

# The package `nicematrix`\*

F. Pantigny  
fpantigny@wanadoo.fr

May 27, 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.

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

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

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

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller).

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).

---

\*This document corresponds to the version 4.1 of `nicematrix`, at the date of 2020/05/27.

# 1 The environments of this package

The package `nicematrix` defines the following new environments.

<code>{NiceTabular}</code>	<code>{NiceArray}</code>	<code>{NiceMatrix}</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}` and `{NiceTabular}` are similar to the environments `{array}` and `{tabular}` of the package `array` (which is loaded by `nicematrix`).

However, there are some small differences:

- For technical reasons, in the preamble of these environments, the user must use the letters `L`, `C` and `R`<sup>1</sup> instead of `l`, `c` and `r`, included in the commands `\multicolumn` and in the types of columns defined by `\newcolumntype`.
- \* In `{NiceArray}` (and its variants), the columns of type `w` (ex. : `wc{1cm}`) are composed in math mode whereas, in `{array}` of `array`, they are composed in text mode.

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.**

## 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`.

```

\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>The column types `L`, `C` and `R` are defined locally inside `{NiceTabular}` or `{NiceArray}` with `\newcolumntype` of `array`. This definition overrides an eventual previous definition.

### 3 The vertical position of the arrays

The package `nicematrix` provides a option `baseline` for the vertical position of the arrays. This option takes 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`<sup>2</sup>).

```
\begin{enumerate}
\item an item
\smallskip
\item \renewcommand{\arraystretch}{1.2}
$\begin{NiceArray}[t]{LCCCCC}
\hline
n & 0 & 1 & 2 & 3 & 4 & 5 \\
u_n & 1 & 2 & 4 & 8 & 16 & 32
\hline
\end{NiceArray}$
\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
$\begin{NiceArray}[t]{LCCCCC}
\toprule
n & 0 & 1 & 2 & 3 & 4 & 5 \\
\midrule
u_n & 1 & 2 & 4 & 8 & 16 & 32
\bottomrule
\end{NiceArray}$
\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.

---

<sup>2</sup>It's also possible to use `\firsthline` in the environments of `nicematrix`.

```

\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		fleurs	lys
arum	iris	jacinthe	muguet

It's also possible to use the command `\Block` in mathematical matrices.

```

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

```

$$\left[ \begin{array}{ccc|c} A & & & 0 \\ & \hspace*{1cm} & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array} \right]$$

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.

```

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

```

$$\left[ \begin{array}{ccc|c} A & & & 0 \\ & \hspace*{1cm} & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array} \right]$$

## 5 The rules

The usual techniques for the rules may be used in the environments of `nicematrix` (excepted `\vline`).

### 5.1 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 22.

## 5.2 A remark about `\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.

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

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

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

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

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

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

## 5.3 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).

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

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

However, there is a difference between the key `vlines` and the use of the specifier “|” in the preamble of the environment: the rules drawn by `vlines` completely cross the double-rules drawn by `\hline\hline` (you don’t need `hhline`).

```
$\begin{NiceMatrix}[vlines] \hline
a & b & c & d \\ \hline \hline
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\ \hline
\end{NiceMatrix}$
```

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

If you use `booktabs` (which provides `\toprule`, `\midrule`, `\bottomrule`, etc.) and you really want to draw vertical rules (something opposed to the spirit of `booktabs`), you should remark that the key `vlines` in compatible with `booktabs`.

```
$\begin{NiceMatrix}[vlines] \toprule
a & b & c & d \\ \midrule
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\ \bottomrule
\end{NiceMatrix}$
```

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

## 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

## 5.5 La commande \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

## 5.6 Dotted rules

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

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

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

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

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

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

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.

<sup>3</sup>In fact, when the key `hvlines` is in force, the rules are also not drawn in the virtual blocks delimited by cells relied par dotted lines (cf. p. 15).

<sup>4</sup>The author of this document considers that type of construction as graphically poor.

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

## 6 The color of the rows and columns

With the classical package `colortbl`, it's possible to color the cells, rows and columns of a tabular. However, the resulting PDF is not always perfectly displayed by the PDF viewers, in particular in conjunction with rules. With some PDF viewers, some vertical rules seem to vanish. On the other side, some thin horizontal white lines may appear in some circumstances.

The package `nicematrix` provides similar tools which do not present these drawbacks. It provides a key `code-before`<sup>6</sup> for some code which will be executed *before* the drawing of the tabular. In this `code-before`, new commands are available: `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors` and `\chessboardcolors`.

These commands are independent of `colortbl`.<sup>7</sup>

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

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

This command takes in 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).

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

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

<sup>6</sup>There is also a key `code-after` : see p. 16.

<sup>7</sup>Thus, it's possible to color the rules, the cells, the rows, the columns, etc. without loading `colortbl`.

- 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>8</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.

```
\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 mandatory arguments two colors and 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. 16).

One should remark that these commands are compatible with the commands `debooktabs` (`\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`.

## 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`. In `{NiceTabular}`, the cells of such columns are composed in `texte` mode but, in `{NiceArray}`,

<sup>8</sup>The command `\rowcolors` of `color` is available when `xcolor` is loaded with the option `table`.



`{pNiceArray}`, etc., they are composed in math mode (whereas, in `{array}` of `array`, they are composed in text mode).

```
\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>

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

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

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}
```

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

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

Several compilations may be necessary to achieve the job.

<sup>9</sup>The result is achieved with only one compilation (but Tikz will have written informations in the `.aux` file and a message requiring a second compilation will appear).

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

## 8 The exterior rows and columns

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

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

```

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

```

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

The dotted lines have been drawn with the tools presented p. 11.

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. 18) is used, `nicematrix` has, in any case, to load the whole body of the environment (and that’s why it’s not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows (but not the number of columns).
  - In the other cases, `nicematrix` compute the number of rows and columns during the first compilation and write the result in the `aux` file for the next run.  
*However, it’s possible to provide the number of the last row and the number of the last column as values of the options `last-row` and `last-col`, tending to an acceleration of the whole compilation of the document. That’s what we will do throughout the rest of the document.*

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

```

\NiceMatrixOptions{code-for-first-row = \color{red},
                  code-for-first-col = \color{blue},
                  code-for-last-row = \color{green},
                  code-for-last-col = \color{magenta}}

```

```

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

```

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

### Remarks

- As shown in the previous example, the horizontal and 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 22.
- 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. 8) 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}

```

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

<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. 23).

<sup>13</sup>It's also possible to change the color of all theses dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 14.

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

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

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

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

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

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

In the first column of this exemple, there are two instructions `\Vdots` but only one dotted line is drawn (there is no overlapping graphic objects in the resulting PDF<sup>14</sup>).

In fact, in this exemple, 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}
```

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.<sup>15</sup>

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

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

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

## 9.1 The option `nullify-dots`

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

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

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

<sup>14</sup>And it's not possible to draw a `\Ldots` and a `\Cdots` line between the same cells.

<sup>15</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. 8

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

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

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

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

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

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

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

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

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

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

## 9.2 The command `\Hdotsfor`

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

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

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

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

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

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

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

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

## 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>16</sup>

<sup>16</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. 16) 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. 16) 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. 10.

### 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>17</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}
```

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

## 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.

```
\NiceMatrixOptions{nullify-dots}
\begin{pNiceMatrix}[rules/color=gray,hvlines,margin]
0      & \Cdots & & & & 0      & \\
1      & \Cdots & & & 1    & 2      & \\
0      & \Ddots & & & \Vdots & \Vdots & \\
\Vdots & \Ddots & & & & & \\
      & & & & & & \\
0      & \Cdots & & 0    & 1    & 2      & \\
\end{pNiceMatrix}
```

$$\left( \begin{array}{cccccc|cc} 0 & \cdots & \cdots & \cdots & 0 & & & \\ 1 & \cdots & \cdots & \cdots & 1 & & 2 & \\ 0 & \cdots & \cdots & \cdots & & \cdots & & \\ \vdots & \vdots & \vdots & \vdots & & \vdots & & \\ \vdots & \vdots & \vdots & \vdots & & \vdots & & \\ 0 & \cdots & \cdots & 0 & 1 & & 2 & \end{array} \right)$$

<sup>17</sup>The first reason of this behaviour is that the PDF format includes a description for dashed lines. The lines specified with this descriptor are displayed very efficiently by the PDF readers. It’s easy, starting from these dashed lines, to create a line composed by square dots whereas a line of rounded dots needs a specification of each dot in the PDF file.

## 10 The code-after

The option `code-after` may be used to give some code that will be excuted after the construction of the matrix.<sup>18</sup>

A special command, called `\line`, is available to draw directly dotted lines between nodes. It takes two arguments for the two cells to rely, both of the form  $i$ - $j$  where  $i$  is the number of row and  $j$  is the number of column. It may be used, for example, to draw a dotted line between two adjacents 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}
```

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

For the readability of the code, an alternative syntax is provided: it's possible to give the instructions of the `\code-after` at the end of the environment, after the keyword `\CodeAfter`. For an example, cf. p. 27.

## 11 Other features

### 11.1 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`.

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

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

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

### 11.2 Aligement option in {NiceMatrix}

The environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.) provide two options `l` and `r` (equivalent at `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>

<sup>18</sup>There is also a key `code-before` described p. 7.

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



### 11.3 The command `\rotate`

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

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

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

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

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

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

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

### 11.4 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`).

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

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

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.

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

In the `code-before` (cf. p. 7) and in the `code-after` (cf. p. 16), `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).

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

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

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 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 eventual exterior rows and columns).

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

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

## 11.6 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>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.

<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

## 11.7 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.

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

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

## 12 Utilisation of Tikz with `nicematrix`

### 12.1 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. 27).

---

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.

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

## 12.2 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.

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

The names of the “large nodes” are constructed by adding the suffix “-large” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.<sup>23</sup>

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

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>

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

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.

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

**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).

<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. 10).

<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 is a 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

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

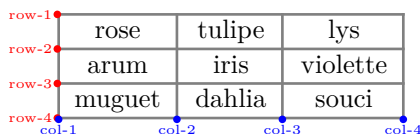
fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

## 12.3 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`.



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

```

## 13 Technical remarks

### 13.1 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 eventual 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:

```

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

```

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

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.

### 13.2 Diagonal lines

By default, all the diagonal lines<sup>26</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):

```

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

```

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

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

<sup>26</sup>We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

```

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

```

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

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

The same example without parallelization:

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

### 13.3 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`.

### 13.4 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>27</sup>. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hspace -\arraycolsep`<sup>28</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).

<sup>27</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>28</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

## 13.5 Incompatibilities

The package `nicematrix` is not compatible with `threeparttable`.

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

## 14 Examples

### 14.1 Dotted lines

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

```


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


```

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

```


$$\begin{matrix} 1 & & \cdots & & 1 \\ \vdots & & & & 0 \\ & & \iddots & & \vdots \\ 1 & & 0 & & \cdots & & 0 \end{matrix}$$


```

An example with `\multicolumn`:

```


$$\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \cdots & & \multicolumn{6}{C}{10 \text{ other rows}} & & \cdots \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{matrix}$$


```

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



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}
```

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

An example for the resultant of two polynoms:

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

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

An example for a linear system:

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

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

## 14.2 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}}
$\begin{pNiceMatrix}[first-row,first-col]
  & & \Ldots[line-style={solid,<->},shorten=0pt]^{n \text{ columns}} \\\
  & 1 & 1 & 1 & \Ldots & 1 \\\
  & 1 & 1 & 1 & & 1 \\\
  \Vdots[line-style={solid,<->}]_{n \text{ rows}} & 1 & 1 & 1 & & 1 \\\
  & 1 & 1 & 1 & & 1 \\\
  & 1 & 1 & 1 & \Ldots & 1
\end{pNiceMatrix}$
```

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

## 14.3 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}
```

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

## 14.4 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 correspondent 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>29</sup>

<sup>29</sup>For the command `\cline`, see the remark p. 5.

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}}

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

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

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`.

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

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

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}
```

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

## 14.5 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}
```

$$L_i \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \dots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} b_{11} & \dots & b_{1n} \\ \vdots & & \vdots \\ b_{n1} & \dots & b_{nn} \end{bmatrix} = \begin{bmatrix} \vdots & & \vdots \\ \dots & c_{ij} & \dots \\ \vdots & & \vdots \end{bmatrix}$$

## 15 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   {Mathematical matrices with PGF/TikZ}
```

The version of 2020/02/08 of `expl3` has replaced `\l_keys_key_tl` by `\l_keys_key_str`. We have immediately changed in this file. Now, you test the existence of `\l_keys_key_str` in order to detect whether the version of LaTeX used by the final user is up to date.

