3693     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
3694     \c_@@_pgfortikzpicture_tl
3695     \@@_hdottedline_i:n { #1 }
3696     \c_@@_endpgfortikzpicture_tl
3697   }
3698 }

```

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

```

3699 \cs_new_protected:Npn \@@_hdottedline_i:n #1
3700 {
3701   \pgfrememberpicturepositiononpagetrue
3702   \@@_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).

```

3703 \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3704 \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
3705 \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}

```

$$\left[ \begin{array}{cccc} 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 4 \\ \hdottedline 1 & 2 & 3 & 4 \end{array} \right]$$

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}

```

$$\left[ \begin{array}{cccc} 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 4 \\ \hdottedline 1 & 2 & 3 & 4 \end{array} \right]$$

```

3706 \@@_qpoint:n { col - 1 }
3707 \dim_set:Nn \l_@@_x_initial_dim
3708 {
3709   \pgf@x +
3710   \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep
3711   - \l_@@_left_margin_dim
3712 }
3713 \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3714 \dim_set:Nn \l_@@_x_final_dim
3715 {
3716   \pgf@x -
3717   \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep
3718   + \l_@@_right_margin_dim
3719 }

```

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.

```

3720 \tl_set:Nn \l_tmpa_tl { ( }
3721 \tl_if_eq:NNF \l_@@_left_delim_tl \l_tmpa_tl
3722 { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
3723 \tl_set:Nn \l_tmpa_tl { ) }
3724 \tl_if_eq:NNF \l_@@_right_delim_tl \l_tmpa_tl
3725 { \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`.

```

3726 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
3727 \@@_draw_line:
3728 }

```

## Vertical dotted lines

```

3729 \cs_new_protected:Npn \@@_vdottedline:n #1
3730 {
3731   \bool_set_true:N \l_@@_initial_open_bool
3732   \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”.

```

3733   \bool_if:NTF \c_@@_tikz_loaded_bool
3734   {
3735     \tikzpicture
3736     \@@_vdottedline_i:n { #1 }
3737     \endtikzpicture
3738   }
3739   {
3740     \pgfpicture
3741     \@@_vdottedline_i:n { #1 }
3742     \endpgfpicture
3743   }
3744 }

3745 \cs_new_protected:Npn \@@_vdottedline_i:n #1
3746 {

```

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.

```

3747   \CT@arc@
3748   \pgfrememberpicturepositiononpagetrue
3749   \@@_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).

```

3750   \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
3751   \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
3752   \@@_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.

```

3753   \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
3754   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3755   \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`.

```

3756   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
3757   \@@_draw_line:
3758 }

```

## 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.

```

3759 \bool_new:N \l_@@_block_auto_columns_width_bool

```

As of now, there is only one option available for the environment `{NiceMatrixBlock}`.

```

3760 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
3761 {
3762   auto-columns-width .code:n =
3763   {
3764     \bool_set_true:N \l_@@_block_auto_columns_width_bool
3765     \dim_gzero_new:N \g_@@_max_cell_width_dim
3766     \bool_set_true:N \l_@@_auto_columns_width_bool
3767   }
3768 }

```



```

3769 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
3770 {
3771   \int_gincr:N \g_@@_NiceMatrixBlock_int
3772   \dim_zero:N \l_@@_columns_width_dim
3773   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
3774   \bool_if:NT \l_@@_block_auto_columns_width_bool
3775   {
3776     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
3777     {
3778       \exp_args:Nnc \dim_set:Nn \l_@@_columns_width_dim
3779       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
3780     }
3781   }
3782 }

```

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).

```

3783 {
3784   \bool_if:NT \l_@@_block_auto_columns_width_bool
3785   {
3786     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
3787     \iow_shipout:Nx \@mainaux
3788     {
3789       \cs_gset:cpn
3790       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }

```

For technical reasons, we have to include the width of a potential rule on the right side of the cells.

```

3791       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
3792     }
3793     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
3794   }
3795 }

```

## The extra nodes

First, two variants of the functions \dim\_min:nn and \dim\_max:nn.

```

3796 \cs_generate_variant:Nn \dim_min:nn { v n }
3797 \cs_generate_variant:Nn \dim_max:nn { v n }

```

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  $\backslash c\_max\_dim$  or  $-\backslash c\_max\_dim$ .

```

3798 \cs_new_protected:Npn \@@_computations_for_medium_nodes:

```

```

3799 {
3800   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3801   {
3802     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
3803     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
3804     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
3805     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
3806   }
3807   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3808   {
3809     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
3810     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
3811     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
3812     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
3813   }

```

We begin the two nested loops over the rows and the columns of the array.

```

3814   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3815   {
3816     \int_step_variable:nnNn
3817     \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.

```

3818     {
3819       \cs_if_exist:cT
3820       { 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.

```

3821     {
3822       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
3823       \dim_set:cn { l_@@_row_\@@_i: _min_dim }
3824       { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
3825       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
3826       {
3827         \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
3828         { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
3829       }

```

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.

```

3830       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
3831       \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
3832       { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
3833       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
3834       {
3835         \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
3836         { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
3837       }
3838     }
3839   }
3840 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

3841   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3842   {
3843     \dim_compare:nnNt
3844     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
3845     {
3846       \@@_qpoint:n { row - \@@_i: - base }
3847       \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
3848       \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
3849     }

```

```

3850     }
3851     \int_step_variable:nNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3852     {
3853         \dim_compare:nNnT
3854         { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
3855         {
3856             \@@_qpoint:n { col - \@@_j: }
3857             \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
3858             \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
3859         }
3860     }
3861 }

```

Here is the command `\@@_create_medium_nodes:`. When this command is used, the “medium nodes” are created.

```

3862 \cs_new_protected:Npn \@@_create_medium_nodes:
3863 {
3864     \pgfpicture
3865     \pgfrememberpicturepositiononpagetrue
3866     \pgf@relevantforpicturesizefalse
3867     \@@_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”.

```

3868     \tl_set:Nn \l_@@_suffix_tl { -medium }
3869     \@@_create_nodes:
3870     \endpgfpicture
3871 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones<sup>43</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:`.

```

3872 \cs_new_protected:Npn \@@_create_large_nodes:
3873 {
3874     \pgfpicture
3875     \pgfrememberpicturepositiononpagetrue
3876     \pgf@relevantforpicturesizefalse
3877     \@@_computations_for_medium_nodes:
3878     \@@_computations_for_large_nodes:
3879     \tl_set:Nn \l_@@_suffix_tl { - large }
3880     \@@_create_nodes:
3881     \endpgfpicture
3882 }

3883 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
3884 {
3885     \pgfpicture
3886     \pgfrememberpicturepositiononpagetrue
3887     \pgf@relevantforpicturesizefalse
3888     \@@_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”.

```

3889     \tl_set:Nn \l_@@_suffix_tl { - medium }
3890     \@@_create_nodes:
3891     \@@_computations_for_large_nodes:
3892     \tl_set:Nn \l_@@_suffix_tl { - large }
3893     \@@_create_nodes:
3894     \endpgfpicture
3895 }

```

---

<sup>43</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.

```

3896 \cs_new_protected:Npn \@@_computations_for_large_nodes:
3897 {
3898   \int_set:Nn \l_@@_first_row_int 1
3899   \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`.

```

3900   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
3901   {
3902     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
3903     {
3904       (
3905         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
3906         \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
3907       )
3908       / 2
3909     }
3910     \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
3911     { l_@@_row _ \@@_i: _ min _ dim }
3912   }
3913   \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
3914   {
3915     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
3916     {
3917       (
3918         \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
3919         \dim_use:c
3920         { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
3921       )
3922       / 2
3923     }
3924     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
3925     { l_@@_column _ \@@_j: _ max _ dim }
3926   }

```

Here, we have to use `\dim_sub:cn` because of the number 1 in the name.

```

3927   \dim_sub:cn
3928   { l_@@_column _ 1 _ min _ dim }
3929   \l_@@_left_margin_dim
3930   \dim_add:cn
3931   { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
3932   \l_@@_right_margin_dim
3933 }

```

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).

```

3934 \cs_new_protected:Npn \@@_create_nodes:
3935 {
3936   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3937   {
3938     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3939     {

```

We draw the rectangular node for the cell (`\@@_i-\@@_j`).

```

3940       \@@_pgf_rect_node:nnnnn
3941       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3942       { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }

```

```

3943         { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
3944         { \dim_use:c { l_@@_column_ \@@_j: _max_dim } }
3945         { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
3946     \str_if_empty:NF \l_@@_name_str
3947     {
3948         \pgfnodealias
3949         { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
3950         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3951     }
3952 }
3953 }

```

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  $n$ .

```

3954 \seq_mapthread_function:NNN
3955     \g_@@_multicolumn_cells_seq
3956     \g_@@_multicolumn_sizes_seq
3957     \@@_node_for_multicolumn:nn
3958 }

3959 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
3960 {
3961     \cs_set:Npn \@@_i: { #1 }
3962     \cs_set:Npn \@@_j: { #2 }
3963 }

```

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”).

```

3964 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
3965 {
3966     \@@_extract_coords_values: #1 \q_stop
3967     \@@_pgf_rect_node:nnnnn
3968     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3969     { \dim_use:c { l_@@_column_ \@@_j: _min _ dim } }
3970     { \dim_use:c { l_@@_row_ \@@_i: _min _ dim } }
3971     { \dim_use:c { l_@@_column_ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
3972     { \dim_use:c { l_@@_row_ \@@_i: _max _ dim } }
3973     \str_if_empty:NF \l_@@_name_str
3974     {
3975         \pgfnodealias
3976         { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
3977         { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
3978     }
3979 }

```

## The blocks

The code deals with the command `\Block`. This command has no direct link with the environment `{NiceMatrixBlock}`.

The following command will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewDocumentCommand` of `xparse` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label)

It’s mandatory to use an expandable command (probably because of the first optional argument ?).

```

3980 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
3981 { \@@_Block_i #2 \q_stop { #1 } { #3 } { #4 } }

```

The first mandatory argument of `\@@_Block:` has a special syntax. It must be of the form  $i$ - $j$  where  $i$  and  $j$  are the size (in rows and columns) of the block.

```

3982 \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 #5 is the label of the block.

```

3983 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
3984 {
3985   \bool_if:NT \l_@@_NiceTabular_bool
3986     { \tl_if_empty:nF { #4 } { \@@_error:n { angle-option~in~NiceTabular } } }

3987   \tl_set:Nx \l_tmpa_tl
3988     {
3989     { \int_use:N \c@iRow }
3990     { \int_use:N \c@jCol }
3991     { \int_eval:n { \c@iRow + #1 - 1 } }
3992     { \int_eval:n { \c@jCol + #2 - 1 } }
3993   }

```

Now, `\l_tmpa_tl` contains an “object” corresponding to the position of the block with four components surrounded by curly brackets:

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

We store this information in the sequence `\g_@@_pos_of_blocks_seq`.

```

3994   \seq_gput_left:NV \g_@@_pos_of_blocks_seq \l_tmpa_tl

```

We also store a complete description of the block in the sequence `\g_@@_blocks_seq`. Of course, the sequences `\g_@@_pos_of_blocks_seq` and `\g_@@_blocks_seq` are redundant, but it’s for efficiency. In `\g_@@_blocks_seq`, each block is represented by an “object” with six components:

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

```

3995   \seq_gput_left:Nx \g_@@_blocks_seq
3996   {
3997     \l_tmpa_tl
3998     { #3 }
3999     \exp_not:n { { #4 \@@_math_toggle_token: #5 \@@_math_toggle_token: } }
4000   }
4001 }

```

The key `tikz` is for Tikz options used when the PGF node of the block is created (the “normal” block node and not the “short” one nor the “medium” one). **In fact, as of now, it is *not* documented.** Is it really a good idea to provide such a key?

```

4002 \keys_define:nn { NiceMatrix / Block }
4003 {
4004   tikz .tl_set:N = \l_@@_tikz_tl ,
4005   tikz .value_required:n = true ,
4006   color .tl_set:N = \l_@@_color_tl ,
4007   color .value_required:n = true ,
4008 }

```

The command `\@@_draw_blocks:` will draw all the blocks. This command is used after the construction of the array.

```

4009 \cs_new_protected:Npn \@@_draw_blocks:
4010 { \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iii:nnnnnn ##1 } }

4011 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
4012 {

```

The group is for the keys.

```

4013   \group_begin:
4014   \keys_set:nn { NiceMatrix / Block } { #5 }

4015   \tl_if_empty:NF \l_@@_color_tl
4016   {
4017     \tl_gput_right:Nx \g_@@_code_before_tl
4018     {
4019       \exp_not:N \rectanglecolor
4020       { \l_@@_color_tl }

```

```

4021         { #1 - #2 }
4022         { #3 - #4 }
4023     }
4024 }

4025 \cs_set_protected:Npn \diagbox ##1 ##2
4026 {
4027     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4028     {
4029         \@@_actually_diagbox:nnnnnn
4030         { #1 } { #2 } { #3 } { #4 }
4031         { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4032     }
4033 }

4034 \bool_lazy_or:nnTF
4035 { \int_compare_p:nNn { #3 } > \g_@@_row_total_int }
4036 { \int_compare_p:nNn { #4 } > \c@jCol }
4037 { \msg_error:nnnn { nicematrix } { Block-too-large } { #1 } { #2 } }
4038 {

```

We put the contents of the cell in the box `\l_@@_cell_box` because we want the command `\rotate` used in the content to be able to rotate the box.