```
9 \msg_new:nnn { nicematrix } { expl3-too-old }
10 {
11   Your-version-of-LaTeX-(especially-expl3)-is-too-old.~
12   You-can-go-on-but-you-will-probably-have-other-errors~
13   if-you-use-the-functionalities-of-nicematrix.
14 }
15 \cs_if_exist:NF \l_keys_key_str
16 { \msg_error:nn { nicematrix } { expl3-too-old } }
```

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

We load some packages.

```

17 \RequirePackage { array }
18 \RequirePackage { amsmath }
19 \RequirePackage { xparse }

20 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { nicematrix } }
21 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { nicematrix } }
22 \cs_new_protected:Npn \@@_error:nnn { \msg_error:nnnn { nicematrix } }
23 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { nicematrix } }
24 \cs_new_protected:Npn \@@_fatal:nn { \msg_fatal:nnn { nicematrix } }
25 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { nicematrix } }
26 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { nicematrix } }

27 \cs_new_protected:Npn \@@_msg_redirect_name:nn
28   { \msg_redirect_name:nnn { nicematrix } }

```

## Technical definitions

```

29 \bool_new:N \c_@@_booktabs_loaded_bool
30 \bool_new:N \c_@@_tikz_loaded_bool
31 \AtBeginDocument
32   {
33     \ifpackageloaded { booktabs }
34       { \bool_set_true:N \c_@@_booktabs_loaded_bool }
35     { }
36     \ifpackageloaded { tikz }
37     {

```

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.

```

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

```

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.

```

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

```

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.



```

54 \ProvideDocumentCommand \iddots { }
55 {
56   \mathinner
57   {
58     \tex_mkern:D 1 mu
59     \box_move_up:nn { 1 pt } { \hbox:n { . } }
60     \tex_mkern:D 2 mu
61     \box_move_up:nn { 4 pt } { \hbox:n { . } }
62     \tex_mkern:D 2 mu
63     \box_move_up:nn { 7 pt }
64     { \vbox:n { \kern 7 pt \hbox:n { . } } }
65     \tex_mkern:D 1 mu
66   }
67 }

```

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.

```

68 \AtBeginDocument
69 {
70   \@ifpackageloaded { booktabs }
71   { \iow_now:Nn \mainaux \nicematrix@redefine@check@rerun }
72   { }
73 }
74 \cs_set_protected:Npn \nicematrix@redefine@check@rerun
75 {
76   \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`).

```

77   \cs_set_protected:Npn \pgfutil@check@rerun ##1 ##2
78   {
79     \str_set:Nx \l_tmpa_str { \tl_range:nnn { ##1 } 1 3 }
80     \str_if_eq:VnF \l_tmpa_str { nm- }
81     { \@_old_pgful@check@rerun { ##1 } { ##2 } }
82   }
83 }

```

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

```

84 \bool_new:N \c_@@_colortbl_loaded_bool
85 \AtBeginDocument
86 {
87   \@ifpackageloaded { colortbl }
88   { \bool_set_true:N \c_@@_colortbl_loaded_bool }
89   {

```

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.

```

90   \cs_set_protected:Npn \CT@arc@ { }
91   \cs_set:Npn \arrayrulecolor #1 # { \CT@arc@ { #1 } }
92   \cs_set:Npn \CT@arc@ #1 #2
93   {
94     \dim_compare:nNt \baselineskip = \c_zero_dim \noalign
95     { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }
96   }
97   \cs_set:Npn \hline
98   {
99     \noalign { \ifnum 0 = ` } \fi
100     \cs_set_eq:NN \hskip \vskip
101     \cs_set_eq:NN \vrule \hrule
102     \cs_set_eq:NN \@width \@height

```

```

103         { \CT@arc@ \vline }
104         \futurelet \reserved@a
105         \@xhline
106     }
107 }
108 }

```

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.

```

109 \cs_set:Npn \@@_standard_cline #1 { \@@_standard_cline:w #1 \q_stop }
110 \cs_set:Npn \@@_standard_cline:w #1-#2 \q_stop
111 {
112     \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
113     \int_compare:nNnT { #1 } > 1 { \multispan { \@@_pred:n { #1 } } & }
114     \multispan { \int_eval:n { #2 - #1 + 1 } }
115     { \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`.

```

116     \everycr { }
117     \cr
118     \noalign { \skip_vertical:N -\arrayrulewidth }
119 }

```

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.

```

120 \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 `\exp_arg:Ne` and not `\exp_arg:Nx`.

```

121 { \exp_args:Ne \@@_cline_i:nn \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*.

```

122 \cs_set:Npn \@@_cline_i:nn #1 #2 { \@@_cline_i:w #1-#2 \q_stop }
123 \cs_set:Npn \@@_cline_i:w #1-#2-#3 \q_stop
124 {

```

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).

```

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

```

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

```

129     \peek_meaning_remove_ignore_spaces:NTF \cline
130     { & \exp_args:Ne \@@_cline_i:nn { \@@_succ:n { #3 } } }
131     { \everycr { } \cr }
132 }
133
134

```

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

```

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

```

The following command is a small shortcut.

```

137 \cs_new:Npn \@@_math_toggle_token:
138 { \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token }

```

```

139 \cs_new_protected:Npn \@@_set_CT@arc@:

```

```

140 { \peek_meaning:NTF [ \@@_set_CT@arc@_i: \@@_set_CT@arc@_ii: }
141 \cs_new_protected:Npn \@@_set_CT@arc@_i: [ #1 ] #2 \q_stop
142 { \cs_set:Npn \CT@arc@ { \color [ #1 ] { #2 } } }
143 \cs_new_protected:Npn \@@_set_CT@arc@_ii: #1 \q_stop
144 { \cs_set:Npn \CT@arc@ { \color { #1 } } }

```

## The column S of siunitx

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

```

145 \bool_new:N \c_@@_siunitx_loaded_bool
146 \AtBeginDocument
147 {
148   \ifpackageloaded { siunitx }
149     { \bool_set_true:N \c_@@_siunitx_loaded_bool }
150     { }
151 }

```

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} }
    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:`.

```

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

```

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

```
158 \cs_set_eq:NN \NC@find \prg_do_nothing:
159 \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`).

```
160 \tl_gset:NV \g_tmpa_tl \@temptokena
161 \group_end:
162 \tl_new:N \c_@@_table_collect_begin_tl
163 \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
164 \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
165 \tl_new:N \c_@@_table_print_tl
166 \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).

```
167 \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
168 }
169 }
```

The command `\@@_renew_NC@rewrite@S:` will be used in each environment of `nicematrix` in order to “rewrite” the `S` column in each environment (only if the boolean `\c_@@_siunitx_loaded_bool` is raised, of course).

```
170 \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
171 {
172   \renewcommand*{\NC@rewrite@S}[1] []
173   {
174     \@temptokena \exp_after:wN
175     {
176       \tex_the:D \@temptokena
177       > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }
178       c
179       < { \c_@@_table_print_tl \@@_end_Cell: }
180     }
181     \NC@find
182   }
183 }
```

## Parameters

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.

```
184 \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).

```
185 \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.

```
186 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
187 { \int_use:N \g_@@_env_int }
```

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

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

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

```
190 \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).

```
191 \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`.

```
192 \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.

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

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

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

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

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

```
196 \cs_new_protected:Npn \@@_test_if_math_mode:
197 {
198   \if_mode_math: \else:
199     \@@_fatal:n { Outside-math-mode }
200   \fi:
201 }
```

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

```
202 \colorlet { nicematrix-last-col } { . }
203 \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`).

```
204 \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*.

```
205 \str_new:N \g_@@_com_or_env_str
206 \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.

```
207 \cs_new:Npn \@@_full_name_env:
208 {
209   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
210     { command \space \c_backslash_str \g_@@_name_env_str }
211     { environment \space \{ \g_@@_name_env_str \} }
212 }
```

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`).

```
213 \tl_new:N \g_@@_code_after_tl
```

The following token list has a function similar to `\g_@@_code_after_tl` but it is used internally by `nicematrix`. In fact, we have to distinguish between `\g_@@_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.

```
214 \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.

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

```
216 \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.

```
217 \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).

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

The following flag will be raised when the key `code-before` is used in the environment. Indeed, if there is a `code-before` in the environment, we will manage to have the `row` nodes and the `col` nodes available *before* the creation of the array.

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

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

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

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

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

```
223 \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`).

```
224 \dim_zero_new:N \l_tmpc_dim
```

```
225 \dim_zero_new:N \l_tmpd_dim
```

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

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

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

```
227 \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”.

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

```
229 \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.

```
230 \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}`.

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

The sequence `\g_@@_pos_of_blocks_seq` will be used by the test of non-overlapping of two blocks and when we will draw the rules required by the key `hvlines` (these rules won't be drawn within the blocks).

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

```
232 \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).

## 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.

```
233 \int_new:N \l_@@_first_row_int
234 \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.

```
235 \int_new:N \l_@@_first_col_int
236 \int_set:Nn \l_@@_first_col_int 1
```

### • Last row

The counter `\l_@@_last_row_int` is the number of the eventual “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).

```
237 \int_new:N \l_@@_last_row_int
238 \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>30</sup>

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

<sup>30</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.

Idem for `\l_@@_last_col_without_value_bool`

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

- **Last column**

For the potential “last column”, we use an integer. A value of  $-2$  means that there is no last column. A value of  $-1$  means that there is a last column but we don’t know its value because the user has used the option `last-col` without value (it’s possible in an environment without preamble like `{pNiceMatrix}`). A value of  $0$  means that the option `last-col` has been used in an environment with preamble (like `{pNiceArray}`).

```
241 \int_new:N \l_@@_last_col_int
242 \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.

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

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

## 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.

```
244 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
245 {
246   \begin { pgfscope }
247   \pgfset
248   {
249     outer-sep = \c_zero_dim ,
250     inner-sep = \c_zero_dim ,
251     minimum-size = \c_zero_dim
252   }
253   \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
254   \pgfnode
255   { rectangle }
256   { center }
257   {
258     \vbox_to_ht:nn
259     { \dim_abs:n { #5 - #3 } }
260     {
261       \vfill
262       \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
263     }
264   }
265   { #1 }
266   { }
267   \end { pgfscope }
268 }
```



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.

```

269 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
270 {
271   \begin { pgfscope }
272   \pgfset
273   {
274     outer-sep = \c_zero_dim ,
275     inner-sep = \c_zero_dim ,
276     minimum-size = \c_zero_dim
277   }
278   \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
279   \pgfpointdiff { #3 } { #2 }
280   \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
281   \pgfnode
282   { rectangle }
283   { center }
284   {
285     \vbox_to_ht:nn
286     { \dim_abs:n \l_tmpb_dim }
287     { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
288   }
289   { #1 }
290   { }
291   \end { pgfscope }
292 }

```

## The options

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`.

```

293 \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`).

```

294 \dim_new:N \l_@@_cell_space_top_limit_dim
295 \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.

```

296 \dim_new:N \l_@@_inter_dots_dim
297 \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).

```

298 \dim_new:N \l_@@_xdots_shorten_dim
299 \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.

```

300 \dim_new:N \l_@@_radius_dim
301 \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.

```
302 \tl_new:N \l_@@_xdots_line_style_tl
303 \tl_const:Nn \c_@@_standard_tl { standard }
304 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

```
305 \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).

```
306 \str_new:N \l_@@_baseline_str
307 \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`).

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

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

```
309 \bool_new:N \l_@@_parallelize_diags_bool
310 \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`. 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`).

```
311 \bool_new:N \l_@@_hlines_bool
312 \bool_new:N \l_@@_vlines_bool
313 \bool_new:N \l_@@_hvlines_bool
```

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.).

```
314 \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).

```
315 \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.

```
316 \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”.

```
317 \bool_new:N \l_@@_medium_nodes_bool
318 \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`).

```
319 \dim_new:N \l_@@_left_margin_dim
320 \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`.

```
321 \dim_new:N \l_@@_extra_left_margin_dim
322 \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.

```
323 \tl_new:N \l_@@_end_of_row_tl
324 \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 “:”.

```
325 \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.

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

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

```
327 \keys_define:nn { NiceMatrix / xdots }
328 {
329   line-style .code:n =
330   {
331     \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.

```
332   { \cs_if_exist_p:N \tikzpicture }
333   { \str_if_eq_p:nn { #1 } { standard } }
334   { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
335   { \@@_error:n { bad-option-for-line-style } }
336   } ,
337   line-style .value_required:n = true ,
338   color .tl_set:N = \l_@@_xdots_color_tl ,
339   color .value_required:n = true ,
340   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
341   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 `_`.

```
342   down .tl_set:N = \l_@@_xdots_down_tl ,
343   up .tl_set:N = \l_@@_xdots_up_tl ,
344   unknown .code:n = \@@_error:n { Unknown-option-for-~xdots }
345 }
```

```

346 \keys_define:nn { NiceMatrix / rules }
347 {
348   color .tl_set:N = \l_@@_rules_color_tl ,
349   color .value_required:n = true ,
350   width .dim_set:N = \arrayrulewidth ,
351   width .value_required:n = true
352 }
353 \keys_define:nn { NiceMatrix / Global }
354 {
355   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
356   standard-cline .default:n = true ,
357   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
358   cell-space-top-limit .value_required:n = true ,
359   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
360   cell-space-bottom-limit .value_required:n = true ,
361   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
362   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
363   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
364   light-syntax .default:n = true ,
365   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
366   end-of-row .value_required:n = true ,
367   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
368   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
369   last-row .int_set:N = \l_@@_last_row_int ,
370   last-row .default:n = -1 ,
371   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
372   code-for-first-col .value_required:n = true ,
373   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
374   code-for-last-col .value_required:n = true ,
375   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
376   code-for-first-row .value_required:n = true ,
377   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
378   code-for-last-row .value_required:n = true ,
379   hlines .bool_set:N = \l_@@_hlines_bool ,
380   vlines .bool_set:N = \l_@@_vlines_bool ,
381   hvlines .code:n =
382   {
383     \bool_set_true:N \l_@@_hvlines_bool
384     \bool_set_true:N \l_@@_vlines_bool
385     \bool_set_true:N \l_@@_hlines_bool
386   } ,
387   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.

```

388   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
389   renew-dots .value_forbidden:n = true ,
390   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
391   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
392   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
393   create-extra-nodes .meta:n =
394   { create-medium-nodes , create-large-nodes } ,
395   left-margin .dim_set:N = \l_@@_left_margin_dim ,
396   left-margin .default:n = \arraycolsep ,
397   right-margin .dim_set:N = \l_@@_right_margin_dim ,
398   right-margin .default:n = \arraycolsep ,
399   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
400   margin .default:n = \arraycolsep ,
401   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
402   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
403   extra-margin .meta:n =
404   { extra-left-margin = #1 , extra-right-margin = #1 } ,

```

```

405     extra-margin .value_required:n = true
406 }

```

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

```

407 \keys_define:nn { NiceMatrix / Env }
408 {
409     code-before .code:n =
410     {
411         \tl_if_empty:nF { #1 }
412         {
413             \tl_set:Nn \l_@@_code_before_tl { #1 }
414             \bool_set_true:N \l_@@_code_before_bool
415         }
416     } ,

```

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

```

417     c .code:n = \str_set:Nn \l_@@_baseline_str c ,
418     t .code:n = \str_set:Nn \l_@@_baseline_str t ,
419     b .code:n = \str_set:Nn \l_@@_baseline_str b ,
420     baseline .tl_set:N = \l_@@_baseline_str ,
421     baseline .value_required:n = true ,
422     columns-width .code:n =
423     { \str_if_eq:nnTF { #1 } { auto }
424       { \bool_set_true:N \l_@@_auto_columns_width_bool }
425       { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
426     columns-width .value_required:n = true ,
427     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.

```

428     \legacy_if:nF { measuring@ }
429     {
430         \str_set:Nn \l_tmpa_str { #1 }
431         \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
432         { \@@_error:nn { Duplicate-name } { #1 } }
433         { \seq_gput_left:N \g_@@_names_seq \l_tmpa_str }
434         \str_set_eq:NN \l_@@_name_str \l_tmpa_str
435     } ,
436     name .value_required:n = true ,
437     code-after .tl_gset:N = \g_@@_code_after_tl ,
438     code-after .value_required:n = true ,
439 }

```

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

```

440 \keys_define:nn { NiceMatrix }
441 {
442     NiceMatrixOptions .inherit:n =
443     {
444         NiceMatrix / Global ,
445     } ,
446     NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
447     NiceMatrix .inherit:n =
448     {
449         NiceMatrix / Global ,
450         NiceMatrix / Env ,
451     } ,
452     NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
453     NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
454     NiceTabular .inherit:n =

```

```

455     {
456       NiceMatrix / Global ,
457       NiceMatrix / Env
458     } ,
459     NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
460     NiceTabular / rules .inherit:n = NiceMatrix / rules ,
461     NiceArray .inherit:n =
462     {
463       NiceMatrix / Global ,
464       NiceMatrix / Env ,
465     } ,
466     NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
467     NiceArray / rules .inherit:n = NiceMatrix / rules ,
468     pNiceArray .inherit:n =
469     {
470       NiceMatrix / Global ,
471       NiceMatrix / Env ,
472     } ,
473     pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
474     pNiceArray / rules .inherit:n = NiceMatrix / rules ,
475   }

```

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

```

476 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
477 {
478   last-col .code:n = \tl_if_empty:nF { #1 }
479                 { \@@_error:n { last-col-non-empty-for-NiceMatrixOptions } }
480                 \int_zero:N \l_@@_last_col_int ,
481   small .bool_set:N = \l_@@_small_bool ,
482   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.

```

483   renew-matrix .code:n = \@@_renew_matrix: ,
484   renew-matrix .value_forbidden:n = true ,
485   transparent .meta:n = { renew-dots , renew-matrix } ,
486   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}`.

```

487   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.

```

488   columns-width .code:n =
489     \str_if_eq:nnTF { #1 } { auto }
490     { \@@_error:n { Option-auto-for-columns-width } }
491     { \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.

```

492   allow-duplicate-names .code:n =
493     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
494   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`).

```

495   letter-for-dotted-lines .code:n =
496   {
497     \int_compare:nTF { \tl_count:n { #1 } = 1 }
498     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
499     { \@@_error:n { Bad-value-for-letter-for-dotted-lines } }
500   } ,
501   letter-for-dotted-lines .value_required:n = true ,
502   unknown .code:n = \@@_error:n { Unknown-key-for-NiceMatrixOptions }
503 }

504 \str_new:N \l_@@_letter_for_dotted_lines_str
505 \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.

```

506 \NewDocumentCommand \NiceMatrixOptions { m }
507 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

508 \keys_define:nn { NiceMatrix / NiceMatrix }
509 {
510   last-col .code:n = \tl_if_empty:nTF {#1}
511   {
512     \bool_set_true:N \l_@@_last_col_without_value_bool
513     \int_set:Nn \l_@@_last_col_int { -1 }
514   }
515   { \int_set:Nn \l_@@_last_col_int { #1 } } ,
516   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl L ,
517   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl R ,
518   L .code:n = \tl_set:Nn \l_@@_type_of_col_tl L ,
519   R .code:n = \tl_set:Nn \l_@@_type_of_col_tl R ,
520   S .code:n = \bool_if:NTF \c_@@_siunitx_loaded_bool
521   { \tl_set:Nn \l_@@_type_of_col_tl S }
522   { \@@_error:n { option-S-without-siunitx } } ,
523   small .bool_set:N = \l_@@_small_bool ,
524   small .value_forbidden:n = true ,
525   unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
526 }

```

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

```

527 \keys_define:nn { NiceMatrix / NiceArray }
528 {

```

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.

```

529   small .bool_set:N = \l_@@_small_bool ,
530   small .value_forbidden:n = true ,
531   last-col .code:n = \tl_if_empty:nF { #1 }
532   { \@@_error:n { last-col-non-empty-for-NiceArray } }
533   \int_zero:N \l_@@_last_col_int ,
534   unknown .code:n = \@@_error:n { Unknown-option-for-NiceArray }
535 }

536 \keys_define:nn { NiceMatrix / pNiceArray }
537 {
538   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
539   last-col .code:n = \tl_if_empty:nF {#1}
540   { \@@_error:n { last-col-non-empty-for-NiceArray } }

```

```

541             \int_zero:N \l_@@_last_col_int ,
542     first-row .code:n = \int_zero:N \l_@@_first_row_int ,
543     small .bool_set:N = \l_@@_small_bool ,
544     small .value_forbidden:n = true ,
545     unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
546 }

```

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

```

547 \keys_define:nn { NiceMatrix / NiceTabular }
548 {
549     last-col .code:n = \tl_if_empty:nF {#1}
550             { \@@_error:n { last-col~non-empty~for~NiceArray } }
551             \int_zero:N \l_@@_last_col_int ,
552     unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
553 }

```

## 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}).

```

554 \cs_new_protected:Npn \@@_Cell:
555 {

```

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

```

556     \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.

```

557     \int_compare:nNnT \c@jCol = 1
558     { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
559     \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).

```

560     \hbox_set:Nw \l_@@_cell_box
561     \bool_if:NF \l_@@_NiceTabular_bool
562     {
563         \c_math_toggle_token
564         \bool_if:NT \l_@@_small_bool \scriptstyle
565     }

```

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.

```

566     \int_compare:nNnTF \c@iRow = 0
567     {
568         \int_compare:nNnT \c@jCol > 0
569         {
570             \l_@@_code_for_first_row_tl
571             \xglobal \colorlet { nicematrix-first-row } { . }
572         }
573     }
574     {
575         \int_compare:nNnT \c@iRow = \l_@@_last_row_int
576         {
577             \l_@@_code_for_last_row_tl

```



```

578         \xglobal \colorlet { nicematrix-last-row } { . }
579     }
580 }
581 }

```

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.

```

582 \cs_new_protected:Npn \@@_begin_of_row:
583 {
584     \int_gincr:N \c@iRow
585     \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
586     \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
587     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
588     \pgfpicture
589     \pgfrememberpicturepositiononpagetrue
590     \pgfcoordinate
591     { \@@_env: - row - \int_use:N \c@iRow - base }
592     { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
593     \str_if_empty:NF \l_@@_name_str
594     {
595         \pgfnodealias
596         { \l_@@_name_str - row - \int_use:N \c@iRow - base }
597         { \@@_env: - row - \int_use:N \c@iRow - base }
598     }
599     \endpgfpicture
600 }

```

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 will be dynamically added to this command.

```

601 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
602 {
603     \int_compare:nNnTF \c@iRow = 0
604     {
605         \dim_gset:Nn \g_@@_dp_row_zero_dim
606         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
607         \dim_gset:Nn \g_@@_ht_row_zero_dim
608         { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
609     }
610     {
611         \int_compare:nNnT \c@iRow = 1
612         {
613             \dim_gset:Nn \g_@@_ht_row_one_dim
614             { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
615         }
616     }
617 }
618 \cs_new_protected:Npn \@@_end_Cell:
619 {
620     \@@_math_toggle_token:
621     \hbox_set_end:
622     \box_set_ht:Nn \l_@@_cell_box
623     { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
624     \box_set_dp:Nn \l_@@_cell_box
625     { \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”).

```

626     \dim_gset:Nn \g_@@_max_cell_width_dim
627     { \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”.

```
628 \@@_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 use 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.

```
629 \bool_if:NTF \g_@@_empty_cell_bool
630 { \box_use_drop:N \l_@@_cell_box }
631 {
632   \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
633     \@@_node_for_the_cell:
634     { \box_use_drop:N \l_@@_cell_box }
635   }
636 \bool_gset_false:N \g_@@_empty_cell_bool
637 }
```

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

```
638 \cs_new_protected:Npn \@@_node_for_the_cell:
639 {
640   \pgfpicture
641   \pgfsetbaseline \c_zero_dim
642   \pgfrememberpicturepositiononpagetrue
643   \pgfset
644   {
645     inner~sep = \c_zero_dim ,
646     minimum~width = \c_zero_dim
647   }
648   \pgfnode
649   { rectangle }
650   { base }
651   { \box_use_drop:N \l_@@_cell_box }
652   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
653   { }
654   \str_if_empty:NF \l_@@_name_str
655   {
656     \pgfnodealias
657     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
658     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
659   }
660   \endpgfpicture
661 }
```

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}
```

```
662 \cs_new_protected:Npn \@@_instruction_of_type:nn #1 #2
663 {
```

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).

```
664 \tl_gput_right:cx
665 { \g_@@_ #1 _ lines _ tl }
666 {
667 \use:c { @@ _ draw _ #1 : nnn }
668 { \int_use:N \c@iRow }
669 { \int_use:N \c@jCol }
670 { \exp_not:n { #2 } }
671 }
672 }
```

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`.