```

4039     \hbox_set:Nn \l_@@_cell_box { #6 }

```

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
three	four	two
six	seven	five
		eight

We highlight the node `1-1-block-short`

our block		one
three	four	two
six	seven	five
		eight

The construction of the node corresponding to the merged cells.

```

4040 \pgfpicture
4041 \pgfrememberpicturepositiononpagetrue
4042 \pgf@relevantforpicturesizefalse
4043 \@@_qpoint:n { row - #1 }
4044 \dim_set_eq:NN \l_tmpa_dim \pgf@y
4045 \@@_qpoint:n { col - #2 }
4046 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4047 \@@_qpoint:n { row - \@@_succ:n { #3 } }
4048 \dim_set_eq:NN \l_tmpc_dim \pgf@y
4049 \@@_qpoint:n { col - \@@_succ:n { #4 } }
4050 \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.

```

4051 \begin { pgfscope }
4052 \exp_args:Nx \pgfset { \l_@@_tikz_tl }
4053 \@@pgf_rect_node:nnnnn
4054 { \@@_env: - #1 - #2 - block }
4055 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpe_dim
4056 \end { pgfscope }

```

We construct the short node.

```

4057 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
4058 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4059 {

```

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.

```

4060 \cs_if_exist:cT
4061 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
4062 {
4063 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
4064 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
4065 }
4066 }

```

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.

```

4067 \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
4068 {
4069 \@@_qpoint:n { col - #2 }
4070 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4071 }
4072 \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
4073 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4074 {
4075 \cs_if_exist:cT
4076 { pgf @ sh @ ns @ \@@_env: - ##1 - #4 }
4077 {
4078 \pgfpointanchor { \@@_env: - ##1 - #4 } { east }
4079 \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
4080 }
4081 }
4082 \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
4083 {
4084 \@@_qpoint:n { col - \@@_succ:n { #4 } }
4085 \dim_set_eq:NN \l_tmpd_dim \pgf@x
4086 }
4087 \@@pgf_rect_node:nnnnn
4088 { \@@_env: - #1 - #2 - block - short }
4089 \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.

```

4090 \bool_if:NT \l_@@_medium_nodes_bool
4091 {
4092 \@@pgf_rect_node:nnn
4093 { \@@_env: - #1 - #2 - block - medium }
4094 { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
4095 { \pgfpointanchor { \@@_env: - #3 - #4 - medium } { south-east } }
4096 }

```

Now, we will put the label of the block.

```

4097 \int_compare:nNnTF { #1 } = { #3 }
4098 {

```



We take into account the case of a block of one row in the “first row” or the “end row”.

```

4099         \int_compare:nNnTF { #1 } = 0
4100         { \l_@@_code_for_first_row_tl }
4101         {
4102             \int_compare:nNnT { #1 } = \l_@@_last_row_int
4103             \l_@@_code_for_last_row_tl
4104         }

```

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`.

```

4105         \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

We retrieve (in `\pgf@x`) the  $x$ -value of the center of the block.

```

4106         \@@_qpoint:n { #1 - #2 - block - short }

```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```

4107         \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
4108         \pgfnode { rectangle } { base }
4109         { \box_use_drop:N \l_@@_cell_box } { } { }
4110     }

```

If the number of rows is different of 1, we put the label of the block in the center of the (short) node (the label of the block has been composed in `\l_@@_cell_box`).

```

4111     {
4112         \pgftransformshift { \@@_qpoint:n { #1 - #2 - block - short } }
4113         \pgfnode { rectangle } { center }
4114         { \box_use_drop:N \l_@@_cell_box } { } { }
4115     }
4116     \endpgfpicture
4117 }
4118 \group_end:
4119 }

```

## How to draw the dotted lines transparently

```

4120 \cs_set_protected:Npn \@@_renew_matrix:
4121 {
4122     \RenewDocumentEnvironment { pmatrix } { } {
4123         { \pNiceMatrix }
4124         { \endpNiceMatrix }
4125     }
4126     \RenewDocumentEnvironment { vmatrix } { } {
4127         { \vNiceMatrix }
4128         { \endvNiceMatrix }
4129     }
4130     \RenewDocumentEnvironment { Vmatrix } { } {
4131         { \VNiceMatrix }
4132         { \endVNiceMatrix }
4133     }
4134     \RenewDocumentEnvironment { bmatrix } { } {
4135         { \bNiceMatrix }
4136         { \endbNiceMatrix }
4137     }
4138     \RenewDocumentEnvironment { Bmatrix } { } {
4139         { \BNiceMatrix }
4140         { \endBNiceMatrix }
4141     }

```

## Automatic arrays

```

4138 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
4139 {
4140     \int_set:Nn \l_@@_nb_rows_int { #1 }
4141     \int_set:Nn \l_@@_nb_cols_int { #2 }

```

```

4142 }
4143 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
4144 {
4145   \int_zero_new:N \l_@@_nb_rows_int
4146   \int_zero_new:N \l_@@_nb_cols_int
4147   \@@_set_size:n #4 \q_stop
4148   \begin { NiceArrayWithDelims } { #1 } { #2 }
4149     { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
4150   \int_compare:nNnT \l_@@_first_row_int = 0
4151     {
4152       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4153       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4154       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4155     }
4156   \prg_replicate:nn \l_@@_nb_rows_int
4157     {
4158       \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).

```

4159       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
4160       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4161     }
4162   \int_compare:nNnT \l_@@_last_row_int > { -2 }
4163     {
4164       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4165       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4166       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4167     }
4168   \end { NiceArrayWithDelims }
4169 }
4170 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
4171 {
4172   \cs_set_protected:cpn { #1 AutoNiceMatrix }
4173   {
4174     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
4175     \AutoNiceMatrixWithDelims { #2 } { #3 }
4176   }
4177 }
4178 \@@_define_com:nnn p ( )
4179 \@@_define_com:nnn b [ ]
4180 \@@_define_com:nnn v | |
4181 \@@_define_com:nnn V \ | \ |
4182 \@@_define_com:nnn B \{ \}

```

We define also an command \AutoNiceMatrix similar to the environment {NiceMatrix}.

```

4183 \NewDocumentCommand \AutoNiceMatrix { O { } m O { } m ! O { } }
4184 {
4185   \group_begin:
4186     \bool_set_true:N \l_@@_NiceArray_bool
4187     \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
4188   \group_end:
4189 }

```

## The redefinition of the command \dotfill

```

4190 \cs_set_eq:NN \@@_dotfill \dotfill
4191 \cs_new_protected:Npn \@@_dotfill:
4192 {

```

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}).

```

4193 \@@_dotfill
4194 \bool_if:NT \l_@@_NiceTabular_bool
4195 { \group_insert_after:N \@@_dotfill_ii: }
4196 { \group_insert_after:N \@@_dotfill_i: }
4197 }
4198 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
4199 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }
Now, if the box is not empty (unfortunately, we can't actually test whether the box is empty and that's
why we only consider its width), we insert \@@_dotfill (which is the saved version of \dotfill) in
the cell of the array, and it will extend, since it is no longer in \l_@@_cell_box.
4200 \cs_new_protected:Npn \@@_dotfill_iii:
4201 { \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } = \c_zero_dim \@@_dotfill }

```

## The command \diagbox

The command \diagbox will be linked to \diagbox:nn in the environments of nicematrix.

```

4202 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
4203 {
4204   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4205   {
4206     \@@_actually_diagbox:nnnnnn
4207     { \int_use:N \c@iRow }
4208     { \int_use:N \c@jCol }
4209     { \int_use:N \c@iRow }
4210     { \int_use:N \c@jCol }
4211     { \exp_not:n { #1 } }
4212     { \exp_not:n { #2 } }
4213   }

```

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 hvlines-except-corners.

```

4214   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
4215   {
4216     { \int_use:N \c@iRow }
4217     { \int_use:N \c@jCol }
4218     { \int_use:N \c@iRow }
4219     { \int_use:N \c@jCol }
4220   }
4221 }

```

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.

```

4222 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
4223 {
4224   \pgfpicture
4225   \pgf@relevantforpicturesizefalse
4226   \pgfrememberpicturepositiononpagetrue
4227   \@@_qpoint:n { row - #1 }
4228   \dim_set_eq:NN \l_tmpa_dim \pgf@y
4229   \@@_qpoint:n { col - #2 }
4230   \dim_set_eq:NN \l_tmpb_dim \pgf@x
4231   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4232   \@@_qpoint:n { row - \@@_succ:n { #3 } }
4233   \dim_set_eq:NN \l_tmpc_dim \pgf@y
4234   \@@_qpoint:n { col - \@@_succ:n { #4 } }
4235   \dim_set_eq:NN \l_tmpd_dim \pgf@x
4236   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
4237   {

```

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.

```

4238     \CT@arc@
4239     \pgfsetroundcap
4240     \pgfusepathqstroke
4241 }
4242 \pgfset { inner~sep = 1 pt }
4243 \pgfscope
4244 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4245 \pgfnode { rectangle } { south~west }
4246 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
4247 \endpgfscope
4248 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
4249 \pgfnode { rectangle } { north~east }
4250 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
4251 \endpgfpicture
4252 }
```

## 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. 82.

The command `\CodeAfter` catches everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```

4253 \cs_new_protected:Npn \@@_CodeAfter:n #1 \end
4254 {
4255   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
4256   \@@_CodeAfter_i:n
4257 }
```

We catch the argument of the command `\end` (in `#1`).

```

4258 \cs_new_protected:Npn \@@_CodeAfter_i:n #1
4259 {
```

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.

```

4260   \str_if_eq:eeTF \@currenvir { #1 }
4261   { \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`.

```

4262   {
4263     \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
4264     \@@_CodeAfter:n
4265   }
4266 }
```

## 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.

```

4267 \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.

```

4268 \bool_new:N \c_@@_footnote_bool

4269 \@@_msg_new:nnn { Unknown~option~for~package }
4270 {
4271   The~option~'\l_keys_key_tl'~is~unknown. \\
4272   If~you~go~on,~it~will~be~ignored. \\
4273   For~a~list~of~the~available~options,~type~H~<return>.
4274 }
4275 {
4276   The~available~options~are~(in~alphabetic~order):~
4277   define-L-C-R,~
4278   footnote,~
4279   footnotehyper,~
4280   renew-dots,~
4281   renew-matrix~and~
4282   transparent.
4283 }

4284 \keys_define:nn { NiceMatrix / Package }
4285 {
4286   define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
4287   define-L-C-R .default:n = true ,
4288   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
4289   renew-dots .value_forbidden:n = true ,
4290   renew-matrix .code:n = \@@_renew_matrix: ,
4291   renew-matrix .value_forbidden:n = true ,
4292   transparent .meta:n = { renew-dots , renew-matrix } ,
4293   transparent .value_forbidden:n = true,
4294   footnote .bool_set:N = \c_@@_footnote_bool ,
4295   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
4296   unknown .code:n = \@@_error:n { Unknown~option~for~package }
4297 }

4298 \ProcessKeysOptions { NiceMatrix / Package }

4299 \@@_msg_new:nn { footnote~with~footnotehyper~package }
4300 {
4301   You~can't~use~the~option~'footnote'~because~the~package~
4302   footnotehyper~has~already~been~loaded.~
4303   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
4304   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
4305   of~the~package~footnotehyper.\\
4306   If~you~go~on,~the~package~footnote~won't~be~loaded.
4307 }

4308 \@@_msg_new:nn { footnotehyper~with~footnote~package }
4309 {
4310   You~can't~use~the~option~'footnotehyper'~because~the~package~
4311   footnote~has~already~been~loaded.~
4312   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
4313   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
4314   of~the~package~footnote.\\
4315   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
4316 }

4317 \bool_if:NT \c_@@_footnote_bool
4318 {
4319   \@ifclassloaded { beamer }
4320   { \msg_info:nn { nicematrix } { Option~incompatible~with~Beamer } }
4321   {
4322     \@ifpackageloaded { footnotehyper }
4323     { \@@_error:n { footnote~with~footnotehyper~package } }
4324     { \usepackage { footnote } }

```

```

4325     }
4326 }
4327 \bool_if:NT \c_@@_footnotehyper_bool
4328 {
4329   \@ifclassloaded { beamer }
4330   { \@@_info:n { Option-incompatible-with-Beamer } }
4331   {
4332     \@ifpackageloaded { footnote }
4333     { \@@_error:n { footnotehyper~with~footnote~package } }
4334     { \usepackage { footnotehyper } }
4335   }
4336   \bool_set_true:N \c_@@_footnote_bool
4337 }

```

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

The following command converts all the elements of a sequence (which are token lists) into strings.

```

4338 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
4339 {
4340   \seq_clear:N \l_tmpa_seq
4341   \seq_map_inline:Nn #1
4342   {
4343     \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
4344   }
4345   \seq_set_eq:NN #1 \l_tmpa_seq
4346 }

```

The following command creates a sequence of strings (`str`) from a `clist`.