```
673 \cs_new_protected:Npn \@@_array:
674 {
675 \bool_if:NTF \c_@@_revtex_bool
676 {
677 \cs_set_eq:NN \@acoll \@arrayacol
678 \cs_set_eq:NN \@acolr \@arrayacol
679 \cs_set_eq:NN \@acol \@arrayacol
680 \cs_set:Npn \@halignto { }
681 \@array@array
682 }
683 \array
```

`\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.

```
684 [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
685 }
```

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.

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

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

```
687 \cs_new_protected:Npn \@@_create_row_node:
688 {
```

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

```
689 \hbox
690 {
691 \bool_if:NT \l_@@_code_before_bool
692 {
693 \vtop
```

```

694         {
695             \skip_vertical:N 0.5\arrayrulewidth
696             \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
697             \skip_vertical:N -0.5\arrayrulewidth
698         }
699     }
700     \pgfpicture
701     \pgfrememberpicturerepositiononpagetrue
702     \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
703     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
704     \str_if_empty:NF \l_@@_name_str
705     {
706         \pgfnodealias
707         { \l_@@_name_str - row - \int_use:N \c@iRow }
708         { \@@_env: - row - \int_use:N \c@iRow }
709     }
710     \endpgfpicture
711 }
712 }

```

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

```

713 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
714 \cs_new_protected:Npn \@@_everycr_i:
715 {
716     \int_gzero:N \c@jCol
717     \bool_if:NF \g_@@_row_of_col_done_bool
718     {
719         \@@_create_row_node:

```

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

```

720     \bool_if:NT \l_@@_hlines_bool
721     {

```

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.

```

722         \int_compare:nNnT \c@iRow > { -1 }
723         {
724             \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@`.

```

725         { \hrule height \arrayrulewidth width \c_zero_dim }
726     }
727 }
728 }
729 }

```

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`, `W`, `p`, `m` and `b`).

```

730 \cs_set_protected:Npn \@@_newcolumnntype #1
731 {
732     \cs_if_free:cT { NC @ find @ #1 }
733     { \NC@list \expandafter { \the \NC@list \NC@do #1 } }
734     \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
735     \peek_meaning:NTF [
736     { \newcol@ #1 }
737     { \newcol@ #1 [ 0 ] }
738 }

```

The following command will be used to redefine the column types `p`, `m` and `b`. That means that it will be used three times. The first argument is the letter of the column type (`p`, `m` or `b`). The second is the letter of position for the environment `{minipage}` (`t`, `c` or `b`).

```
739 \cs_new_protected:Npn \@@_define_columntype:nn #1 #2
740 {
```

We don't want a warning for redefinition of the column type. That's why we use `\@@_newcolumntype` and not `\newcolumntype`.

```
741 \@@_newcolumntype #1 [ 1 ]
742 {
743   > {
744     \@@_Cell:
745     \begin { minipage } [ #2 ] { ##1 }
746     \mode_leave_vertical: \box_use:N \@arstrutbox
747   }
```

Here, we put `c` but we would have the result with `l` or `r`.

```
748   c
749   < { \box_use:N \@arstrutbox \end { minipage } \@@_end_Cell: }
750 }
751 }
```

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

```
752 \cs_new_protected:Npn \@@_pre_array:
753 {
```

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>31</sup>.

```
754 \bool_if:NT \c_@@_booktabs_loaded_bool
755 { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
756 \box_clear_new:N \l_@@_cell_box
757 \cs_if_exist:NT \theiRow
758 { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
759 \int_gzero_new:N \c@iRow
760 \cs_if_exist:NT \thejCol
761 { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
762 \int_gzero_new:N \c@jCol
763 \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}`).

```
764 \bool_if:NT \l_@@_small_bool
765 {
766   \cs_set:Npn \arraystretch { 0.47 }
767   \dim_set:Nn \arraycolsep { 1.45 pt }
768 }
```

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`.

```
769 \cs_set:Npn \ialign
770 {
771   \bool_if:NT \l_@@_NiceTabular_bool
```

---

<sup>31</sup>cf. `\nicematrix@redefine@check@rerun`

```

772     { \dim_set_eq:NN \arraycolsep \@@_old_arraycolsep_dim }
773     \bool_if:NTF \c_@@_colortbl_loaded_bool
774     {
775         \CT@everycr
776         {
777             \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
778             \@@_everycr:
779         }
780     }
781     { \everycr { \@@_everycr: } }
782     \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>32</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`.

```

783     \dim_gzero_new:N \g_@@_dp_row_zero_dim
784     \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
785     \dim_gzero_new:N \g_@@_ht_row_zero_dim
786     \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
787     \dim_gzero_new:N \g_@@_ht_row_one_dim
788     \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
789     \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
790     \dim_gzero_new:N \g_@@_ht_last_row_dim
791     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
792     \dim_gzero_new:N \g_@@_dp_last_row_dim
793     \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`.<sup>33</sup>

```

794     \cs_set_eq:NN \ialign \@@_old_ialign:
795     \halign
796     }

```

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).

```

797     \cs_set_eq:NN \@@_old_ldots \ldots
798     \cs_set_eq:NN \@@_old_cdots \cdots
799     \cs_set_eq:NN \@@_old_vdots \vdots
800     \cs_set_eq:NN \@@_old_ddots \ddots
801     \cs_set_eq:NN \@@_old_iddots \iddots
802     \cs_set_eq:NN \firsthline \hline
803     \cs_set_eq:NN \lasthline \hline
804     \bool_if:NTF \l_@@_standard_cline_bool
805     { \cs_set_eq:NN \cline \@@_standard_cline }
806     { \cs_set_eq:NN \cline \@@_cline }
807     \cs_set_eq:NN \Ldots \@@_Ldots
808     \cs_set_eq:NN \Cdots \@@_Cdots
809     \cs_set_eq:NN \Vdots \@@_Vdots
810     \cs_set_eq:NN \Ddots \@@_Ddots
811     \cs_set_eq:NN \Iddots \@@_Iddots
812     \cs_set_eq:NN \hdottedline \@@_hdottedline:
813     \cs_set_eq:NN \Hspace \@@_Hspace:
814     \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
815     \cs_set_eq:NN \Vdotsfor \@@_Vdotsfor:
816     \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
817     \cs_set_eq:NN \Block \@@_Block:

```

<sup>32</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

<sup>33</sup>The user will probably not use directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`).

```

818 \cs_set_eq:NN \rotate \@@_rotate:
819 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
820 \cs_set_eq:NN \dotfill \@@_dotfill:
821 \cs_set_eq:NN \CodeAfter \@@_CodeAfter:n
822 \cs_set_eq:NN \diagbox \@@_diagbox:nn
823 \bool_if:NT \l_@@_renew_dots_bool
824 {
825   \cs_set_eq:NN \ldots \@@_Ldots
826   \cs_set_eq:NN \cdots \@@_Cdots
827   \cs_set_eq:NN \vdots \@@_Vdots
828   \cs_set_eq:NN \ddots \@@_Ddots
829   \cs_set_eq:NN \iddots \@@_Iddots
830   \cs_set_eq:NN \dots \@@_Ldots
831   \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
832 }

```

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

```

833 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
834 \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).

```

835 \int_gset:Nn \c@iRow { \l_@@_first_row_int - 1 }

```

At the end of the environment `{array}`, `\c@iRow` will be the total number de rows.

`\g_@@_row_total_int` will be the number or rows excepted the last row (if `\l_@@_last_row_bool` has been raised with the option `last-row`).

```

836 \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.

```

837 \int_gzero_new:N \g_@@_col_total_int
838 \cs_set_eq:NN \@@_ifnextchar \new@@_ifnextchar

```

We define the new column types `L`, `C` and `R` that must be used instead of `l`, `c` and `r` in the preamble of `{NiceArray}`. We use `\@@_newcolumnntype` because it will be slightly quicker than `\newcolumnntype`.

```

839 \@@_newcolumnntype L { > \@@_Cell: l < \@@_end_Cell: }
840 \@@_newcolumnntype C { > \@@_Cell: c < \@@_end_Cell: }
841 \@@_newcolumnntype R { > \@@_Cell: r < \@@_end_Cell: }

```

We redefine the column types `p`, `m` and `b`. The command `\@@_define_columnntype:nn` is only used here.

```

842 \@@_define_columnntype:nn p t
843 \@@_define_columnntype:nn m c
844 \@@_define_columnntype:nn b b

```

We redefine the column types `w` and `W`. We use `\@@_newcolumnntype` instead of `\newcolumnntype` because we don’t want warnings for column types already defined.

```

845 \@@_newcolumnntype w [ 2 ]
846 {
847   > {
848     \hbox_set:Nw \l_@@_cell_box
849     \@@_Cell:
850   }
851   c
852   < {
853     \@@_end_Cell:
854     \hbox_set_end:

```

The `\str_lowercase:n` is only for giving the user the ability to write `wC{1cm}` instead of `wc{1cm}` for homogeneity with the letters L, C and R used elsewhere in the preamble instead of l, c and r.

```

855         \makebox [ ##2 ] [ \str_lowercase:n { ##1 } ]
856         { \box_use_drop:N \l_@@_cell_box }
857     }
858 }
859 \@@_newcolumnntype W [ 2 ]
860 {
861     > {
862         \hbox_set:Nw \l_@@_cell_box
863         \@@_Cell:
864     }
865     c
866     < {
867         \@@_end_Cell:
868         \hbox_set_end:
869         \cs_set_eq:NN \hss \hfil
870         \makebox [ ##2 ] [ \str_lowercase:n { ##1 } ]
871         { \box_use_drop:N \l_@@_cell_box }
872     }
873 }

```

By default, the letter used to specify a dotted line in the preamble of an environment of `nicematrix` (for example in `{pNiceArray}`) is the letter `:`. However, this letter is used by some packages, for example `arydshln`. That's why it's possible to change the letter used by `nicematrix` with the option `letter-for-dotted-lines` which changes the value of `\l_@@_letter_for_dotted_lines_str`. We rescan this string (which is always of length 1) in particular for the case where `pdflatex` is used with `french-babel` (the colon is activated by `french-babel` at the beginning of the document).

```

874 \tl_set_rescan:Nno
875 \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
876 \exp_args:NV \newcolumnntype \l_@@_letter_for_dotted_lines_str
877 {
878     !
879     {

```

The following code 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).

```

880 \int_compare:nNnF \c@iRow = 0
881 {
882     \int_compare:nNnF \c@iRow = \l_@@_last_row_int
883     { \skip_horizontal:N 2\l_@@_radius_dim }
884 }

```

Consider the following code:

```

\begin{NiceArray}{C:CC:C}
a & b
c & d \\
e & f & g & h \\
i & j & k & l
\end{NiceArray}

```

The first `:` in the preamble will be encountered during the first row of the environment `{NiceArray}` but the second one will be encountered only in the third row. We have to issue a command `\vdottedline:n` in the `code-after` only one time for each `:` in the preamble. That's why we keep a counter `\g_@@_last_vdotted_col_int` and with this counter, we know whether a letter `:` encountered during the parsing has already been taken into account in the `code-after`.

```

885 \int_compare:nNnT \c@jCol > \g_@@_last_vdotted_col_int
886 {
887     \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
888     \tl_gput_right:Nx \g_@@_internal_code_after_tl

```



The command `\@@_vdottedline:n` is protected, and, therefore, won't be expanded before writing on `\g_@@_internal_code_after_tl`.

```

889         { \@@_vdottedline:n { \int_use:N \c@jCol } }
890     }
891 }
892 }
893 \int_gzero_new:N \g_@@_last_vdotted_col_int
894 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:
895 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
896 \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.

```

897 \tl_gclear_new:N \g_@@_Cdots_lines_tl
898 \tl_gclear_new:N \g_@@_Ldots_lines_tl
899 \tl_gclear_new:N \g_@@_Vdots_lines_tl
900 \tl_gclear_new:N \g_@@_Ddots_lines_tl
901 \tl_gclear_new:N \g_@@_Iddots_lines_tl
902 \tl_gclear_new:N \g_@@_HVdotsfor_lines_tl
903 }

```

## The environment `{NiceArrayWithDelims}`

```

904 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
905 {
906   \tl_set:Nn \l_@@_left_delim_tl { #1 }
907   \tl_set:Nn \l_@@_right_delim_tl { #2 }
908   \bool_gset_false:N \g_@@_row_of_col_done_bool
909   \str_if_empty:NT \g_@@_name_env_str
910     { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
911   \@@_adapt_S_column:
912   \bool_if:NTF \l_@@_NiceTabular_bool
913     \mode_leave_vertical:
914     \@@_test_if_math_mode:
915   \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
916   \bool_set_true:N \l_@@_in_env_bool

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>34</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.

```

917   \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).

```

918   \cs_if_exist:NT \tikz@library@external@loaded
919   {
920     \tikzset { external / export = false }
921     \cs_if_exist:NT \ifstandalone
922       { \tikzset { external / optimize = false } }
923   }

```

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

```

924   \int_gincr:N \g_@@_env_int
925   \bool_if:NF \l_@@_block_auto_columns_width_bool
926   { \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.

```

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

```

---

<sup>34</sup>e.g. `\color[rgb]{0.5,0.5,0}`

The sequence `\g_@@_blocks_seq` will contain the characteristics of the blocks (specified by `\Block`) of the array. The sequence `\g_@@_pos_of_blocks_seq` will contain only the position of the blocks. Of course, this is redundant but it's for efficiency.

```

928 \seq_clear:N \g_@@_blocks_seq
929 \seq_clear:N \g_@@_pos_of_blocks_seq

```

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`.

```

930 \bool_if:NTF \l_@@_NiceArray_bool
931 { \keys_set:nn { NiceMatrix / NiceArray } }
932 { \keys_set:nn { NiceMatrix / pNiceArray } }
933 { #3 , #5 }

934 \tl_if_empty:NF \l_@@_rules_color_tl
935 { \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.

```

936 \bool_if:NT \l_@@_code_before_bool
937 {
938   \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
939   {

```

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).

```

940 \int_zero_new:N \c@iRow
941 \int_set:Nn \c@iRow
942 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
943 \int_zero_new:N \c@jCol
944 \int_set:Nn \c@jCol
945 { \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.

```

946 \int_compare:nNnF \l_@@_last_row_int = { -2 }
947 { \int_decr:N \c@iRow }
948 \int_compare:nNnF \l_@@_last_col_int = { -2 }
949 { \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.

```

950 \pgfsys@markposition { \@@_env: - position }
951 \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
952 \pgfpicture

```

First, the creation of the `row` nodes.

```

953 \int_step_inline:nnn
954 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
955 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
956 {
957   \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
958   \pgfcoordinate { \@@_env: - row - ##1 }
959   { \pgfpointdiff \@@_picture_position: \@@_node_position: }
960 }

```

Now, the creation of the `col` nodes.

```

961 \int_step_inline:nnn
962 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
963 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
964 {
965   \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
966   \pgfcoordinate { \@@_env: - col - ##1 }

```

```

967         { \pgfpointdiff \@@_picture_position: \@@_node_position: }
968     }
969     \endpgfpicture
970     \group_begin:
971         \bool_if:NT \c_@@_tikz_loaded_bool
972         {
973             \tikzset
974             {
975                 every~picture / .style =
976                 { overlay , name~prefix = \@@_env: - }
977             }
978         }
979         \cs_set_eq:NN \cellcolor \@@_cellcolor
980         \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
981         \cs_set_eq:NN \rowcolor \@@_rowcolor
982         \cs_set_eq:NN \rowcolors \@@_rowcolors
983         \cs_set_eq:NN \columncolor \@@_columncolor
984         \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.

```

985         \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
986         \l_@@_code_before_tl
987         \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
988     \group_end:
989 }
990 }

```

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).

```

991     \int_compare:nNnT \l_@@_last_row_int > { -2 }
992     {
993         \tl_put_right:Nn \@@_update_for_first_and_last_row:
994         {
995             \dim_gset:Nn \g_@@_ht_last_row_dim
996             { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
997             \dim_gset:Nn \g_@@_dp_last_row_dim
998             { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
999         }
1000     }
1001     \int_compare:nNnT \l_@@_last_row_int = { -1 }
1002     {
1003         \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.

```

1004     \str_if_empty:NTF \l_@@_name_str
1005     {
1006         \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1007         {
1008             \int_set:Nn \l_@@_last_row_int
1009             { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1010         }
1011     }
1012     {
1013         \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1014         {
1015             \int_set:Nn \l_@@_last_row_int
1016             { \use:c { @@_last_row_ \l_@@_name_str } }
1017         }
1018     }
1019 }

```

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).

```

1020 \int_compare:nNnT \l_@@_last_col_int = { -1 }
1021 {
1022   \str_if_empty:NTF \l_@@_name_str
1023   {
1024     \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1025     {
1026       \int_set:Nn \l_@@_last_col_int
1027       { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1028     }
1029   }
1030   {
1031     \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1032     {
1033       \int_set:Nn \l_@@_last_col_int
1034       { \use:c { @@_last_col_ \l_@@_name_str } }
1035     }
1036   }
1037 }

```

The code in \@@\_pre\_array: is used only by {NiceArrayWithDelims}.

```

1038 \@@_pre_array:

```

We compute the width of the two delimiters.

```

1039 \dim_zero_new:N \l_@@_left_delim_dim
1040 \dim_zero_new:N \l_@@_right_delim_dim
1041 \bool_if:NTF \l_@@_NiceArray_bool
1042 {
1043   \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1044   \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1045 }
1046 {

```

The command \bBigg@ is a command of amsmath.

```

1047 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
1048 \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1049 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
1050 \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1051 }

```

The array will be composed in a box (named \l\_@@\_the\_array\_box) because we have to do manipulations concerning the potential exterior rows.

```

1052 \box_clear_new:N \l_@@_the_array_box

```

We construct the preamble of the array in \l\_tmpa\_tl.

```

1053 \tl_set:Nn \l_tmpa_tl { #4 }
1054 \int_compare:nNnTF \l_@@_first_col_int = 0
1055 { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
1056 {
1057   \bool_lazy_all:nT
1058   {
1059     \l_@@_NiceArray_bool
1060     { \bool_not_p:n \l_@@_NiceTabular_bool }
1061     { \bool_not_p:n \l_@@_vlines_bool }
1062     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1063   }
1064   { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
1065 }
1066 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1067 { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
1068 {
1069   \bool_lazy_all:nT
1070   {
1071     \l_@@_NiceArray_bool

```

```

1072         { \bool_not_p:n \l_@@_NiceTabular_bool }
1073         { \bool_not_p:n \l_@@_vlines_bool }
1074         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1075     }
1076     { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
1077 }
1078 \tl_put_right:Nn \l_tmpa_tl { > { \@@_error_too_much_cols: } L }

```

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).

```

1079 \hbox_set:Nw \l_@@_the_array_box

```

Here is a trick. We will call `\array` and, at the beginning, `\array` will set `\col@sep` equal to the current value of `\arraycolsep`. In we are in an environment `{NiceTabular}`, we would like that `\array` sets `\col@sep` equal to the current value of `\tabcolsep`. That's why we set `\arraycolsep` equal to `\tabcolsep`. However, the value of `\tabcolsep` in each cell of the array should be equal to the current value of `\tabcolsep` outside `{NiceTabular}`. That's why we save the current value of `\arraycolsep` and we will restore the value just before the `\halign`. It's possible because we do a redefinition of `\ialign` (see just below).

```

1080 \bool_if:NT \l_@@_NiceTabular_bool
1081 {
1082     \dim_set_eq:NN \@@_old_arraycolsep_dim \arraycolsep
1083     \dim_set_eq:NN \arraycolsep \tabcolsep
1084 }

```

If the key `\vlines` is used, we increase `\arraycolsep` by `0.5\arrayrulewidth` in order to reserve space for the width of the vertical rules drawn with Tikz after the end of the array. However, the first `\arraycolsep` is used once (between columns, `\arraycolsep` is used twice). That's why we add a `0.5\arrayrulewidth` more.

```

1085 \bool_if:NT \l_@@_vlines_bool
1086 {
1087     \dim_add:Nn \arraycolsep { 0.5 \arrayrulewidth }
1088     \skip_horizontal:N 0.5\arrayrulewidth
1089 }
1090 \skip_horizontal:N \l_@@_left_margin_dim
1091 \skip_horizontal:N \l_@@_extra_left_margin_dim
1092 \c_math_toggle_token
1093 \bool_if:NTF \l_@@_light_syntax_bool
1094 { \use:c { @@-light-syntax } }
1095 { \use:c { @@-normal-syntax } }
1096 }
1097 {
1098     \bool_if:NTF \l_@@_light_syntax_bool
1099     { \use:c { end @@-light-syntax } }
1100     { \use:c { end @@-normal-syntax } }
1101     \c_math_toggle_token
1102     \skip_horizontal:N \l_@@_right_margin_dim
1103     \skip_horizontal:N \l_@@_extra_right_margin_dim

```

If the key `\vlines` is used, we have increased `\arraycolsep` by `0.5\arrayrulewidth` in order to reserve space for the width of the vertical rules drawn with Tikz after the end of the array. However, the last `\arraycolsep` is used once (between columns, `\arraycolsep` is used twice). That's we add a `0.5 \arrayrulewidth` more.

```

1104 \bool_if:NT \l_@@_vlines_bool { \skip_horizontal:N 0.5\arrayrulewidth }
1105 \hbox_set_end:

```

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

If the user has used the key `last-row` with a value, we control that the given value is correct (since we have just constructed the array, we know the real number of rows of the array).

```

1106 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1107 {
1108     \bool_if:NF \l_@@_last_row_without_value_bool

```

```

1109     {
1110         \int_compare:nNf \l_@@_last_row_int = \c@iRow
1111         {
1112             \@@_error:n { Wrong~last~row }
1113             \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1114         }
1115     }
1116 }

```

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>35</sup>

```

1117     \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1118     \bool_if:nT \g_@@_last_col_found_bool { \int_gdecr:N \c@jCol }

```

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

```

1119     \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1120     \int_compare:nNt \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. 70).

```

1121     \int_compare:nNt \l_@@_first_col_int = 0
1122     {
1123         \skip_horizontal:N \arraycolsep
1124         \skip_horizontal:N \g_@@_width_first_col_dim
1125     }

```

The construction of the real box is different in `{NiceArray}` and in the other environments because, in `{NiceArray}`, we have to take into account the value of `baseline` and we have no delimiter to put. We begin with `{NiceArray}`.

```

1126     \bool_if:nTF \l_@@_NiceArray_bool
1127     {

```

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`.

```

1128         \str_if_eq:VnTF \l_@@_baseline_str { b }
1129         {
1130             \pgfpicture
1131             \@@_qpoint:n { row - 1 }
1132             \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1133             \@@_qpoint:n { row - \int_use:N \c@iRow - base }
1134             \dim_gsub:Nn \g_tmpa_dim \pgf@y
1135             \endpgfpicture
1136             \int_compare:nNt \l_@@_first_row_int = 0
1137             {
1138                 \dim_gadd:Nn \g_tmpa_dim
1139                 { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
1140             }
1141             \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_@@_the_array_box }
1142         }
1143         {
1144             \str_if_eq:VnTF \l_@@_baseline_str { c }
1145             { \box_use_drop:N \l_@@_the_array_box }
1146             {

```

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

```

1147             \str_if_eq:VnT \l_@@_baseline_str { t }
1148             { \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`.

```

1149             \int_set:Nn \l_tmpa_int \l_@@_baseline_str
1150             \bool_if:nT
1151             {

```

---

<sup>35</sup>We remind that the potential “first column” (exterior) has the number 0.

```

1152         \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
1153     || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
1154 }
1155 {
1156     \@@_error:n { bad-value-for-baseline }
1157     \int_set:Nn \l_tmpa_int 1
1158 }
1159 \pgfpicture
1160 \@@_qpoint:n { row - 1 }
1161 \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1162 \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1163 \dim_gsub:Nn \g_tmpa_dim \pgf@y
1164 \endpgfpicture
1165 \int_compare:nNnT \l_@@_first_row_int = 0
1166 {
1167     \dim_gadd:Nn \g_tmpa_dim
1168         { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
1169 }
1170 \box_move_up:nn \g_tmpa_dim
1171 { \box_use_drop:N \l_@@_the_array_box }
1172 }
1173 }
1174 }

```

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).

```

1175 {
1176     \int_compare:nNnTF \l_@@_first_row_int = 0
1177     {
1178         \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1179         \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1180     }
1181     { \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>36</sup>

```

1182     \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1183     {
1184         \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1185         \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1186     }
1187     { \dim_zero:N \l_tmpb_dim }
1188     \hbox_set:Nn \l_tmpa_box
1189     {
1190         \c_math_toggle_token
1191         \left #1
1192         \vcenter
1193         {

```

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.