```

4347 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
4348 {
4349   \seq_set_from_clist:Nn #1 { #2 }
4350   \@@_convert_to_str_seq:N #1
4351 }
4352 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
4353 {
4354   NiceMatrix ,
4355   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
4356 }

```

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`.

```

4357 \cs_new_protected:Npn \@@_error_too_much_cols:
4358 {
4359   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
4360   {
4361     \int_compare:nNnTF \l_@@_last_col_int = { -2 }
4362     { \@@_fatal:n { too-much-cols-for-matrix } }
4363     {
4364       \bool_if:NF \l_@@_last_col_without_value_bool
4365       { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
4366     }
4367   }
4368   { \@@_fatal:n { too-much-cols-for-array } }
4369 }

```

The following command must *not* be protected since it's used in an error message.

```

4370 \cs_new:Npn \@@_message_hdotsfor:
4371 {
4372   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
4373   { ~Maybe~your~use~of~\token_to_str:N \Hdotsfor\ is~incorrect.}
4374 }
4375 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
4376 {
4377   You~try~to~use~more~columns~than~allowed~by~your~
4378   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~
4379   columns~is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus~the~
4380   exterior~columns).~This~error~is~fatal.
4381 }
4382 \@@_msg_new:nn { too-much-cols-for-matrix }
4383 {
4384   You~try~to~use~more~columns~than~allowed~by~your~
4385   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall~that~the~maximal~
4386   number~of~columns~for~a~matrix~is~fixed~by~the~LaTeX~counter~
4387   'MaxMatrixCols'.~Its~actual~value~is~\int_use:N \c@MaxMatrixCols.~
4388   This~error~is~fatal.
4389 }

```

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`.

```

4390 \@@_msg_new:nn { too-much-cols-for-array }
4391 {
4392   You~try~to~use~more~columns~than~allowed~by~your~
4393   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~columns~is~
4394   \int_use:N \g_@@_static_num_of_col_int\
4395   ~~~~~(plus~the~potential~exterior~ones).~
4396   This~error~is~fatal.
4397 }
4398 \@@_msg_new:nn { last-col-not-used }
4399 {
4400   The~key~'last-col'~is~in~force~but~you~have~not~used~that~last~column~
4401   in~your~\@@_full_name_env:~.~However,~you~can~go~on.
4402 }
4403 \@@_msg_new:nn { columns-not-used }
4404 {
4405   The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
4406   \g_@@_static_num_of_col_int\
4407   columns~but~you~use~only~\int_use:N \c@jCol.\\
4408   However,~you~can~go~on.
4409 }
4410 \@@_msg_new:nn { in-first-col }
4411 {
4412   You~can't~use~the~command~#1 in~the~first~column~(number~0)~of~the~array.\\
4413   If~you~go~on,~this~command~will~be~ignored.
4414 }
4415 \@@_msg_new:nn { in-last-col }
4416 {
4417   You~can't~use~the~command~#1 in~the~last~column~(exterior)~of~the~array.\\
4418   If~you~go~on,~this~command~will~be~ignored.
4419 }
4420 \@@_msg_new:nn { in-first-row }
4421 {
4422   You~can't~use~the~command~#1 in~the~first~row~(number~0)~of~the~array.\\
4423   If~you~go~on,~this~command~will~be~ignored.
4424 }

```

```

4425 \@@_msg_new:nn { in-last-row }
4426 {
4427     You~can't~use~the~command~\#1~in~the~last~row~(exterior)~of~the~array.\
4428     If~you~go~on,~this~command~will~be~ignored.
4429 }
4430 \@@_msg_new:nn { option-S~without~siunitx }
4431 {
4432     You~can't~use~the~option~'S'~in~your~environment~\@@_full_name_env:
4433     because~you~have~not~loaded~siunitx.\
4434     If~you~go~on,~this~option~will~be~ignored.
4435 }
4436 \@@_msg_new:nn { bad-option-for~line-style }
4437 {
4438     Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line-style'~
4439     is~'standard'.~If~you~go~on,~this~option~will~be~ignored.
4440 }
4441 \@@_msg_new:nn { Unknown~option~for~xdots }
4442 {
4443     As~for~now~there~is~only~three~options~available~here:~'color',~'line-style'~
4444     and~'shorten'~(and~you~try~to~use~'\l_keys_key_tl').~If~you~go~on,~
4445     this~option~will~be~ignored.
4446 }
4447 \@@_msg_new:nn { ampersand~in~light-syntax }
4448 {
4449     You~can't~use~an~ampersand~(\token_to_str &)~to~separate~columns~because
4450     ~you~have~used~the~option~'light-syntax'.~This~error~is~fatal.
4451 }
4452 \@@_msg_new:nn { double-backslash~in~light-syntax }
4453 {
4454     You~can't~use~\token_to_str:N \~to~separate~rows~because~you~have~used~
4455     the~option~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
4456     (set~by~the~option~'end-of-row').~This~error~is~fatal.
4457 }
4458 \@@_msg_new:nn { standard-cline~in~document }
4459 {
4460     The~key~'standard-cline'~is~available~only~in~the~preamble.\
4461     If~you~go~on~this~command~will~be~ignored.
4462 }
4463 \@@_msg_new:nn { bad~value~for~baseline }
4464 {
4465     The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
4466     valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
4467     \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\
4468     If~you~go~on,~a~value~of~1~will~be~used.
4469 }
4470 \@@_msg_new:nn { empty~environment }
4471 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }
4472 \@@_msg_new:nn { unknown~cell~for~line~in~code~after }
4473 {
4474     Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
4475     can't~be~executed~because~a~cell~doesn't~exist.\
4476     If~you~go~on~this~command~will~be~ignored.
4477 }
4478 \@@_msg_new:nn { last-col~non-empty~for~NiceArray }
4479 {
4480     In~the~\@@_full_name_env:,~you~must~use~the~option~
4481     'last-col'~without~value.\
4482     However,~you~can~go~on~for~this~time~
4483     (the~value~'\l_keys_value_tl'~will~be~ignored).
4484 }

```



```

4485 \@@_msg_new:nn { last-col~non~empty~for~NiceMatrixOptions }
4486 {
4487   In~\NiceMatrixoptions,~you~must~use~the~option~
4488   'last-col'~without~value.\\
4489   However,~you~can~go~on~for~this~time~
4490   (the~value~'\l_keys_value_tl'~will~be~ignored).
4491 }
4492 \@@_msg_new:nn { Block~too~large }
4493 {
4494   You~try~to~draw~a~block~in~the~cell~#1-#2~of~your~matrix~but~the~matrix~is~
4495   too~small~for~that~block. \\
4496 }
4497 \@@_msg_new:nn { unknown~column~type }
4498 {
4499   The~column~type~'#1'~in~your~\@@_full_name_env:\
4500   is~unknown. \\
4501   This~error~is~fatal.
4502 }
4503 \@@_msg_new:nn { angle~option~in~NiceTabular }
4504 {
4505   You~should~not~the~option~between~angle~brackets~(<~and~>)~for~a~command~
4506   \token_to_str:N \Block\ in~\{NiceTabular\}.~However,~you~can~go~on.
4507 }
4508 \@@_msg_new:nn { tabularnote~forbidden }
4509 {
4510   You~can't~use~the~command~\token_to_str:N\tabularnote\
4511   ~in~a~\@@_full_name_env:~This~command~is~available~only~in~
4512   \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
4513   If~you~go~on,~this~command~will~be~ignored.
4514 }
4515 \@@_msg_new:nn { bottomule~without~booktabs }
4516 {
4517   You~can't~use~the~option~'tabular/bottomrule'~because~you~haven't~
4518   loaded~'booktabs'.\\
4519   If~you~go~on,~this~option~will~be~ignored.
4520 }
4521 \@@_msg_new:nn { enumitem~not~loaded }
4522 {
4523   You~can't~use~the~command~\token_to_str:N\tabularnote\
4524   ~because~you~haven't~loaded~'enumitem'.\\
4525   If~you~go~on,~this~command~will~be~ignored.
4526 }
4527 \@@_msg_new:nn { Wrong~last~row }
4528 {
4529   You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
4530   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
4531   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
4532   last~row.~You~can~avoid~this~problem~by~using~'last-row'~
4533   without~value~(more~compilations~might~be~necessary).
4534 }
4535 \@@_msg_new:nn { Yet~in~env }
4536 { Environments~of~nicematrix~can't~be~nested.\\ This~error~is~fatal. }
4537 \@@_msg_new:nn { Outside~math~mode }
4538 {
4539   The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
4540   (and~not~in~\token_to_str:N \vcenter).\\
4541   This~error~is~fatal.
4542 }
4543 \@@_msg_new:nn { Bad~value~for~letter~for~dotted~lines }

```

```

4544 {
4545     The~value~of~key~'\l_keys_key_tl'~must~be~of~length~1.\\
4546     If~you~go~on,~it~will~be~ignored.
4547 }

4548 \@@_msg_new:nnn { Unknown~key~for~notes }
4549 {
4550     The~key~'\l_keys_key_tl'~is~unknown.\\
4551     If~you~go~on,~it~will~be~ignored. \\
4552     For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
4553 }
4554 {
4555     The~available~options~are~(in~alphabetic~order):~
4556     bottomrule,~
4557     code~after,~
4558     code~before,~
4559     enumitem~keys,~
4560     enumitem~keys~para,~
4561     para,~
4562     label~in~list,~
4563     label~in~tabular~and~
4564     style.
4565 }

4566 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
4567 {
4568     The~key~'\l_keys_key_tl'~is~unknown~for~the~command~
4569     \token_to_str:N \NiceMatrixOptions. \\
4570     If~you~go~on,~it~will~be~ignored. \\
4571     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
4572 }
4573 {
4574     The~available~options~are~(in~alphabetic~order):~
4575     allow~duplicate~names,~
4576     cell~space~bottom~limit,~
4577     cell~space~top~limit,~
4578     code~for~first~col,~
4579     code~for~first~row,~
4580     code~for~last~col,~
4581     code~for~last~row,~
4582     create~extra~nodes,~
4583     create~medium~nodes,~
4584     create~large~nodes,~
4585     end~of~row,~
4586     first~col,~
4587     first~row,~
4588     hlines,~
4589     hvlines,~
4590     hvlines~except~corners,~
4591     last~col,~
4592     last~row,~
4593     left~margin,~
4594     letter~for~dotted~lines,~
4595     light~syntax,~
4596     notes~(several subkeys),~
4597     nullify~dots,~
4598     renew~dots,~
4599     renew~matrix,~
4600     right~margin,~
4601     small,~
4602     transparent,~
4603     vlines,~
4604     xdots/color,~
4605     xdots/shorten~and~
4606     xdots/line~style.

```

```

4607     }
4608 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
4609 {
4610     The~option~'\l_keys_key_tl'~is~unknown~for~the~environment~
4611     \{NiceArray\}. \\
4612     If~you~go~on,~it~will~be~ignored. \\
4613     For~a~list~of~the~*principal*~available~options,~type~H~<return>.
4614 }
4615 {
4616     The~available~options~are~(in~alphabetic~order):~
4617     b,~
4618     baseline,~
4619     c,~
4620     cell-space-bottom-limit,~
4621     cell-space-top-limit,~
4622     code-after,~
4623     code-for-first-col,~
4624     code-for-first-row,~
4625     code-for-last-col,~
4626     code-for-last-row,~
4627     colortbl-like,~
4628     columns-width,~
4629     create-extra-nodes,~
4630     create-medium-nodes,~
4631     create-large-nodes,~
4632     extra-left-margin,~
4633     extra-right-margin,~
4634     first-col,~
4635     first-row,~
4636     hlines,~
4637     hvlines,~
4638     last-col,~
4639     last-row,~
4640     left-margin,~
4641     light-syntax,~
4642     name,~
4643     notes/bottomrule,~
4644     notes/para,~
4645     nullify-dots,~
4646     renew-dots,~
4647     right-margin,~
4648     rules/color,~
4649     rules/width,~
4650     small,~
4651     t,~
4652     vlines,~
4653     xdots/color,~
4654     xdots/shorten~and~
4655     xdots/line-style.
4656 }

```

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 options `t`, `c` and `b`).