```

1194         \skip_vertical:N -\l_tmpa_dim
1195         \hbox
1196         {
1197             \bool_if:NTF \l_@@_NiceTabular_bool
1198             { \skip_horizontal:N -\tabcolsep }
1199             { \skip_horizontal:N -\arraycolsep }
1200             \box_use_drop:N \l_@@_the_array_box
1201             \bool_if:NTF \l_@@_NiceTabular_bool
1202             { \skip_horizontal:N -\tabcolsep }
1203             { \skip_horizontal:N -\arraycolsep }
1204         }

```

---

<sup>36</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).

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

```

1205         \skip_vertical:N -\l_tmpb_dim
1206     }
1207     \right #2
1208     \c_math_toggle_token
1209 }

```

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.

```

1210     \bool_if:NTF \l_@@_max_delimiter_width_bool
1211     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1212     \@@_put_box_in_flow:
1213 }

```

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. 71).

```

1214     \bool_if:NT \g_@@_last_col_found_bool
1215     {
1216         \skip_horizontal:N \g_@@_width_last_col_dim
1217         \skip_horizontal:N \arraycolsep
1218     }
1219     \@@_after_array:
1220 }

```

This is the end of the environment `{NiceArrayWithDelims}`.

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`).

```

1221 \cs_new_protected:Npn \@@_put_box_in_flow:
1222 {
1223     \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1224     \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1225     \str_if_eq:VnTF \l_@@_baseline_str { c }
1226     { \box_use_drop:N \l_tmpa_box }
1227     \@@_put_box_in_flow_i:
1228 }

```

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).

```

1229 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1230 {
1231     \str_case:VnF \l_@@_baseline_str
1232     {
1233         { t } { \int_set:Nn \l_tmpa_int 1 }
1234         { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1235     }
1236     { \int_set:Nn \l_tmpa_int \l_@@_baseline_str }
1237     \bool_if:nT
1238     {
1239         \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
1240         || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
1241     }
1242     {
1243         \@@_error:n { bad-value-for-baseline }
1244         \int_set:Nn \l_tmpa_int 1
1245     }
1246     \pgfpicture
1247     \@@_qpoint:n { row - 1 }
1248     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1249     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1250     \dim_gadd:Nn \g_tmpa_dim \pgf@y
1251     \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).

```
1252 \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1253 \dim_gsub:Nn \g_tmpa_dim \pgf@y
```

We take into account the position of the mathematical axis.

```
1254 \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
```

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

```
1255 \endpgfpicture
1256 \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1257 \box_use_drop:N \l_tmpa_box
1258 }
```

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.

```
1259 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1260 {
```

We will compute the real width of both delimiters used.

```
1261 \dim_zero_new:N \l_@@_real_left_delim_dim
1262 \dim_zero_new:N \l_@@_real_right_delim_dim
1263 \hbox_set:Nn \l_tmpb_box
1264 {
1265   \c_math_toggle_token
1266   \left #1
1267   \vcenter
1268   {
1269     \vbox_to_ht:nn
1270     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1271     { }
1272   }
1273   \right .
1274   \c_math_toggle_token
1275 }
1276 \dim_set:Nn \l_@@_real_left_delim_dim
1277 { \box_wd:N \l_tmpb_box - \nullldelimiterspace }
1278 \hbox_set:Nn \l_tmpb_box
1279 {
1280   \c_math_toggle_token
1281   \left .
1282   \vbox_to_ht:nn
1283   { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1284   { }
1285   \right #2
1286   \c_math_toggle_token
1287 }
1288 \dim_set:Nn \l_@@_real_right_delim_dim
1289 { \box_wd:N \l_tmpb_box - \nullldelimiterspace }
```

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

```
1290 \skip_horizontal:N \l_@@_left_delim_dim
1291 \skip_horizontal:N -\l_@@_real_left_delim_dim
1292 \@@_put_box_in_flow:
1293 \skip_horizontal:N \l_@@_right_delim_dim
1294 \skip_horizontal:N -\l_@@_real_right_delim_dim
1295 }
```

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 used 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).

```
1296 \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).

```

1297 {
1298   \peek_meaning_ignore_spaces:NTF \end
1299   { \@@_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`).

```

1300   { \exp_args:NV \@@_array: \l_tmpa_tl }
1301 }
1302 {
1303   \@@_create_col_nodes:
1304   \endarray
1305 }

```

When the key `light-syntax` is used, we use an environment which takes its whole body as an argument (with the specifier `b` of `xparse`).

```

1306 \NewDocumentEnvironment { @@-light-syntax } { b }
1307 {

```

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`.

```

1308   \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1309   \tl_map_inline:nn { #1 }
1310   {
1311     \tl_if_eq:nnT { ##1 } { & }
1312     { \@@_fatal:n { ampersand-in-light-syntax } }
1313     \tl_if_eq:nnT { ##1 } { \ }
1314     { \@@_fatal:n { double-backslash-in-light-syntax } }
1315   }

```

Now, you extract the `code-after` or 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_@@_code_after_tl`. That doesn't matter because `\CodeAfter` will be set to *no-op* before the execution of `\g_@@_code_after_tl`.

```

1316   \@@_light_syntax_i #1 \CodeAfter \q_stop
1317 }

```

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`.

```

1318 { }
1319 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1320 {
1321   \tl_gput_right:Nn \g_@@_code_after_tl { #2 }

```

The body of the array, which is stored in the argument `#1`, is now splitted into items (and *not* tokens).

```

1322   \seq_gclear_new:N \g_@@_rows_seq
1323   \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1324   \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.

```

1325   \int_compare:nNnT \l_@@_last_row_int = { -1 }
1326   { \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`).

```

1327   \exp_args:NV \@@_array: \l_tmpa_tl

```

We need a global affectation because, when executing `\l_tmpa_tl`, we will exit the first cell of the array.

```

1328 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1329 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1330 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1331 \@@_create_col_nodes:
1332 \endarray
1333 }
1334 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1335 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }
1336 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1337 {
1338 \seq_gclear_new:N \g_@@_cells_seq
1339 \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1340 \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1341 \l_tmpa_tl
1342 \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1343 }

```

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).

```

1344 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1345 {
1346 \str_if_eq:VnT \g_@@_name_env_str { #2 }
1347 { \@@_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.

```

1348 \end { #2 }
1349 }

```

The command `\@@_create_col_nodes:` will construct a special last row. That last row is a false row used to create the col nodes and to fix the width of the columns (when the array is constructed with an option which specify the width of the columns).

```

1350 \cs_new:Npn \@@_create_col_nodes:
1351 {
1352 \crrc
1353 \int_compare:nNnT \c@iRow = 0 { \@@_fatal:n { Zero~row } }
1354 \int_compare:nNnT \l_@@_first_col_int = 0
1355 {
1356 \omit
1357 \skip_horizontal:N -2\col@sep
1358 \bool_if:NT \l_@@_code_before_bool
1359 { \pgfsys@markposition { \@@_env: - col - 0 } }
1360 \pgfpicture
1361 \pgfrememberpicturepositiononpagetrue
1362 \pgfcoordinate { \@@_env: - col - 0 } \pgfpintorigin
1363 \str_if_empty:NF \l_@@_name_str
1364 { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1365 \endpgfpicture
1366 &
1367 }
1368 \omit

```

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

```

1369 \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`).

```

1370 \int_compare:nNnTF \l_@@_first_col_int = 0
1371 {
1372 \bool_if:NT \l_@@_code_before_bool
1373 {
1374 \hbox
1375 {

```

```

1376         \skip_horizontal:N -0.5\arrayrulewidth
1377         \pgfsys@markposition { \@@_env: - col - 1 }
1378         \skip_horizontal:N 0.5\arrayrulewidth
1379     }
1380 }
1381 \pgfpicture
1382 \pgfrememberpicturepositiononpagetrue
1383 \pgfcoordinate { \@@_env: - col - 1 }
1384 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1385 \str_if_empty:NF \l_@@_name_str
1386 { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1387 \endpgfpicture
1388 }
1389 {
1390     \bool_if:NT \l_@@_code_before_bool
1391     {
1392         \hbox
1393         {
1394             \skip_horizontal:N 0.5 \arrayrulewidth
1395             \pgfsys@markposition { \@@_env: - col - 1 }
1396             \skip_horizontal:N -0.5\arrayrulewidth
1397         }
1398     }
1399     \pgfpicture
1400     \pgfrememberpicturepositiononpagetrue
1401     \pgfcoordinate { \@@_env: - col - 1 }
1402     { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
1403     \str_if_empty:NF \l_@@_name_str
1404     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1405     \endpgfpicture
1406 }

```

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 erased by a fixed value in the concerned cases.

```

1407     \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1408     \bool_if:NF \l_@@_auto_columns_width_bool
1409     { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
1410     {
1411         \bool_lazy_and:nnTF
1412         \l_@@_auto_columns_width_bool
1413         { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1414         { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1415         { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1416         \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
1417     }
1418     \skip_horizontal:N \g_tmpa_skip
1419     \hbox
1420     {
1421         \bool_if:NT \l_@@_code_before_bool
1422         {
1423             \hbox
1424             {
1425                 \skip_horizontal:N -0.5\arrayrulewidth
1426                 \pgfsys@markposition { \@@_env: - col - 2 }
1427                 \skip_horizontal:N 0.5\arrayrulewidth
1428             }
1429         }
1430         \pgfpicture
1431         \pgfrememberpicturepositiononpagetrue
1432         \pgfcoordinate { \@@_env: - col - 2 }

```

```

1433     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1434     \str_if_empty:NF \l_@@_name_str
1435     { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
1436     \endpgfpicture
1437 }

```

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.

```

1438     \int_gset:Nn \g_tmpa_int 1
1439     \bool_if:NTF \g_@@_last_col_found_bool
1440     { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1441     { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1442     {
1443         &
1444         \omit

```

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

```

1445     \int_gincr:N \g_tmpa_int
1446     \skip_horizontal:N \g_tmpa_skip
1447     \bool_if:NT \l_@@_code_before_bool
1448     {
1449         \hbox
1450         {
1451             \skip_horizontal:N -0.5\arrayrulewidth
1452             \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1453             \skip_horizontal:N 0.5\arrayrulewidth
1454         }
1455     }

```

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

```

1456     \pgfpicture
1457     \pgfrememberpicturepositiononpagetrue
1458     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1459     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1460     \str_if_empty:NF \l_@@_name_str
1461     {
1462         \pgfnodealias
1463         { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
1464         { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1465     }
1466     \endpgfpicture
1467 }
1468 \bool_if:NT \g_@@_last_col_found_bool
1469 {
1470     \bool_if:NT \l_@@_code_before_bool
1471     {
1472         \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1473     }
1474     \skip_horizontal:N 2\col@sep
1475     \pgfpicture
1476     \pgfrememberpicturepositiononpagetrue
1477     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1478     \pgfpointorigin
1479     \str_if_empty:NF \l_@@_name_str
1480     {
1481         \pgfnodealias
1482         { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
1483         { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1484     }
1485     \endpgfpicture
1486     \skip_horizontal:N -2\col@sep
1487 }
1488 \cr
1489 }

```

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

```

1490 \tl_const:Nn \c_@@_preamble_first_col_tl
1491 {
1492   >
1493   {
1494     \@@_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.

```

1495     \hbox_set:Nw \l_@@_cell_box
1496     \@@_math_toggle_token:
1497     \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”.

```

1498     \bool_lazy_and:nnT
1499     { \int_compare_p:nNn \c@iRow > 0 }
1500     {
1501       \bool_lazy_or_p:nn
1502       { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1503       { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1504     }
1505     {
1506       \l_@@_code_for_first_col_tl
1507       \xglobal \colorlet { nicematrix-first-col } { . }
1508     }
1509   }

```

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`.

```

1510   l
1511   <
1512   {
1513     \@@_math_toggle_token:
1514     \hbox_set_end:
1515     \@@_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.

```

1516     \dim_gset:Nn \g_@@_width_first_col_dim
1517     { \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.

```

1518     \hbox_overlap_left:n
1519     {
1520       \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1521       \@@_node_for_the_cell:
1522       { \box_use_drop:N \l_@@_cell_box }
1523       \skip_horizontal:N \l_@@_left_delim_dim
1524       \skip_horizontal:N \l_@@_left_margin_dim
1525       \skip_horizontal:N \l_@@_extra_left_margin_dim
1526     }
1527     \skip_horizontal:N -2\col@sep
1528   }
1529 }

```

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

```

1530 \tl_const:Nn \c_@@_preamble_last_col_tl
1531 {
1532   >
1533   {

```

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

```

1534     \bool_gset_true:N \g_@@_last_col_found_bool
1535     \int_gincr:N \c@jCol
1536     \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.

```

1537     \hbox_set:Nw \l_@@_cell_box
1538     \@@_math_toggle_token:
1539     \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”.

```

1540     \int_compare:nNnT \c@iRow > 0
1541     {
1542         \bool_lazy_or:nnT
1543         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1544         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1545         {
1546             \l_@@_code_for_last_col_tl
1547             \xglobal \colorlet { nicematrix-last-col } { . }
1548         }
1549     }
1550 }
1551 l
1552 <
1553 {
1554     \@@_math_toggle_token:
1555     \hbox_set_end:
1556     \@@_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.

```

1557     \dim_gset:Nn \g_@@_width_last_col_dim
1558     { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
1559     \skip_horizontal:N -2\col@sep

```

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

```

1560     \hbox_overlap_right:n
1561     {
1562         \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1563         {
1564             \skip_horizontal:N \l_@@_right_delim_dim
1565             \skip_horizontal:N \l_@@_right_margin_dim
1566             \skip_horizontal:N \l_@@_extra_right_margin_dim
1567             \@@_node_for_the_cell:
1568         }
1569     }
1570 }
1571 }

```

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.

```

1572 \NewDocumentEnvironment { NiceArray } { }
1573 {
1574     \bool_set_true:N \l_@@_NiceArray_bool
1575     \str_if_empty:NT \g_@@_name_env_str
1576     { \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).

```

1577     \NiceArrayWithDelims . .
1578 }
1579 { \endNiceArrayWithDelims }

```

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

```

1580 \NewDocumentEnvironment { pNiceArray } { }

```

```

1581 {
1582   \str_if_empty:NT \g_@@_name_env_str
1583   { \str_gset:Nn \g_@@_name_env_str { pNiceArray } }
1584   \@@_test_if_math_mode:
1585   \NiceArrayWithDelims ( )
1586 }
1587 { \endNiceArrayWithDelims }

1588 \NewDocumentEnvironment { bNiceArray } { }
1589 {
1590   \str_if_empty:NT \g_@@_name_env_str
1591   { \str_gset:Nn \g_@@_name_env_str { bNiceArray } }
1592   \@@_test_if_math_mode:
1593   \NiceArrayWithDelims [ ]
1594 }
1595 { \endNiceArrayWithDelims }

1596 \NewDocumentEnvironment { BNiceArray } { }
1597 {
1598   \str_if_empty:NT \g_@@_name_env_str
1599   { \str_gset:Nn \g_@@_name_env_str { BNiceArray } }
1600   \@@_test_if_math_mode:
1601   \NiceArrayWithDelims \{ \}
1602 }
1603 { \endNiceArrayWithDelims }

1604 \NewDocumentEnvironment { vNiceArray } { }
1605 {
1606   \str_if_empty:NT \g_@@_name_env_str
1607   { \str_gset:Nn \g_@@_name_env_str { vNiceArray } }
1608   \@@_test_if_math_mode:
1609   \NiceArrayWithDelims | |
1610 }
1611 { \endNiceArrayWithDelims }

1612 \NewDocumentEnvironment { VNiceArray } { }
1613 {
1614   \str_if_empty:NT \g_@@_name_env_str
1615   { \str_gset:Nn \g_@@_name_env_str { VNiceArray } }
1616   \@@_test_if_math_mode:
1617   \NiceArrayWithDelims \| \|
1618 }
1619 { \endNiceArrayWithDelims }

```

## The environment {NiceMatrix} and its variants

```

1620 \cs_new_protected:Npn \@@_define_env:n #1
1621 {
1622   \NewDocumentEnvironment { #1 NiceMatrix } { ! O { } }
1623   {
1624     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
1625     \tl_set:Nn \l_@@_type_of_col_tl C
1626     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1627     \exp_args:Nnx \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
1628   }
1629   { \use:c { end #1 NiceArray } }
1630 }

1631 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
1632 {
1633   \use:c { #1 NiceArray }
1634   {
1635     *
1636     {
1637       \int_compare:nNnTF \l_@@_last_col_int < 0
1638       \c@MaxMatrixCols

```



```

1639         { \@@_pred:n \l_@@_last_col_int }
1640     }
1641     #2
1642 }
1643 }
1644 \@@_define_env:n { }
1645 \@@_define_env:n p
1646 \@@_define_env:n b
1647 \@@_define_env:n B
1648 \@@_define_env:n v
1649 \@@_define_env:n V

```

## The environment {NiceTabular}

```

1650 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
1651 {
1652     \str_gset:Nn \g_@@_name_env_str { NiceTabular }
1653     \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
1654     \bool_set_true:N \l_@@_NiceTabular_bool
1655     \NiceArray { #2 }
1656 }
1657 { \endNiceArray }

```

## After the construction of the array

```

1658 \cs_new_protected:Npn \@@_after_array:
1659 {
1660     \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.

```

1661     \bool_if:NT \g_@@_last_col_found_bool
1662     { \int_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int }

```

If we are in an environment without preamble (like `{NiceMatrix}` or `{pNiceMatrix}`) and if the option `last-col` has been used without value we fix the real value of `\l_@@_last_col_int`.

```

1663     \bool_if:NT \l_@@_last_col_without_value_bool
1664     {
1665         \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
1666         \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1667         \iow_shipout:Nx \@mainaux
1668         {
1669             \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
1670             { \int_use:N \g_@@_col_total_int }
1671         }
1672         \str_if_empty:NF \l_@@_name_str
1673         {
1674             \iow_shipout:Nx \@mainaux
1675             {
1676                 \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
1677                 { \int_use:N \g_@@_col_total_int }
1678             }
1679         }
1680         \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1681     }

```

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.

```

1682     \bool_if:NT \l_@@_last_row_without_value_bool

```

```

1683 {
1684   \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.

```

1685   \bool_if:NF \l_@@_light_syntax_bool
1686   {
1687     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1688     \iow_shipout:Nx \@mainaux
1689     {
1690       \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1691       { \int_use:N \g_@@_row_total_int }
1692     }

```

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.

```

1693     \str_if_empty:NF \l_@@_name_str
1694     {
1695       \iow_shipout:Nx \@mainaux
1696       {
1697         \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1698         { \int_use:N \g_@@_row_total_int }
1699       }
1700     }
1701     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1702   }
1703 }

```

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

```

1704   \bool_if:NT \l_@@_code_before_bool
1705   {
1706     \iow_now:Nn \@mainaux \ExplSyntaxOn
1707     \iow_now:Nx \@mainaux
1708     { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
1709     \iow_now:Nx \@mainaux
1710     {
1711       \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
1712       {
1713         \int_use:N \l_@@_first_row_int ,
1714         \int_use:N \g_@@_row_total_int ,
1715         \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`.

```

1716       \bool_lazy_and:nnTF
1717       { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
1718       { \bool_not_p:n \g_@@_last_col_found_bool }
1719       \@@_succ:n
1720       \int_use:N
1721       \g_@@_col_total_int
1722     }
1723   }
1724   \iow_now:Nn \@mainaux \ExplSyntaxOff
1725 }

```

By default, the diagonal lines will be parallelized<sup>37</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.

```

1726   \bool_if:NT \l_@@_parallelize_diags_bool
1727   {
1728     \int_gzero_new:N \g_@@_ddots_int
1729     \int_gzero_new:N \g_@@_iddots_int

```

<sup>37</sup>It's possible to use the option `parallelize-diags` to disable this parallelization.

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

```

1730     \dim_gzero_new:N \g_@@_delta_x_one_dim
1731     \dim_gzero_new:N \g_@@_delta_y_one_dim
1732     \dim_gzero_new:N \g_@@_delta_x_two_dim
1733     \dim_gzero_new:N \g_@@_delta_y_two_dim
1734   }
1735   \bool_if:NTF \l_@@_medium_nodes_bool
1736   {
1737     \bool_if:NTF \l_@@_large_nodes_bool
1738     \@@_create_medium_and_large_nodes:
1739     \@@_create_medium_nodes:
1740   }
1741   { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
1742   \int_zero_new:N \l_@@_initial_i_int
1743   \int_zero_new:N \l_@@_initial_j_int
1744   \int_zero_new:N \l_@@_final_i_int
1745   \int_zero_new:N \l_@@_final_j_int
1746   \bool_set_false:N \l_@@_initial_open_bool
1747   \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.

```

1748   \bool_if:NT \l_@@_small_bool
1749   {
1750     \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1751     \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.

```

1752     \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
1753   }

```

Now, we actually draw the dotted lines.

```

1754   \@@_draw_dotted_lines:
1755   \bool_if:NTF \l_@@_hvlines_bool
1756   \@@_draw_hvlines:
1757   {
1758     \bool_if:NT \l_@@_hlines_bool \@@_draw_hlines:
1759     \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
1760   }

```

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

```

1761   \cs_set_eq:NN \ialign \@@_old_ialign:
1762   \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
1763   \g_@@_internal_code_after_tl
1764   \tl_gclear:N \g_@@_internal_code_after_tl
1765   \bool_if:NT \c_@@_tikz_loaded_bool
1766   {
1767     \tikzset
1768     {
1769       every~picture / .style =
1770       {
1771         overlay ,
1772         remember~picture ,
1773         name~prefix = \@@_env: -
1774       }
1775     }
1776   }
1777   \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 one is eventually present in `\g_@@_code_after_tl`. That's why we set `\Code-after` to be *no-op* now.

```
1778 \cs_set_eq:NN \CodeAfter \prg_do_nothing:
```

And here's the `code-after`:

```
1779 \g_@@_code_after_tl
1780 \tl_gclear:N \g_@@_code_after_tl
1781 \group_end:
1782 \str_gclear:N \g_@@_name_env_str
1783 \@@_restore_iRow_jCol:
```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>38</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`.

```
1784 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
1785 }
```

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).

```
1786 \AtBeginDocument
1787 {
1788   \cs_new_protected:Npx \@@_draw_dotted_lines:
1789   {
1790     \c_@@_pgfortikzpicture_tl
1791     \@@_draw_dotted_lines_i:
1792     \c_@@_endpgfortikzpicture_tl
1793   }
1794 }
```

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

```
1795 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
1796 {
1797   \pgfrememberpicturerepositiononpagetrue
1798   \pgf@relevantforpicturesizefalse
1799   \g_@@_HVdotsfor_lines_tl
1800   \g_@@_Vdots_lines_tl
1801   \g_@@_Ddots_lines_tl
1802   \g_@@_Iddots_lines_tl
1803   \g_@@_Cdots_lines_tl
1804   \g_@@_Ldots_lines_tl
1805 }

1806 \cs_new_protected:Npn \@@_restore_iRow_jCol:
1807 {
1808   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
1809   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
1810 }
```

## 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:

---

<sup>38</sup>e.g. `\color[rgb]{0.5,0.5,0}`)

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

This command computes:

- `\l_@@_initial_i_int` and `\l_@@_initial_j_int` which are the coordinates of one extremity of the line;
- `\l_@@_final_i_int` and `\l_@@_final_j_int` which are the coordinates of the other extremity of the line;
- `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` to indicate whether the extremities are open or not.

```
1811 \cs_new_protected:Npn \l_@@_find_extremities_of_line:nnnn #1 #2 #3 #4
1812 {
```

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

```
1813 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
1814 \int_set:Nn \l_@@_initial_i_int { #1 }
1815 \int_set:Nn \l_@@_initial_j_int { #2 }
1816 \int_set:Nn \l_@@_final_i_int { #1 }
1817 \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.