```

4657 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
4658 {
4659     The~option~'\l_keys_key_tl'~is~unknown~for~the~
4660     \@@_full_name_env:. \\
4661     If~you~go~on,~it~will~be~ignored. \\
4662     For~a~list~of~the~*principal*~available~options,~type~H~<return>.
4663 }
4664 {
4665     The~available~options~are~(in~alphabetic~order):~
4666     b,~

```

```

4667 baseline,~
4668 c,~
4669 cell-space-bottom-limit,~
4670 cell-space-top-limit,~
4671 code-after,~
4672 code-for-first-col,~
4673 code-for-first-row,~
4674 code-for-last-col,~
4675 code-for-last-row,~
4676 colortbl-like,~
4677 columns-width,~
4678 create-extra-nodes,~
4679 create-medium-nodes,~
4680 create-large-nodes,~
4681 extra-left-margin,~
4682 extra-right-margin,~
4683 first-col,~
4684 first-row,~
4685 hlines,~
4686 hvlines,~
4687 l,~
4688 last-col,~
4689 last-row,~
4690 left-margin,~
4691 light-syntax,~
4692 name,~
4693 nullify-dots,~
4694 r,~
4695 renew-dots,~
4696 right-margin,~
4697 rules/color,~
4698 rules/width,~
4699 S,~
4700 small,~
4701 t,~
4702 vlines,~
4703 xdots/color,~
4704 xdots/shorten~and~
4705 xdots/line-style.
4706 }

4707 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
4708 {
4709   The~option~'\l_keys_key_tl'~is~unknown~for~the~environment~
4710   \{NiceTabular\}. \\
4711   If~you~go~on,~it~will~be~ignored. \\
4712   For~a~list~of~the~*principal*~available~options,~type~H~<return>.
4713 }
4714 {
4715   The~available~options~are~(in~alphabetic~order):~
4716   b,~
4717   baseline,~
4718   c,~
4719   cell-space-bottom-limit,~
4720   cell-space-top-limit,~
4721   code-after,~
4722   code-for-first-col,~
4723   code-for-first-row,~
4724   code-for-last-col,~
4725   code-for-last-row,~
4726   colortbl-like,~
4727   columns-width,~
4728   create-extra-nodes,~
4729   create-medium-nodes,~

```

```

4730     create-large-nodes,~
4731     extra-left-margin,~
4732     extra-right-margin,~
4733     first-col,~
4734     first-row,~
4735     hlines,~
4736     hvlines,~
4737     last-col,~
4738     last-row,~
4739     left-margin,~
4740     light-syntax,~
4741     name,~
4742     notes/bottomrule,~
4743     notes/para,~
4744     nullify-dots,~
4745     renew-dots,~
4746     right-margin,~
4747     rules/color,~
4748     rules/width,~
4749     t,~
4750     vlines,~
4751     xdots/color,~
4752     xdots/shorten~and~
4753     xdots/line-style.
4754 }

4755 \@@_msg_new:nnn { Duplicate-name }
4756 {
4757     The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
4758     the~same~environment~name~twice.~You~can~go~on,~but,~
4759     maybe,~you~will~have~incorrect~results~especially~
4760     if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
4761     message~again,~use~the~option~'allow-duplicate-names'.\\
4762     For~a~list~of~the~names~already~used,~type~H~<return>. \\
4763 }
4764 {
4765     The~names~already~defined~in~this~document~are:~
4766     \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
4767 }

4768 \@@_msg_new:nn { Option~auto~for~columns-width }
4769 {
4770     You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
4771     If~you~go~on,~the~option~will~be~ignored.
4772 }

```

## 18 History

### 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>44</sup>, Tikz externalization is now deactivated in the environments of the package `nicematrix`.<sup>45</sup>

## Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\begin{pmatrix} & C_j & \\ 0 & \vdots & 0 \\ & a & \cdots \\ 0 & & 0 \end{pmatrix} L_i$$

## 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>44</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>45</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>46</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>46</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 try 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, row 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.

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<b>@@ commands:</b>	
<code>\@@_Block:</code> . . . . .	986, 3980
<code>\@@_Block_i</code> . . . . .	3981, 3982
<code>\@@_Block_ii:nnnnn</code> . . . . .	3982, 3983
<code>\@@_Block_iii:nnnnnn</code> . . . . .	4010, 4011
<code>\@@_Cdots</code> . . . . .	912, 977, 2792
<code>\g_@@_Cdots_lines_tl</code> . . . . .	1002, 2119
<code>\@@_Cell:</code> <i>181, 716, 1364, 1413, 1430, 1927, 2882</i>	
<code>\@@_CodeAfter:n</code> . . . . .	990, 4253, 4264
<code>\@@_CodeAfter_i:n</code> . . . . .	4256, 4258
<code>\@@_Ddots</code> . . . . .	914, 979, 2822
<code>\g_@@_Ddots_lines_tl</code> . . . . .	1005, 2117
<code>\g_@@_HVdotsfor_lines_tl</code> . . . . .	1007, 2115, 2917, 2993, 4372
<code>\@@_Hdotsfor:</code> . . . . .	917, 983, 2906
<code>\@@_Hdotsfor:nnnn</code> . . . . .	2919, 2931
<code>\@@_Hdotsfor_i</code> . . . . .	2909, 2915
<code>\@@_Hspace:</code> . . . . .	982, 2867
<code>\@@_Iddots</code> . . . . .	915, 980, 2844
<code>\g_@@_Iddots_lines_tl</code> . . . . .	1006, 2118
<code>\@@_Ldots</code> . . . . .	911, 916, 976, 2777
<code>\g_@@_Ldots_lines_tl</code> . . . . .	1003, 2120
<code>\l_@@_Matrix_bool</code> . . . . .	222, 1204, 1278, 1918
<code>\l_@@_NiceArray_bool</code> . . . . .	219, 305, 1045, 1156, 1216, 1309, 1321, 1894, 3399, 3400, 3419, 3420, 3450, 3451, 3509, 3510, 4186
<code>\g_@@_NiceMatrixBlock_int</code> . . . . .	215, 3771, 3776, 3779, 3790
<code>\l_@@_NiceTabular_bool</code> . . . . .	135, 143, 220, 723, 848, 1021, 1100, 1102, 1248, 1252, 1310, 1322, 1945, 1954, 2380, 2388, 3064, 3710, 3717, 3985, 4194
<code>\@@_OnlyMainNiceMatrix:n</code> . . . . .	988, 3329
<code>\@@_OnlyMainNiceMatrix_i:n</code> . . . . .	3332, 3339, 3342
<code>\@@_Vdots</code> . . . . .	913, 978, 2807
<code>\g_@@_Vdots_lines_tl</code> . . . . .	1004, 2116
<code>\@@_Vdotsfor:</code> . . . . .	984, 2991
<code>\@@_Vdotsfor:nnnn</code> . . . . .	2995, 3006
<code>\@@_W:</code> . . . . .	1282, 1352
<code>\@@_actually_diagbox:nnnnnn</code> . . . . .	4029, 4206, 4222
<code>\@@_actually_draw_Cdots:</code> . . . . .	2369, 2373
<code>\@@_actually_draw_Ddots:</code> . . . . .	2496, 2500
<code>\@@_actually_draw_Iddots:</code> . . . . .	2547, 2551
<code>\@@_actually_draw_Ldots:</code> . . . . .	2327, 2331, 2982
<code>\@@_actually_draw_Vdots:</code> . . . . .	2423, 2427, 3057
<code>\@@_adapt_S_column:</code> . . . . .	151, 166, 1020
<code>\@@_after_array:</code> . . . . .	1271, 1958
<code>\@@_analyze_end:Nn</code> . . . . .	1620, 1665
<code>\l_@@_argspec_tl</code> . . . . .	2775, 2776, 2777, 2792, 2807, 2822, 2844, 2913, 2914, 2915, 2989, 2990, 2991, 3087, 3088, 3089
<code>\@@_array:</code> . . . . .	843, 1621, 1648
<code>\l_@@_auto_columns_width_bool</code> . . . . .	411, 522, 1728, 1732, 3766
<code>\l_@@_baseline_str</code> . . . . .	399, 400, 515, 516, 517, 518, 856, 1218, 1466, 1472, 1477, 1557, 1558, 1559
<code>\@@_begin_of_NiceMatrix:nn</code> . . . . .	1916, 1937

\@@_begin_of_row: .....	720, 744, 1814	\@@_def_env:nnn .....	1900, 1911, 1912, 1913, 1914, 1915
\l_@@_block_auto_columns_width_bool ..	1034, 1733, 3759, 3764, 3774, 3784	\@@_define_L_C_R: .....	203, 1168
\g_@@_blocks_seq .....	257, 1037, 2064, 3995, 4010	\c_@@_define_L_C_R_bool .....	202, 1168, 2883, 4286
\c_@@_booktabs_loaded_bool .....	24, 30, 927, 1526	\@@_define_com:nnn .....	4170, 4178, 4179, 4180, 4181, 4182
\l_@@_cell_box .....	722, 768, 770, 776, 784, 785, 786, 787, 789, 792, 794, 796, 813, 929, 1111, 1113, 1429, 1437, 1815, 1837, 1840, 1842, 1857, 1878, 1882, 3072, 3075, 3079, 4039, 4109, 4114, 4201	\g_@@_delta_x_one_dim .....	2030, 2523, 2533
\l_@@_cell_space_bottom_limit_dim ..	388, 455, 787	\g_@@_delta_x_two_dim .....	2032, 2574, 2584
\l_@@_cell_space_top_limit_dim .....	387, 453, 785	\g_@@_delta_y_one_dim .....	2031, 2525, 2533
\@@_cellcolor .....	1094, 3215, 3300, 3301	\g_@@_delta_y_two_dim .....	2033, 2576, 2584
\@@_cellcolor_tabular .....	921, 3306	\@@_diagbox:nn .....	991, 4202
\g_@@_cells_seq .....	1659, 1660, 1661, 1663	\@@_dotfill .....	4190, 4193, 4201
\@@_chessboardcolors .....	1099, 3293	\@@_dotfill: .....	989, 4191
\@@_cline .....	118, 975	\@@_dotfill_i: .....	4196, 4198
\@@_cline_i:nn .....	119, 120, 128, 131	\@@_dotfill_ii: .....	4195, 4198, 4199
\@@_cline_i:w .....	120, 121	\@@_dotfill_iii: .....	4199, 4200
\l_@@_code_before_bool .....	246, 512, 539, 863, 1041, 1051, 1678, 1692, 1710, 1741, 1767, 1790, 2004, 2096	\@@_double_int_eval:n .....	3083, 3097, 3098
\g_@@_code_before_tl .....	1008, 2084, 2093, 3308, 3313, 3324, 4017	\g_@@_dp_ante_last_row_dim .....	747, 960
\l_@@_code_before_tl ..	245, 511, 1042, 1101	\g_@@_dp_last_row_dim .....	747, 748, 963, 964, 1112, 1113, 1235
\l_@@_code_for_first_col_tl .....	467, 1826	\g_@@_dp_row_zero_dim .....	767, 768, 954, 955, 1228, 1551, 1577
\l_@@_code_for_first_row_tl ..	471, 732, 4100	\@@_draw_Cdots:nnn .....	2355
\l_@@_code_for_last_col_tl .....	469, 1866	\@@_draw_Ddots:nnn .....	2488
\l_@@_code_for_last_row_tl ..	473, 739, 4103	\@@_draw_Iddots:nnn .....	2539
\g_@@_col_total_int .....	721, 998, 1197, 1760, 1761, 1792, 1797, 1802, 1803, 1856, 1962, 1965, 1970, 1977, 2021, 2460, 3052, 3807, 3817, 3851, 3938	\@@_draw_Ldots:nnn .....	2313
\l_@@_color_tl .....	261, 4006, 4015, 4020	\@@_draw_Vdots:nnn .....	2409
\@@_colortbl_like: .....	919, 992	\@@_draw_blocks: .....	2064, 4009
\l_@@_colortbl_like_bool .....	385, 538, 992, 1297	\@@_draw_dotted_lines: .....	2054, 2104
\c_@@_colortbl_loaded_bool .....	82, 86, 944	\@@_draw_dotted_lines_i: .....	2107, 2111
\@@_columncolor .....	1098, 3179	\@@_draw_hlines: .....	2058, 3411, 3442
\@@_columncolor_preamble .....	923, 3320	\@@_draw_hvlines: .....	2056, 3432
\c_@@_columncolor_regex .....	190, 1300	\@@_draw_hvlines_except_corners: .....	2061, 3588
\l_@@_columns_width_dim .....	216, 523, 645, 1729, 1735, 3772, 3778	\@@_draw_hvlines_i: .....	3437, 3440
\g_@@_com_or_env_str .....	232, 235	\@@_draw_hvlines_ii: .....	3438, 3445, 3595
\@@_computations_for_large_nodes: ..	3878, 3891, 3896	\@@_draw_line: .....	2353, 2407, 2486, 2537, 2588, 2590, 3136, 3727, 3757
\@@_computations_for_medium_nodes: ..	3798, 3867, 3877, 3888	\@@_draw_line_ii:nn .....	3116, 3120
\@@_compute_a_corner:nnnnnn .....	3591, 3592, 3593, 3594, 3597	\@@_draw_line_iii:nn .....	3123, 3127
\@@_construct_preamble:n .....	1169, 1275	\@@_draw_non_standard_dotted_line: ..	2596, 2598
\@@_convert_to_str_seq:N .....	4338, 4350	\@@_draw_non_standard_dotted_line:n ..	2601, 2604
\@@_create_col_nodes: .....	1624, 1652, 1671	\@@_draw_standard_dotted_line: ..	2595, 2624
\@@_create_large_nodes: .....	2041, 3872	\@@_draw_standard_dotted_line_i: ..	2689, 2693
\@@_create_medium_and_large_nodes: ..	2038, 3883	\@@_draw_vlines: .....	2059, 3385, 3443
\@@_create_medium_nodes: .....	2039, 3862	\g_@@_empty_cell_bool .....	253, 791, 798, 2790, 2805, 2820, 2842, 2864, 2869
\@@_create_nodes: .....	3869, 3880, 3890, 3893, 3934	\l_@@_empty_corner_cells_seq .....	3466, 3472, 3479, 3523, 3529, 3536, 3590, 3652
\@@_create_row_node: .....	859, 891, 928	\@@_end_Cell: .....	183, 780, 1364, 1419, 1434, 1927, 2885
\@@_cut_on_hyphen:w .....	3138, 3160, 3161, 3194, 3195, 3224, 3255, 3266	\l_@@_end_of_row_tl .....	419, 420, 461, 1644, 1645, 4455
\g_@@_ddots_int .....	2028, 2520, 2521	\c_@@_endpgfortikzpicture_tl .....	39, 43, 2108, 3124, 3696
		\c_@@_enumitem_loaded_bool .....	25, 33, 282, 566, 571, 582, 587
		\@@_env: .....	210, 214, 753, 759, 814, 820, 868, 874, 880, 1065, 1066, 1072, 1073, 1080, 1081, 1091, 1679, 1682, 1684, 1697, 1703, 1706, 1715, 1721, 1724, 1746, 1752, 1755, 1772,

1778, 1784, 1792, 1797, 1803, 2075, 2185, 2253, 2295, 2306, 2945, 2963, 3020, 3038, 3109, 3111, 3130, 3133, 3607, 3625, 3643, 3820, 3822, 3830, 3941, 3950, 3968, 4054, 4061, 4063, 4076, 4078, 4088, 4093, 4094, 4095	\l_@@_hlines_bool .. 404, 475, 481, 892, 2058
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\g_@@_ht_last_row_dim ..... ..... 749, 961, 962, 1110, 1111, 1234
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\g_@@_ht_row_one_dim .... 775, 776, 958, 959
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\g_@@_ht_row_zero_dim ..... ..... 769, 770, 956, 957, 1229, 1550, 1576
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_hvlines_bool ... 406, 479, 2055, 2270
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_hvlines_except_corners_bool ... ..... 407, 505, 2060, 3461, 3518
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\@@_i: ..... 3800, 3802, 3803, 3804, 3805, 3814, 3820, 3822, 3823, 3824, 3825, 3830, 3831, 3832, 3833, 3841, 3844, 3846, 3847, 3848, 3900, 3902, 3905, 3906, 3910, 3911, 3936, 3941, 3943, 3945, 3949, 3950, 3961, 3968, 3970, 3972, 3976, 3977
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\g_@@_iddots_int ..... 2029, 2571, 2572
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_in_env_bool .... 218, 305, 1024, 1025
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\c_@@_in_preamble_bool . 21, 22, 23, 562, 578
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\@@_info:n ..... 4330
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_initial_i_int ..... 2042, 2130, 2205, 2208, 2233, 2241, 2245, 2254, 2262, 2274, 2296, 2338, 2395, 2397, 2439, 2504, 2555, 2935, 2936, 2946, 3014, 3024, 3026
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_initial_j_int ..... ..... 2043, 2131, 2206, 2213, 2218, 2224, 2234, 2242, 2246, 2255, 2263, 2275, 2297, 2335, 2377, 2453, 2455, 2460, 2506, 2557, 2939, 2949, 2951, 3010, 3011, 3022
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_initial_open_bool ..... ..... 2046, 2207, 2211, 2214, 2221, 2227, 2231, 2247, 2333, 2375, 2392, 2402, 2430, 2437, 2449, 2502, 2553, 2695, 2742, 2933, 2940, 2952, 3008, 3015, 3027, 3105, 3692, 3731
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\@@_instruction_of_type:nn ..... ..... 824, 2785, 2800, 2815, 2836, 2858
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_inter_dots_dim ..... ..... 389, 390, 2051, 2700, 2707, 2718, 2726, 2733, 2738, 2750, 2758, 3722, 3725, 3753, 3755
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\g_@@_internal_code_after_tl ..... ..... 240, 1402, 1458, 2065, 2066, 3685, 4027, 4204
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\@@_j: ..... 3807, 3809, 3810, 3811, 3812, 3817, 3820, 3822, 3825, 3827, 3828, 3830, 3833, 3835, 3836, 3851, 3854, 3856, 3857, 3858, 3913, 3915, 3918, 3920, 3924, 3925, 3938, 3941, 3942, 3944, 3949, 3950, 3962, 3968, 3969, 3971, 3976, 3977
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_l_dim ..... ..... 2673, 2674, 2687, 2688, 2700, 2706, 2717, 2725, 2733, 2738, 2750, 2751, 2758, 2759
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_large_nodes_bool 414, 488, 2037, 2041
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\g_@@_last_col_found_bool ..... 272, 1001, 1198, 1266, 1759, 1788, 1854, 1961, 2018
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_last_col_int ..... ..... 270, 271, 634, 669, 671, 687, 699, 713, 1063, 1135, 1141, 1148, 1201, 1316, 1923, 1925, 1962, 1965, 2017, 2419, 2458, 2782, 2797, 2833, 2855, 4154, 4160, 4166, 4361, 4379
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_last_col_without_value_bool ... ..... 269, 668, 1963, 4364
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_last_empty_column_int ..... ..... 3618, 3631, 3637, 3650
\g_@@_env_int ..... 209, 210, 212, 1033, 1039, 1043, 1053, 1057, 1060, 1069, 1070, 1077, 1078, 1121, 1124, 1139, 1142, 1969, 1990, 2008, 2011, 2092, 3977	\l_@@_last_empty_row_int . 3600, 3614, 3634

<code>\l_@@_last_row_int</code> .. 266, 267, 465, 737, 896, 1061, 1106, 1116, 1123, 1130, 1186, 1190, 1193, 1210, 1232, 1646, 1647, 1822, 1823, 1863, 1864, 1984, 2322, 2364, 2812, 2827, 2849, 3073, 3337, 3345, 4102, 4162, 4529	<code>\l_@@_notes_code_before_tl</code> ..... 546, 1509
<code>\l_@@_last_row_without_value_bool</code> ... 268, 1118, 1188, 1982	<code>\@@_notes_label_in_list:n</code> 278, 293, 301, 558
<code>\l_@@_left_delim_dim</code> ..... 1154, 1158, 1163, 1612, 1843	<code>\@@_notes_label_in_tabular:n</code> . 277, 314, 555
<code>\l_@@_left_delim_tl</code> ..... 1015, 3721	<code>\l_@@_notes_para_bool</code> .. 544, 688, 707, 1510
<code>\l_@@_left_margin_dim</code> ..... 415, 491, 1171, 1844, 3711, 3929	<code>\@@_notes_style:n</code> ..... 276, 279, 293, 301, 317, 322, 552
<code>\l_@@_letter_for_dotted_lines_str</code> ... 652, 660, 661, 1357	<code>\l_@@_nullify_dots_bool</code> ..... 410, 486, 2789, 2804, 2819, 2841, 2863
<code>\l_@@_light_syntax_bool</code> ..... 398, 459, 1174, 1179, 1985	<code>\l_@@_number_of_notes_int</code> 275, 308, 318, 328
<code>\@@_light_syntax_i</code> ..... 1637, 1640	<code>\@@_old_CT@arc@</code> ..... 1026, 2100
<code>\@@_line</code> ..... 2079, 3089	<code>\@@_old_arraycolsep_dim</code> ..... 254
<code>\@@_line_i:nn</code> ..... 3096, 3103	<code>\@@_old_cdots</code> ..... 969, 2804
<code>\@@_line_with_light_syntax:n</code> ... 1651, 1655	<code>\@@_old_ddots</code> ..... 971, 2841
<code>\@@_line_with_light_syntax_i:n</code> ..... 1650, 1656, 1657	<code>\l_@@_old_iRow_int</code> ..... 241, 931, 2124
<code>\@@_math_toggle_token:</code> ..... 134, 782, 1816, 1833, 1858, 1874, 3999, 4246, 4250	<code>\@@_old_ialign:</code> ..... 858, 965, 2063
<code>\g_@@_max_cell_width_dim</code> ..... 788, 789, 1035, 1734, 3765, 3791	<code>\@@_old_iddots</code> ..... 972, 2863
<code>\l_@@_max_delimiter_width_bool</code> ..... 422, 458, 1262	<code>\l_@@_old_jCol_int</code> ..... 242, 934, 2125
<code>\c_@@_max_l_dim</code> ..... 2687, 2692	<code>\@@_old_ldots</code> ..... 968, 2789
<code>\l_@@_medium_nodes_bool</code> 413, 487, 2035, 4090	<code>\@@_old_multicolumn</code> ..... 2872, 2879
<code>\@@_message_hdotsfor:</code> 4370, 4378, 4385, 4393	<code>\@@_old_pgfulil@check@rerun</code> ..... 75, 79
<code>\@@_msg_new:nn</code> 17, 4299, 4308, 4375, 4382, 4390, 4398, 4403, 4410, 4415, 4420, 4425, 4430, 4436, 4441, 4447, 4452, 4458, 4463, 4470, 4472, 4478, 4485, 4492, 4497, 4503, 4508, 4515, 4521, 4527, 4535, 4537, 4543, 4768	<code>\@@_old_vdots</code> ..... 970, 2819
<code>\@@_msg_new:nnn</code> ..... 18, 4269, 4548, 4566, 4608, 4657, 4707, 4755	<code>\l_@@_parallelize_diags_bool</code> ..... 402, 403, 483, 2026, 2518, 2569
<code>\@@_msg_redirect_name:nn</code> ..... 19, 647	<code>\@@_patch_preamble:n</code> ..... 1294, 1336, 1371, 1376, 1405, 1422, 1441, 1452, 1460
<code>\@@_multicolumn:nnn</code> ..... 985, 2873	<code>\@@_patch_preamble_i:n</code> 1340, 1341, 1342, 1362
<code>\g_@@_multicolumn_cells_seq</code> ..... 994, 2892, 3825, 3833, 3955	<code>\@@_patch_preamble_ii:nn</code> ..... 1343, 1344, 1345, 1346, 1373
<code>\g_@@_multicolumn_sizes_seq</code> 995, 2894, 3956	<code>\@@_patch_preamble_iii:n</code> . 1347, 1378, 1386
<code>\g_@@_name_env_str</code> ..... 231, 236, 237, 1018, 1019, 1667, 1895, 1896, 1904, 1905, 1934, 1943, 1951, 2098, 4174, 4359	<code>\@@_patch_preamble_iii_i:n</code> .... 1381, 1383
<code>\l_@@_name_str</code> ..... 412, 532, 755, 758, 816, 819, 876, 879, 1119, 1128, 1131, 1137, 1146, 1149, 1683, 1684, 1705, 1706, 1723, 1724, 1754, 1755, 1780, 1783, 1799, 1802, 1972, 1976, 1993, 1997, 3946, 3949, 3973, 3976	<code>\@@_patch_preamble_iv:nnn</code> ..... 1348, 1349, 1350, 1408
<code>\g_@@_names_seq</code> ..... 217, 529, 531, 4766	<code>\@@_patch_preamble_v:nnnn</code> 1351, 1352, 1424
<code>\l_@@_nb_cols_int</code> ..... 4141, 4146, 4149, 4153, 4159, 4165	<code>\@@_patch_preamble_vi:n</code> ..... 1353, 1443
<code>\l_@@_nb_rows_int</code> ..... 4140, 4145, 4156	<code>\@@_patch_preamble_vii:n</code> ..... 1358, 1454
<code>\@@_newcolumnntype</code> ..... 902, 1281, 1282	<code>\@@_pgf_rect_node:nnn</code> ..... 361, 4092
<code>\@@_node_for_multicolumn:nn</code> .... 3957, 3964	<code>\@@_pgf_rect_node:nnnnn</code> ..... 336, 3940, 3967, 4053, 4087
<code>\@@_node_for_the_cell:</code> 795, 800, 1841, 1887	<code>\c_@@_pgfortikzpicture_tl</code> ..... 38, 42, 2106, 3122, 3694
<code>\@@_node_position:</code> .. 1072, 1074, 1080, 1082	<code>\@@_picture_position:</code> .... 1066, 1074, 1082
<code>\l_@@_notes_above_space_dim</code> ..... 408, 409	<code>\g_@@_pos_of_blocks_seq</code> ..... 258, 1038, 2895, 3435, 3457, 3514, 3664, 3994, 4214
<code>\l_@@_notes_bottomrule_bool</code> ..... 550, 690, 709, 1524	<code>\g_@@_pos_of_xdots_seq</code> ..... 259, 2272, 3436, 3459, 3516, 3564
<code>\l_@@_notes_code_after_tl</code> ..... 548, 1533	<code>\@@_pre_array:</code> ..... 925, 1153
	<code>\c_@@_preamble_first_col_tl</code> .... 1305, 1810
	<code>\c_@@_preamble_last_col_tl</code> .... 1317, 1850
	<code>\g_@@_preamble_tl</code> ..... 1279, 1289, 1292, 1302, 1305, 1314, 1317, 1326, 1331, 1364, 1368, 1375, 1388, 1410, 1426, 1445, 1449, 1456, 1621, 1648
	<code>\@@_pred:n</code> ..... 111, 133, 1925, 3467, 3480, 3524, 3537
	<code>\@@_provide_pgfsyspdfmark:</code> . 191, 200, 1012
	<code>\@@_put_box_in_flow:</code> ..... 1264, 1462, 1614
	<code>\@@_put_box_in_flow_bis:nn</code> .... 1263, 1581
	<code>\@@_put_box_in_flow_i:</code> ..... 1468, 1470
	<code>\@@_qpoint:n</code> ..... 213, 1488, 1490, 1493, 1542, 1544, 1568, 1570, 2335, 2338, 2344, 2347, 2377, 2385, 2395, 2397, 2439, 2445, 2453, 2455, 2504, 2506, 2512, 2514, 2555, 2557, 2563, 2565, 3130, 3133, 3150,

3154, 3167, 3169, 3186, 3188, 3201, 3205, 3225, 3231, 3233, 3237, 3260, 3262, 3271, 3273, 3368, 3370, 3373, 3394, 3396, 3402, 3422, 3425, 3493, 3495, 3497, 3550, 3552, 3554, 3702, 3706, 3713, 3749, 3752, 3754, 3846, 3856, 4043, 4045, 4047, 4049, 4069, 4084, 4105, 4106, 4112, 4227, 4229, 4232, 4234	1792, 1797, 1802, 1803, 2019, 2344, 2385, 2397, 2445, 2455, 2512, 2514, 2557, 2563, 3154, 3167, 3188, 3205, 3231, 3237, 3271, 3273, 3373, 3396, 3400, 3420, 3425, 3451, 3497, 3510, 3520, 3554, 3713, 3754, 3906, 3910, 3920, 3924, 4047, 4049, 4084, 4232, 4234
\l_@@radius_dim ..... 393, 394, 1457, 2050, 2351, 2352, 2767, 3680, 3704, 3750, 3751	\l_@@suffix_tl ..... 3868, 3879, 3889, 3892, 3941, 3949, 3950, 3968, 3976, 3977
\l_@@real_left_delim_dim 1583, 1598, 1613	\@@_tab_or_array_colsep: .. 142, 2337, 2346
\l_@@real_right_delim_dim 1584, 1610, 1616	\c_@@table_collect_begin_tl . 161, 163, 181
\@@rectanglecolor ..... 1095, 3248	\c_@@table_print_tl ..... 164, 165, 183
\@@renew_NC@rewrite@S: .... 172, 174, 1000	\l_@@tabular_width_dim ..... ..... 221, 851, 853, 1328, 1952
\@@renew_dots: ..... 909, 993	\g_@@tabularnotes_seq ..... ..... 274, 309, 1513, 1519, 1535
\l_@@renew_dots_bool ..... 484, 993, 4288	\@@_test_if_cell_in_a_block:nn ..... ..... 3603, 3621, 3639, 3659
\@@renew_matrix: ..... 637, 4120, 4290	\@@_test_if_cell_in_block:nnnnnnn ... ..... 3665, 3667
\@@restore_iRow_jCol: ..... 2099, 2122	\@@_test_if_hline_in_block:nnnn ..... ..... 3458, 3460, 3566
\@@revtex_array: ..... 835, 846	\@@_test_if_math_mode: .... 223, 1023, 1906
\c_@@revtex_bool ..... 46, 48, 51, 845	\@@_test_if_vline_in_block:nnnn ..... ..... 3515, 3517, 3577
\l_@@right_delim_dim ..... ..... 1155, 1159, 1165, 1615, 1884	\l_@@the_array_box ..... ..... 1167, 1170, 1503, 1505, 1506
\l_@@right_delim_tl ..... 1016, 3724	\c_@@tikz_loaded_bool ..... ..... 26, 37, 1086, 2067, 3349, 3733
\l_@@right_margin_dim ..... ..... 416, 493, 1183, 1885, 2462, 3718, 3932	\l_@@tikz_tl ..... 4004, 4052
\@@rotate: ..... 987, 3062	\@@true_c: ..... 182, 1353
\@@rotate_i: ..... 3066, 3068	\l_@@type_of_col_tl 672, 673, 675, 1935, 1937
\@@rotate_ii: ..... 3065, 3068, 3069	\c_@@types_of_matrix_seq ..... 4352, 4359
\@@rotate_iii: ..... 3069, 3070	\@@update_for_first_and_last_row: .. ..... 763, 790, 1108, 1835, 1876
\g_@@row_of_col_done_bool ..... ..... 244, 889, 1017, 1689	\@@_use_arraybox_with_notes: ... 1223, 1555
\g_@@row_total_int ..... 997, 1209, 1481, 1562, 1984, 1991, 1998, 2014, 2977, 3800, 3814, 3841, 3936, 4035, 4058, 4073, 4467	\@@_use_arraybox_with_notes_b: . 1220, 1539
\@@rowcolor ..... 1096, 3143, 3288, 3289	\@@_use_arraybox_with_notes_c: ..... ..... 1221, 1251, 1500, 1553, 1579
\@@rowcolor_tabular ..... 922, 3311	\@@_vdottedline:n ..... 1459, 3729
\@@rowcolors ..... 1097, 3283	\@@_vdottedline_i:n ..... 3736, 3741, 3745
\g_@@rows_seq . 1643, 1645, 1647, 1649, 1651	\@@_vline: ..... 1036
\l_@@rules_color_tl .. 243, 444, 1049, 1050	\@@_vline:nn ..... 1403, 3347
\@@set_CT@arc@: ..... 136, 1050	\@@_vline_i:nn ..... 3352, 3357, 3361
\@@set_CT@arc@_i: ..... 137, 138	\l_@@vlines_bool ..... 405, 476, 480, 1287, 1311, 1323, 1366, 1447, 2059
\@@set_CT@arc@_ii: ..... 137, 140	\@@_w: ..... 1281, 1351
\@@set_final_coords: ..... 2286, 2311	\g_@@width_first_col_dim ..... ..... 256, 1214, 1836, 1837
\@@set_final_coords_from_anchor:n .. ... 2302, 2350, 2390, 2433, 2448, 2517, 2568	\g_@@width_last_col_dim ..... ..... 255, 1268, 1877, 1878
\@@set_initial_coords: ..... 2281, 2300	\l_@@x_final_dim ..... 249, 2288, 2345, 2346, 2386, 2387, 2435, 2457, 2465, 2469, 2473, 2475, 2480, 2482, 2515, 2524, 2532, 2566, 2575, 2583, 2621, 2635, 2644, 2680, 2732, 2748, 3134, 3714, 3725, 3751
\@@set_initial_coords_from_anchor:n . ... 2291, 2341, 2382, 2432, 2442, 2509, 2560	\l_@@x_initial_dim ..... ..... 247, 2283, 2336, 2337, 2378, 2379, 2435, 2456, 2457, 2464, 2469, 2473, 2475, 2477, 2480, 2482, 2507, 2524, 2532, 2558, 2575, 2583, 2612, 2634, 2644, 2680, 2732, 2746, 2748, 2766, 2768, 3131, 3707, 3722, 3750
\@@set_seq_of_str_from_clist:Nn 4347, 4352	\l_@@xdots_color_tl .... 421, 434, 2326, 2368, 2411, 2495, 2546, 2602, 2981, 3056, 3093
\@@set_size:n ..... 4138, 4147	
\c_@@siunitx_loaded_bool ..... ..... 144, 148, 153, 171, 674	
\l_@@small_bool ..... 635, 677, 683, 701, 726, 937, 1817, 1859, 2048	
\@@standard_cline ..... 107, 974	
\@@standard_cline:w ..... 107, 108	
\l_@@standard_cline_bool ... 386, 451, 973	
\c_@@standard_tl 396, 397, 2594, 3726, 3756	
\g_@@static_num_of_col_int ..... ..... 260, 1206, 1295, 4394, 4406	
\l_@@stop_loop_bool ..... 2134, 2135, 2167, 2180, 2189, 2202, 2203, 2235, 2248, 2257	
\@@succ:n ..... 128, 132, 868, 874, 1490, 1772, 1778, 1783, 1784,	



<code>\l_@@_xdots_down_tl</code> . . .	438, 2618, 2628, 2663
<code>\l_@@_xdots_line_style_tl</code> . . . . .	
. . . . .	395, 397, 430, 2594, 2602, 3726, 3756
<code>\l_@@_xdots_shorten_dim</code> . . . . .	391,
. . . . .	392, 436, 2052, 2609, 2610, 2706, 2717, 2725
<code>\l_@@_xdots_up_tl</code> . . . .	439, 2614, 2627, 2653
<code>\l_@@_y_final_dim</code> . . . . .	250,
. . . . .	2289, 2348, 2352, 2399, 2403, 2405, 2446,
. . . . .	2513, 2526, 2529, 2564, 2577, 2580, 2621,
. . . . .	2635, 2643, 2682, 2737, 2756, 3135, 3705, 3755
<code>\l_@@_y_initial_dim</code> . . . . .	
. . . . .	248, 2284, 2339, 2351, 2398, 2399, 2403,
. . . . .	2405, 2440, 2505, 2526, 2531, 2556, 2577,
. . . . .	2582, 2612, 2634, 2643, 2682, 2737, 2754,
. . . . .	2756, 2766, 2769, 3132, 3703, 3704, 3705, 3753
<code>\</code> . . . . .	1634, 1656, 4154, 4160, 4166,
. . . . .	4271, 4272, 4305, 4314, 4407, 4412, 4417,
. . . . .	4422, 4427, 4433, 4454, 4460, 4467, 4475,
. . . . .	4481, 4488, 4495, 4500, 4512, 4518, 4524,
. . . . .	4536, 4540, 4545, 4550, 4551, 4569, 4570,
. . . . .	4611, 4612, 4660, 4661, 4710, 4711, 4761, 4762
<code>\{</code> . .	237, 1913, 4182, 4474, 4506, 4512, 4611, 4710
<code>\}</code> . .	237, 1913, 4182, 4474, 4506, 4512, 4611, 4710
<code>\ </code> . . . . .	1915, 4181
<code>\_</code> . . . . .	4373, 4378,
. . . . .	4385, 4393, 4394, 4405, 4406, 4466, 4467,
. . . . .	4471, 4499, 4506, 4510, 4523, 4530, 4531, 4539
<b>A</b>	
<code>\aboverulesep</code> . . . . .	1528
<code>\addtocounter</code> . . . . .	326
<code>\alph</code> . . . . .	276
<code>\arraycolsep</code> . . . . .	
. . . . .	143, 492, 494, 496, 850, 940, 1158, 1159,
. . . . .	1213, 1250, 1254, 1269, 2380, 2388, 3710, 3717
<code>\arrayrulecolor</code> . . . . .	89
<code>\arrayrulewidth</code> . . . . .	
. . . . .	113, 116, 126, 446, 754, 867, 869,
. . . . .	875, 897, 1245, 1257, 1290, 1369, 1396,
. . . . .	1450, 1547, 1573, 1696, 1698, 1704, 1714,
. . . . .	1716, 1722, 1745, 1747, 1753, 1771, 1773,
. . . . .	1779, 3152, 3153, 3155, 3168, 3170, 3187,
. . . . .	3189, 3203, 3204, 3206, 3230, 3232, 3235,
. . . . .	3236, 3238, 3261, 3264, 3265, 3272, 3274,
. . . . .	3366, 3378, 3392, 3417, 3426, 3491, 3548, 3791
<code>\arraystretch</code> . . . . .	939
<code>\AtBeginDocument</code> . . . . .	
. . . . .	23, 27, 67, 83, 145, 169, 280, 564,
. . . . .	580, 2102, 2773, 2911, 2987, 3085, 3118, 3688
<code>\AutoNiceMatrix</code> . . . . .	4183
<code>\AutoNiceMatrixWithDelims</code> . . .	4143, 4175, 4187
<b>B</b>	
<code>\baselineskip</code> . . . . .	92
<code>\bgroup</code> . . . . .	1014
<code>\Block</code> . . . . .	986, 4506
<code>\BNiceMatrix</code> . . . . .	4135
<code>\bNiceMatrix</code> . . . . .	4132
bool commands:	
<code>\bool_do_until:Nn</code> . . . . .	2135, 2203
<code>\bool_gset_false:N</code> . . . . .	798, 1001, 1017, 3575, 3586
<code>\bool_gset_true:N</code> . . . . .	1689, 1854,
. . . . .	2790, 2805, 2820, 2842, 2864, 2869, 3456, 3513
<code>\bool_if:NTF</code> . . . . .	135, 153, 566, 571, 582, 587,
. . . . .	723, 726, 863, 889, 892, 927, 937, 992, 993,
. . . . .	1013, 1024, 1034, 1051, 1086, 1100, 1102,
. . . . .	1168, 1188, 1204, 1266, 1273, 1297, 1366,
. . . . .	1447, 1524, 1678, 1692, 1710, 1728, 1741,
. . . . .	1767, 1788, 1790, 1817, 1859, 1961, 1963,
. . . . .	1982, 1985, 2004, 2026, 2041, 2048, 2058,
. . . . .	2059, 2060, 2067, 2270, 2402, 2404, 2518,
. . . . .	2569, 2789, 2804, 2819, 2841, 2863, 2883,
. . . . .	3461, 3486, 3518, 3543, 3613, 3630, 3648,
. . . . .	3774, 3784, 3985, 4090, 4194, 4317, 4327, 4364
<code>\bool_if:nTF</code> . . . . .	171, 282,
. . . . .	305, 1198, 1478, 2035, 3107, 3604, 3622, 3640
<code>\bool_lazy_all:nTF</code> . . . . .	
. . . . .	1307, 1319, 3568, 3579, 3669
<code>\bool_lazy_and:nnTF</code> . . . . .	
. . . . .	1731, 1818, 2016, 2391, 2626, 3226, 3256, 3434
<code>\bool_lazy_or:nnTF</code> . . . . .	
. . . . .	427, 1560, 1862, 2430, 2686, 4034
<code>\bool_lazy_or_p:nn</code> . . . . .	1821
<code>\bool_not_p:n</code> . . . . .	
. . . . .	1310, 1311, 1312, 1322, 1323, 1324, 1733, 2018
<code>\bool_set:Nn</code> . . . . .	2434
<code>\g_tmpa_bool</code> . . . .	3456, 3468, 3476, 3481,
. . . . .	3486, 3513, 3525, 3533, 3538, 3543, 3575, 3586
<code>\l_tmpb_bool</code> . . .	3609, 3626, 3644, 3663, 3676
box commands:	
<code>\box_clear_new:N</code> . . . . .	929, 1167
<code>\box_dp:N</code> . . . . .	748,
. . . . .	768, 787, 955, 964, 1113, 1465, 1592, 1605
<code>\box_ht:N</code> . . . . .	749, 770, 776, 785,
. . . . .	957, 959, 962, 1111, 1464, 1592, 1605, 3078
<code>\box_move_up:nn</code> . .	58, 60, 62, 1497, 1553, 1579
<code>\box_rotate:Nn</code> . . . . .	3072
<code>\box_set_dp:Nn</code> . . . . .	786, 1465
<code>\box_set_ht:Nn</code> . . . . .	784, 1464
<code>\box_use:N</code> . . . . .	329, 1416, 1419, 3079
<code>\box_use_drop:N</code> . . . .	792, 796, 813, 1437,
. . . . .	1467, 1497, 1498, 1503, 1506, 1842, 4109, 4114
<code>\box_wd:N</code> . . . . .	330, 789, 794, 1163, 1165,
. . . . .	1505, 1599, 1611, 1837, 1840, 1878, 1882, 4201
<code>\l_tmpa_box</code> . . . . .	
. . . . .	312, 329, 330, 1162, 1163, 1164, 1165,
. . . . .	1238, 1464, 1465, 1467, 1497, 1498, 1592, 1605
<code>\l_tmpb_box</code> . . . . .	1585, 1599, 1600, 1611
<b>C</b>	
<code>\c</code> . . . . .	190, 1301
<code>\Cdots</code> . . . . .	977, 2795, 2798
<code>\cdots</code> . . . . .	912, 969
<code>\cellcolor</code> . . . . .	921, 1094, 3309
<code>\chessboardcolors</code> . . . . .	1099
<code>\cline</code> . . . . .	127, 974, 975
clist commands:	
<code>\clist_map_inline:nn</code> .	1930, 3156, 3190, 3222
<code>\CodeAfter</code> . . . . .	990, 1637, 1640, 2080
<code>\color</code> . . . . .	93, 139, 141,
. . . . .	2320, 2323, 2326, 2362, 2365, 2368, 2411,
. . . . .	2417, 2420, 2495, 2546, 2975, 2978, 2981,
. . . . .	3050, 3053, 3056, 3093, 3149, 3185, 3221, 3254
<code>\colorlet</code> . . . . .	229, 230, 733, 740, 1827, 1867
<code>\columncolor</code> . . . . .	923, 1098, 2087, 3325





<code>\Hspace</code> .....	982	mode commands:	
<code>\hspace</code> .....	2870	<code>\mode_leave_vertical:</code> .....	1022, 1415
<code>\hss</code> .....	1352	msg commands:	
<b>I</b>		<code>\msg_error:nn</code> .....	12
<code>\ialign</code> .....	858, 942, 965, 2063	<code>\msg_error:nnn</code> .....	13
<code>\iddots</code> .....	980, 2848, 2849, 2854, 2855	<code>\msg_error:nnnn</code> .....	14, 4037
<code>\iddots</code> .....	53, 915, 972	<code>\msg_fatal:nn</code> .....	15
if commands:		<code>\msg_fatal:nnn</code> .....	16
<code>\if_mode_math:</code> .....	225	<code>\msg_info:nn</code> .....	4320
<code>\ifnum</code> .....	97	<code>\msg_new:nnn</code> .....	17
<code>\ifstandalone</code> .....	1030	<code>\msg_new:nnnn</code> .....	18
int commands:		<code>\msg_redirect_name:nnn</code> .....	20
<code>\int_case:nnTF</code> .....	2824, 2830, 2846, 2852	<code>\multicolumn</code> .....	985, 2872, 2876, 2908, 2928
<code>\int_compare:nNnTF</code> .....	110, 111, 123, 719, 720, 730, 737, 773, 894, 896, 1061, 1063, 1106, 1116, 1135, 1186, 1190, 1201, 1206, 1210, 1211, 1548, 1574, 1646, 1674, 1860, 2145, 2152, 2156, 2158, 2213, 2220, 2224, 2226, 2322, 2364, 2419, 2458, 2460, 2890, 2977, 3052, 3073, 3165, 3199, 3267, 3269, 3322, 3336, 3337, 3344, 3345, 4102, 4150, 4152, 4154, 4158, 4160, 4162, 4164, 4166	<code>\multispan</code> .....	111, 112, 124, 125
<code>\int_compare_p:n</code> .....	3227, 3228, 3257, 3258, 3671, 3672, 3673, 3674	<code>\myfiledate</code> .....	6
<code>\int_gadd:Nn</code> .....	2903	<code>\myfileversion</code> .....	7
<code>\int_gincr:N</code> .....	718, 746, 1033, 1365, 1421, 1440, 1446, 1765, 1855, 2520, 2571, 3771	<b>N</b>	
<code>\int_if_even:nTF</code> .....	3299	<code>\newcolumntype</code> .....	205, 206, 207
<code>\int_incr:N</code> .....	308, 1380	<code>\newcounter</code> .....	273
<code>\int_step_inline:nn</code> .....	3295, 3297	<code>\NewDocumentCommand</code> .....	284, 303, 662, 2777, 2792, 2807, 2822, 2844, 2915, 2991, 3089, 3143, 3179, 3215, 3248, 3283, 3293, 3306, 3311, 3320, 4143, 4183
<code>\int_step_inline:nnnn</code> .....	3601, 3619, 3634, 3637	<code>\NewDocumentEnvironment</code> .....	1010, 1618, 1627, 1892, 1902, 1932, 1941, 1949, 3769
iow commands:		<code>\NewExpandableDocumentCommand</code> .....	211, 3980
<code>\iow_now:Nn</code> .....	70, 193, 2006, 2007, 2009, 2024, 2088, 2089, 2095	<code>\newlist</code> .....	288, 295
<code>\iow_shipout:Nn</code> .....	1966, 1967, 1974, 1980, 1987, 1988, 1995, 2001, 3786, 3787, 3793	<code>\NiceArray</code> .....	1946, 1955
<code>\item</code> .....	1513, 1519	<code>\NiceArrayWithDelims</code> .....	1897, 1907
<b>K</b>		nicematrix commands:	
<code>\kern</code> .....	63	<code>\g_nicematrix_code_after_tl</code> .....	239, 535, 1642, 2081, 2082, 4255, 4263
keys commands:		<code>\NiceMatrixLastEnv</code> .....	211
<code>\keys_define:nn</code> .....	423, 442, 449, 503, 542, 594, 630, 664, 681, 694, 705, 3760, 4002, 4284	<code>\NiceMatrixOptions</code> .....	662, 4569
<code>\l_keys_key_tl</code> .....	4271, 4444, 4545, 4550, 4568, 4610, 4659, 4709	<code>\NiceMatrixoptions</code> .....	4487
<code>\keys_set:nn</code> .....	457, 656, 663, 1046, 1047, 1936, 1944, 1953, 2325, 2367, 2422, 2494, 2545, 2980, 3055, 3092, 3773, 4014	<code>\noalign</code> .....	92, 97, 116, 885, 948, 3680
<code>\l_keys_value_tl</code> .....	4483, 4490, 4757	<code>\nobreak</code> .....	298
<b>L</b>		<code>\normalbaselines</code> .....	936
<code>\Ldots</code> .....	976, 2780, 2783	<code>\nulldelimiterspace</code> .....	1599, 1611
<code>\ldots</code> .....	911, 968	<code>\numexpr</code> .....	132, 133
<code>\leaders</code> .....	113, 126	<b>O</b>	
<code>\left</code> .....	1241, 1588, 1603	<code>\omit</code> .....	110, 1676, 1688, 1764
legacy commands:		<code>\OnlyMainNiceMatrix</code> .....	988, 3328
<code>\legacy_if:nTF</code> .....	526	<b>P</b>	
<code>\line</code> .....	2079, 4474	<code>\par</code> .....	1515
<b>M</b>		peek commands:	
<code>\makebox</code> .....	1437	<code>\peek_meaning:NnTF</code> .....	137, 310, 905
<code>\mathinner</code> .....	55	<code>\peek_meaning_ignore_spaces:NnTF</code> .....	1620
		<code>\peek_meaning_remove_ignore_spaces:NnTF</code> .....	127
		<code>\peek_remove_spaces:n</code> .....	2888
		<code>\pgfextracty</code> .....	4105
		<code>\pgfgetlastxy</code> .....	372
		<code>\pgfpathcircle</code> .....	2765
		<code>\pgfpathlineto</code> .....	3375, 3381, 3405, 3427, 3500, 3557, 4236
		<code>\pgfpathmoveto</code> .....	3372, 3380, 3404, 3424, 3499, 3556, 4231
		<code>\pgfpathrectanglecorners</code> .....	3171, 3207, 3239, 3275
		<code>\pgfpointhead</code> .....	370
		<code>\pgfpointheadanchor</code> .....	214, 2293, 2304, 3822, 3830, 4063, 4078, 4094, 4095

`\pgfpointdiff` ..... 371, 1074, 1082  
`\pgfpointlineatime` ..... 2633  
`\pgfpointorigin` ..... 1682, 1798  
`\pgfpointscale` ..... 370  
`\pgfpointshapeborder` ..... 3130, 3133  
`\pgfrememberpicturepositiononpagetrue` ...  
     ..... 751, 804, 873,  
     1681, 1702, 1720, 1751, 1777, 1796, 2113,  
     2592, 2670, 3129, 3364, 3390, 3415, 3489,  
     3546, 3701, 3748, 3865, 3875, 3886, 4041, 4226  
`\pgfscope` ..... 2630, 4243  
`\pgfset` ..... 339, 364, 805, 4052, 4242  
`\pgfsetbaseline` ..... 803  
`\pgfsetlinewidth` .. 3366, 3392, 3417, 3491, 3548  
`\pgfsetrectcap` ..... 3367, 3393, 3492, 3549  
`\pgfsetroundcap` ..... 4239  
`\pgfsyspdfmark` ..... 196, 197  
`\pgftransformrotate` ..... 2637  
`\pgftransformshift` .....  
     ..... 345, 370, 2631, 4107, 4112, 4244, 4248  
`\pgfusepath` ..... 2657, 2667  
`\pgfusepathqfill` .. 2771, 3175, 3211, 3244, 3278  
`\pgfusepathqstroke` .....  
     ..... 3383, 3407, 3429, 3501, 3558, 4240  
`\phantom` ..... 2789, 2804, 2819, 2841, 2863  
`\pNiceMatrix` ..... 4123  
prg commands:  
    `\prg_do_nothing`: 157, 166, 172, 200, 948, 2080  
    `\prg_replicate:nn` ..... 318,  
     1760, 1761, 2928, 3376, 4153, 4156, 4159, 4165  
`\ProcessKeysOptions` ..... 4298  
`\ProvideDocumentCommand` ..... 53  
`\ProvidesExplPackage` ..... 4

## Q

`\quad` ..... 299  
quark commands:  
    `\q_stop` ..... 107, 108, 120, 121, 138, 140,  
     1050, 1294, 1354, 1637, 1640, 3083, 3097,  
     3098, 3138, 3160, 3161, 3194, 3195, 3224,  
     3255, 3266, 3959, 3966, 3981, 3982, 4138, 4147

## R

`\rectanglecolor` ..... 1095, 2086, 3315, 4019  
`\refstepcounter` ..... 327  
regex commands:  
    `\regex_const:Nn` ..... 190  
    `\regex_replace_all:NnN` ..... 1299  
`\relax` ..... 132, 133  
`\renewcommand` ..... 176  
`\RenewDocumentEnvironment` .....  
     ..... 4122, 4125, 4128, 4131, 4134  
`\RequirePackage` ..... 1, 3, 9, 10, 11  
`\right` ..... 1259, 1595, 1607  
`\rotate` ..... 987  
`\rowcolor` ..... 922, 1096  
`\rowcolors` ..... 1097

## S

`\savenotes` ..... 1013  
`\scriptstyle` .....  
     ..... 726, 1817, 1859, 2614, 2618, 2653, 2663  
seq commands:  
    `\seq_clear:N` ..... 1037, 1038, 4340

`\seq_clear_new:N` ..... 2008, 3590  
`\seq_count:N` ..... 1647  
`\seq_gclear:N` ..... 1535, 3564  
`\seq_gclear_new:N` .... 994, 995, 1643, 1659  
`\seq_gpop_left:NN` ..... 1649, 1661  
`\seq_gput_left:Nn` 531, 2892, 2894, 3994, 3995  
`\seq_gput_right:Nn` ... 309, 2272, 2895, 4214  
`\seq_gset_from_clist:Nn` ..... 2011  
`\seq_gset_split:Nnn` ..... 1645, 1660  
`\seq_if_empty:NTF` ..... 2064  
`\seq_if_empty_p:N` ..... 3435, 3436  
`\seq_if_exist:NTF` ..... 1053  
`\seq_if_in:NnTF` ..... 529, 3465,  
     3471, 3478, 3522, 3528, 3535, 3825, 3833, 4359  
`\seq_item:Nn` 1057, 1060, 1069, 1070, 1077, 1078  
`\seq_map_function:NN` ..... 1651  
`\seq_map_inline:Nn` ..... 1513, 1519,  
     1663, 3457, 3459, 3514, 3516, 3664, 4010, 4341  
`\seq_mapthread_function:NNN` ..... 3954  
`\seq_new:N` ..... 217, 257, 258, 259, 274  
`\seq_put_left:Nn` ..... 4343  
`\seq_put_right:Nn` ..... 3651  
`\seq_set_eq:NN` ..... 4345  
`\seq_set_from_clist:Nn` ..... 4349  
`\seq_use:Nnnn` ..... 4766  
`\l_tmpa_seq` ..... 4340, 4343, 4345  
`\setlist` ..... 289, 296, 567, 572, 583, 588  
skip commands:  
    `\skip_gadd:Nn` ..... 1736  
    `\skip_gset:Nn` ..... 1727  
    `\skip_gset_eq:NN` ..... 1734, 1735  
    `\skip_horizontal:N` .....  
     ..... 1171, 1172, 1183, 1184, 1213,  
     1214, 1249, 1250, 1253, 1254, 1268, 1269,  
     1290, 1369, 1450, 1457, 1612, 1613, 1615,  
     1616, 1677, 1696, 1698, 1714, 1716, 1738,  
     1745, 1747, 1766, 1771, 1773, 1794, 1806,  
     1843, 1844, 1845, 1847, 1879, 1884, 1885, 1886  
    `\skip_horizontal:n` ..... 330, 1392  
    `\skip_vertical:N` ..... 116, 867,  
     869, 1244, 1245, 1256, 1257, 1507, 1528, 3680  
    `\skip_vertical:n` ..... 3078  
    `\g_tmpa_skip` 1727, 1734, 1735, 1736, 1738, 1766  
    `\c_zero_skip` ..... 953  
`\space` ..... 236, 237  
`\stepcounter` ..... 316, 321  
str commands:  
    `\c_backslash_str` ..... 236  
    `\c_colon_str` ..... 661  
    `\str_case:nnTF` ..... 1218, 1338, 1472  
    `\str_gclear:N` ..... 2098  
    `\str_gset:Nn` .....  
     ... 1019, 1896, 1905, 1934, 1943, 1951, 4174  
    `\str_if_empty:NTF` ..... 755, 816, 876,  
     1018, 1119, 1137, 1683, 1705, 1723, 1754,  
     1780, 1799, 1895, 1904, 1972, 1993, 3946, 3973  
    `\str_if_eq:nnTF` ..... 78, 235, 521,  
     643, 856, 1357, 1385, 1466, 1557, 1667, 4260  
    `\str_if_eq_p:nn` ..... 429  
    `\str_new:N` ..... 231, 399, 412, 660  
    `\str_set:Nn` .....  
     ..... 232, 400, 515, 516, 517, 528, 652, 1558  
    `\str_set_eq:NN` ..... 532, 661

`\l_tmpa_str` ..... 528, 529, 531, 532  
`\strut` ..... 1513, 1519

## T

`\tabcolsep` .....  
 143, 849, 1249, 1253, 2380, 2388, 3710, 3717  
`\tabskip` ..... 953  
`\tabularnote` ..... 284, 303, 310, 4510, 4523  
`\tabularnotes` ..... 1518  
 T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> commands:  
`\@BTnormal` ..... 928  
`\@acol` ..... 839  
`\@acoll` ..... 837  
`\@acolr` ..... 838  
`\@addtopreamble` ..... 1036  
`\@array@array` ..... 841  
`\@arrayacol` ..... 837, 838, 839  
`\@arrayrule` ..... 1036  
`\@arstrutbox` ..... 748,  
 749, 955, 957, 959, 962, 964, 1416, 1419, 3078  
`\@currenvir` ..... 4260  
`\@halignto` ..... 840, 852, 853  
`\@height` ..... 100, 113, 126  
`\@ifclassloaded` ..... 47, 50, 4319, 4329  
`\@ifnextchar` ..... 999  
`\@ifpackageloaded` .....  
 ..... 29, 32, 35, 69, 85, 147, 4322, 4332  
`\@mainaux` ..... 70, 193, 1966, 1967, 1974,  
 1980, 1987, 1988, 1995, 2001, 2006, 2007,  
 2009, 2024, 2088, 2089, 2095, 3786, 3787, 3793  
`\@tabarray` ..... 854  
`\@tempswafalse` ..... 1285  
`\@tempswatruer` ..... 1284  
`\@temptokena` .. 156, 159, 178, 180, 1283, 1294  
`\@whiles` ..... 1285  
`\@width` ..... 100  
`\@xhline` ..... 103  
`\bBigg@` ..... 1162, 1164  
`\c@MaxMatrixCols` ..... 1924, 4387  
`\c@tabularnote` ..... 1502, 1536  
`\col@sep` .....  
 849, 850, 1677, 1736, 1794, 1806, 1847, 1879  
`\CT@arc` ..... 89, 90  
`\CT@arc@` .....  
 ... 88, 93, 101, 113, 126, 139, 141, 1026,  
 1529, 2100, 3363, 3388, 3414, 3448, 3747, 4238  
`\CT@everycr` ..... 946  
`\CT@row@color` ..... 948  
`\if@temp` ..... 1285  
`\NC@` ..... 904  
`\NC@find` ..... 157, 185  
`\NC@list` ..... 1285  
`\NC@rewrite@S` ..... 158, 176  
`\new@ifnextchar` ..... 999  
`\newcol@` ..... 906, 907  
`\nicematrix@redefine@check@rerun` .. 70, 73  
`\pgf@relevantforpicturesizefalse` .....  
 ..... 2114, 2593, 2671,  
 2762, 3148, 3184, 3220, 3253, 3365, 3391,  
 3416, 3490, 3547, 3866, 3876, 3887, 4042, 4225  
`\pgfsys@getposition` ..... 1066, 1072, 1080  
`\pgfsys@markposition` .....  
 868, 1065, 1679, 1697, 1715, 1746, 1772, 1792  
`\pgfutil@check@rerun` ..... 75, 76

`\reserved@a` ..... 102  
`\tikz@library@external@loaded` ..... 1027

tex commands:

`\tex_mkern:D` ..... 57, 59, 61, 64  
`\tex_the:D` ..... 180  
`\textfont` ..... 1495  
`\textsuperscript` ..... 277, 278  
`\the` ..... 132, 133, 1285, 1294  
`\thetabularnote` ..... 279  
`\tikzset` ..... 1029, 1031, 1088, 2069

tl commands:

`\tl_const:Nn` .. 38, 39, 42, 43, 396, 1810, 1850  
`\tl_count:n` ..... 651  
`\tl_gclear:N` ..... 1292, 2066, 2082  
`\tl_gclear_new:N` .....  
 ... 1002, 1003, 1004, 1005, 1006, 1007, 1008  
`\tl_gput_left:Nn` ..... 1305, 1314, 3324  
`\tl_gput_right:Nn` 826, 1317, 1326, 1330,  
 1364, 1368, 1375, 1388, 1402, 1410, 1426,  
 1445, 1449, 1456, 1458, 1642, 2917, 2993,  
 3308, 3313, 3685, 4017, 4027, 4204, 4255, 4263  
`\tl_gset:Nn` .. 159, 163, 165, 1279, 1289, 2091  
`\tl_if_blank:nTF` .... 3145, 3181, 3217, 3250  
`\tl_if_empty:N` .....  
 ... 1049, 2084, 3162, 3163, 3196, 3197, 4015  
`\tl_if_empty:nTF` .....  
 ..... 509, 632, 666, 685, 697, 711, 1629,  
 1656, 2326, 2368, 2411, 2495, 2546, 2981,  
 3056, 3093, 3149, 3185, 3221, 3254, 3986, 4372  
`\tl_if_empty_p:N` ..... 2627, 2628  
`\tl_if_eq:NNTF` ..... 2594, 3721, 3724  
`\tl_if_eq:nnTF` ..... 1632, 1634  
`\tl_if_exist:N` ..... 1039  
`\tl_if_in:NnTF` ..... 3159, 3193  
`\tl_item:Nn` ..... 162, 163, 165  
`\tl_lower_case:n` ..... 2883  
`\tl_map_inline:nn` ..... 1630  
`\tl_new:N` ..... 161,  
 164, 239, 240, 243, 245, 261, 395, 419, 421  
`\tl_put_left:Nn` ..... 928  
`\tl_put_right:Nn` ..... 511, 1042, 1108  
`\tl_range:nnn` ..... 78  
`\tl_set:Nn` .....  
 162, 3164, 3166, 3198, 3200, 3268, 3270, 3987  
`\tl_set_eq:NN` ..... 397, 3726, 3756  
`\tl_set_rescan:Nnn` .....  
 ..... 1644, 2776, 2914, 2990, 3088  
`\tl_to_str:n` ..... 4343  
`\g_tmpa_tl` ..... 159, 162, 165  
`\l_tmpb_tl` ..... 3141, 3163,  
 3164, 3165, 3166, 3167, 3197, 3198, 3199,  
 3200, 3205, 3228, 3233, 3234, 3237, 3258,  
 3262, 3263, 3269, 3270, 3273, 3454, 3467,  
 3473, 3480, 3495, 3497, 3511, 3520, 3524,  
 3530, 3532, 3537, 3552, 3572, 3573, 3583, 3584

token commands:

`\token_to_str` ..... 4449  
`\token_to_str:N` .....  
 4373, 4454, 4474, 4506, 4510, 4523, 4540, 4569

## U

`\unskip` ..... 1522



<b>11</b>	<b>The notes in the tabulars</b>	<b>19</b>
11.1	The footnotes . . . . .	19
11.2	The notes of tabular . . . . .	19
11.3	Customisation of the tabular notes . . . . .	21
11.4	Utilisation of <code>{NiceTabular}</code> with <code>threeparttable</code> . . . . .	23
<b>12</b>	<b>Other features</b>	<b>23</b>
12.1	Use of the column type <code>S</code> of <code>siunitx</code> . . . . .	23
12.2	Alignment option in <code>{NiceMatrix}</code> . . . . .	23
12.3	The command <code>\rotate</code> . . . . .	23
12.4	The option <code>small</code> . . . . .	24
12.5	The counters <code>iRow</code> and <code>jCol</code> . . . . .	24
12.6	The option <code>light-syntax</code> . . . . .	25
12.7	The environment <code>{NiceArrayWithDelims}</code> . . . . .	26
<b>13</b>	<b>Utilisation of Tikz with <code>nicematrix</code></b>	<b>26</b>
13.1	The nodes corresponding to the contents of the cells . . . . .	26
13.2	The “medium nodes” and the “large nodes” . . . . .	27
13.3	The “row-nodes” and the “col-nodes” . . . . .	28
<b>14</b>	<b>Tools for the developpers</b>	<b>29</b>
<b>15</b>	<b>Technical remarks</b>	<b>29</b>
15.1	Definition of new column types . . . . .	29
15.2	Diagonal lines . . . . .	30
15.3	The “empty” cells . . . . .	31
15.4	The option <code>exterior-arraycolsep</code> . . . . .	31
15.5	Incompatibilities . . . . .	31
<b>16</b>	<b>Examples</b>	<b>31</b>
16.1	Notes in the tabulars . . . . .	31
16.2	Dotted lines . . . . .	33
16.3	Dotted lines which are no longer dotted . . . . .	34
16.4	Width of the columns . . . . .	35
16.5	How to highlight cells of the matrix . . . . .	36
16.6	Direct use of the Tikz nodes . . . . .	38
<b>17</b>	<b>Implementation</b>	<b>40</b>
<b>18</b>	<b>History</b>	<b>149</b>
	<b>Index</b>	<b>154</b>