```
1818 \bool_set_false:N \l_@@_stop_loop_bool
1819 \bool_do_until:Nn \l_@@_stop_loop_bool
1820 {
1821 \int_add:Nn \l_@@_final_i_int { #3 }
1822 \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1823 \bool_set_false:N \l_@@_final_open_bool
1824 \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
1825 {
1826 \int_compare:nNnTF { #3 } = 1
1827 { \bool_set_true:N \l_@@_final_open_bool }
1828 {
1829 \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1830 { \bool_set_true:N \l_@@_final_open_bool }
1831 }
1832 }
1833 {
1834 \int_compare:nNnTF \l_@@_final_j_int < 1
1835 {
1836 \int_compare:nNnT { #4 } = { -1 }
1837 { \bool_set_true:N \l_@@_final_open_bool }
1838 }
1839 {
1840 \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1841 {
1842 \int_compare:nNnT { #4 } = 1
1843 { \bool_set_true:N \l_@@_final_open_bool }
1844 }
1845 }
1846 }
1847 \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.

```

1848     {
We do a step backwards.
1849         \int_sub:Nn \l_@@_final_i_int { #3 }
1850         \int_sub:Nn \l_@@_final_j_int { #4 }
1851         \bool_set_true:N \l_@@_stop_loop_bool
1852     }

```

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`.

```

1853     {
1854         \cs_if_exist:cTF
1855         {
1856             @@ _ dotted _
1857             \int_use:N \l_@@_final_i_int -
1858             \int_use:N \l_@@_final_j_int
1859         }
1860         {
1861             \int_sub:Nn \l_@@_final_i_int { #3 }
1862             \int_sub:Nn \l_@@_final_j_int { #4 }
1863             \bool_set_true:N \l_@@_final_open_bool
1864             \bool_set_true:N \l_@@_stop_loop_bool
1865         }
1866         {
1867             \cs_if_exist:cTF
1868             {
1869                 pgf @ sh @ ns @ \@@_env:
1870                 - \int_use:N \l_@@_final_i_int
1871                 - \int_use:N \l_@@_final_j_int
1872             }
1873             { \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 mark 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.

```

1874     {
1875         \cs_set:cpn
1876         {
1877             @@ _ dotted _
1878             \int_use:N \l_@@_final_i_int -
1879             \int_use:N \l_@@_final_j_int
1880         }
1881         { }
1882     }
1883 }
1884 }
1885 }

```

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

```

1886     \bool_set_false:N \l_@@_stop_loop_bool
1887     \bool_do_until:Nn \l_@@_stop_loop_bool
1888     {
1889         \int_sub:Nn \l_@@_initial_i_int { #3 }
1890         \int_sub:Nn \l_@@_initial_j_int { #4 }
1891         \bool_set_false:N \l_@@_initial_open_bool
1892         \int_compare:nNnTF \l_@@_initial_i_int < 1
1893         {
1894             \int_compare:nNnTF { #3 } = 1

```

```

1895         { \bool_set_true:N \l_@@_initial_open_bool }
1896     {
1897         \int_compare:nNnT \l_@@_initial_j_int = 0
1898         { \bool_set_true:N \l_@@_initial_open_bool }
1899     }
1900 }
1901 {
1902     \int_compare:nNnTF \l_@@_initial_j_int < 1
1903     {
1904         \int_compare:nNnT { #4 } = 1
1905         { \bool_set_true:N \l_@@_initial_open_bool }
1906     }
1907     {
1908         \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
1909         {
1910             \int_compare:nNnT { #4 } = { -1 }
1911             { \bool_set_true:N \l_@@_initial_open_bool }
1912         }
1913     }
1914 }
1915 \bool_if:NNTF \l_@@_initial_open_bool
1916 {
1917     \int_add:Nn \l_@@_initial_i_int { #3 }
1918     \int_add:Nn \l_@@_initial_j_int { #4 }
1919     \bool_set_true:N \l_@@_stop_loop_bool
1920 }
1921 {
1922     \cs_if_exist:cTF
1923     {
1924         @@ _ dotted _
1925         \int_use:N \l_@@_initial_i_int -
1926         \int_use:N \l_@@_initial_j_int
1927     }
1928     {
1929         \int_add:Nn \l_@@_initial_i_int { #3 }
1930         \int_add:Nn \l_@@_initial_j_int { #4 }
1931         \bool_set_true:N \l_@@_initial_open_bool
1932         \bool_set_true:N \l_@@_stop_loop_bool
1933     }
1934     {
1935         \cs_if_exist:cTF
1936         {
1937             pgf @ sh @ ns @ \@@_env:
1938             - \int_use:N \l_@@_initial_i_int
1939             - \int_use:N \l_@@_initial_j_int
1940         }
1941         { \bool_set_true:N \l_@@_stop_loop_bool }
1942         {
1943             \cs_set:cpn
1944             {
1945                 @@ _ dotted _
1946                 \int_use:N \l_@@_initial_i_int -
1947                 \int_use:N \l_@@_initial_j_int
1948             }
1949             { }
1950         }
1951     }
1952 }
1953 }

```

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.

```

1954     \bool_if:NT \l_@@_hvlines_bool
1955     {

```

```

1956     \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
1957     {
1958         { \int_use:N \l_@@_initial_i_int }
1959         { \int_use:N \l_@@_initial_j_int }
1960         { \int_use:N \l_@@_final_i_int }
1961         { \int_use:N \l_@@_final_j_int }
1962     }
1963 }
1964 }

1965 \cs_new_protected:Npn \@@_set_initial_coords:
1966 {
1967     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1968     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1969 }
1970 \cs_new_protected:Npn \@@_set_final_coords:
1971 {
1972     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1973     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1974 }
1975 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
1976 {
1977     \pgfpointanchor
1978     {
1979         \@@_env:
1980         - \int_use:N \l_@@_initial_i_int
1981         - \int_use:N \l_@@_initial_j_int
1982     }
1983     { #1 }
1984     \@@_set_initial_coords:
1985 }
1986 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
1987 {
1988     \pgfpointanchor
1989     {
1990         \@@_env:
1991         - \int_use:N \l_@@_final_i_int
1992         - \int_use:N \l_@@_final_j_int
1993     }
1994     { #1 }
1995     \@@_set_final_coords:
1996 }

```

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.

```

1997 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
1998 {
1999     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2000     {
2001         \@@_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.

```

2002     \group_begin:
2003     \int_compare:nNnTF { #1 } = 0
2004     { \color { nicematrix-first-row } }
2005     {

```

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.

```

2006         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2007         { \color { nicematrix-last-row } }

```



```

2008         }
2009         \keys_set:nn { NiceMatrix / xdots } { #3 }
2010         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2011         \@@_actually_draw_Ldots:
2012     \group_end:
2013 }
2014 }

```

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`.

```

2015 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2016 {
2017     \bool_if:NTF \l_@@_initial_open_bool
2018     {
2019         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2020         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2021         \dim_add:Nn \l_@@_x_initial_dim
2022             { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
2023         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2024         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2025     }
2026     { \@@_set_initial_coords_from_anchor:n { base-east } }
2027     \bool_if:NTF \l_@@_final_open_bool
2028     {
2029         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2030         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2031         \dim_sub:Nn \l_@@_x_final_dim
2032             { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
2033         \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2034         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2035     }
2036     { \@@_set_final_coords_from_anchor:n { base-west } }

```

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

```

2037     \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2038     \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2039     \@@_draw_line:
2040 }

```

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.

```

2041 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2042 {
2043     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2044     {
2045         \@@_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.

```

2046     \group_begin:
2047     \int_compare:nNnTF { #1 } = 0
2048     { \color { nicematrix-first-row } }
2049     {

```

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.

```

2050     \int_compare:nNnT { #1 } = \l_@@_last_row_int
2051     { \color { nicematrix-last-row } }
2052     }
2053     \keys_set:nn { NiceMatrix / xdots } { #3 }
2054     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2055     \@@_actually_draw_Cdots:
2056     \group_end:
2057 }
2058 }

```

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`.

```

2059 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2060 {
2061     \bool_if:NTF \l_@@_initial_open_bool
2062     {
2063         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2064         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2065         \dim_add:Nn \l_@@_x_initial_dim
2066         { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
2067     }
2068     { \@@_set_initial_coords_from_anchor:n { mid-east } }
2069     \bool_if:NTF \l_@@_final_open_bool
2070     {
2071         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2072         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2073         \dim_sub:Nn \l_@@_x_final_dim
2074         { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
2075     }
2076     { \@@_set_final_coords_from_anchor:n { mid-west } }
2077     \bool_lazy_and:nnTF
2078     \l_@@_initial_open_bool
2079     \l_@@_final_open_bool
2080     {
2081         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2082         \dim_set_eq:NN \l_tmpa_dim \pgf@y
2083         \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2084         \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2085         \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2086     }
2087     {
2088         \bool_if:NT \l_@@_initial_open_bool
2089         { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2090         \bool_if:NT \l_@@_final_open_bool

```

```

2091         { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2092     }
2093     \@@_draw_line:
2094 }

```

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.

```

2095 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2096 {
2097     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2098     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2099     {
2100         \@@_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.

```

2101     \group_begin:
2102     \int_compare:nNnTF { #2 } = 0
2103     { \color { nicematrix-first-col } }
2104     {
2105         \int_compare:nNnT { #2 } = \l_@@_last_col_int
2106         { \color { nicematrix-last-col } }
2107     }
2108     \keys_set:nn { NiceMatrix / xdots } { #3 }
2109     \@@_actually_draw_Vdots:
2110 \group_end:
2111 }
2112 }

```

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`.

```

2113 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2114 {

```

The boolean `\l_tmpa_bool` indicates whether the column is of type `l` (L of `{NiceArray}`) or may be considered as if.

```

2115     \bool_set_false:N \l_tmpa_bool
2116     \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2117     {
2118         \@@_set_initial_coords_from_anchor:n { south-west }
2119         \@@_set_final_coords_from_anchor:n { north-west }
2120         \bool_set:Nn \l_tmpa_bool
2121         { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2122     }

```

Now, we try to determine whether the column is of type `c` (C of `{NiceArray}`) or may be considered as if.

```

2123     \bool_if:NTF \l_@@_initial_open_bool
2124     {
2125         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2126         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2127     }

```

```

2128     { \@@_set_initial_coords_from_anchor:n { south } }
2129 \bool_if:NTF \l_@@_final_open_bool
2130 {
2131     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2132     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2133 }
2134 { \@@_set_final_coords_from_anchor:n { north } }
2135 \bool_if:NTF \l_@@_initial_open_bool
2136 {
2137     \bool_if:NTF \l_@@_final_open_bool
2138     {
2139         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2140         \dim_set_eq:NN \l_tmpa_dim \pgf@x
2141         \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2142         \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2143         \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.

```

2144     \int_compare:nNnT \l_@@_last_col_int > { -2 }
2145     {
2146         \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2147         {
2148             \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2149             \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2150             \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2151             \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2152         }
2153     }
2154 }
2155 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2156 }
2157 {
2158     \bool_if:NTF \l_@@_final_open_bool
2159     { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2160 }

```

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.

```

2161     \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2162     {
2163         \dim_set:Nn \l_@@_x_initial_dim
2164         {
2165             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2166             \l_@@_x_initial_dim \l_@@_x_final_dim
2167         }
2168         \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2169     }
2170 }
2171 }
2172 \@@_draw_line:
2173 }

```

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.

```

2174 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2175 {
2176     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }

```

```

2177 {
2178   \l_@@_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.

```

2179   \group_begin:
2180     \keys_set:nn { NiceMatrix / xdots } { #3 }
2181     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2182     \l_@@_actually_draw_Ddots:
2183   \group_end:
2184 }
2185 }

```

The command `\l_@@_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.`

```

2186 \cs_new_protected:Npn \l_@@_actually_draw_Ddots:
2187 {
2188   \bool_if:NTF \l_@@_initial_open_bool
2189   {
2190     \l_@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2191     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2192     \l_@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2193     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2194   }
2195   { \l_@@_set_initial_coords_from_anchor:n { south-east } }
2196   \bool_if:NTF \l_@@_final_open_bool
2197   {
2198     \l_@@_qpoint:n { row - \l_@@_succ:n \l_@@_final_i_int }
2199     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2200     \l_@@_qpoint:n { col - \l_@@_succ:n \l_@@_final_j_int }
2201     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2202   }
2203   { \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.

```

2204   \bool_if:NT \l_@@_parallelize_diags_bool
2205   {
2206     \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).

```

2207     \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.

```

2208   {
2209     \dim_gset:Nn \g_@@_delta_x_one_dim
2210     { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2211     \dim_gset:Nn \g_@@_delta_y_one_dim
2212     { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2213   }

```

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`.

```

2214     {
2215         \dim_set:Nn \l_@@_y_final_dim
2216         {
2217             \l_@@_y_initial_dim +
2218             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2219             \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
2220         }
2221     }
2222 }
2223 \@@_draw_line:
2224 }

```

We draw the `\Iddots` diagonals in the same way.

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

2225 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2226 {
2227     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2228     {
2229         \@@_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.

```

2230         \group_begin:
2231         \keys_set:nn { NiceMatrix / xdots } { #3 }
2232         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2233         \@@_actually_draw_Iddots:
2234         \group_end:
2235     }
2236 }

```

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`.

```

2237 \cs_new_protected:Npn \@@_actually_draw_Iddots:
2238 {
2239     \bool_if:NTF \l_@@_initial_open_bool
2240     {
2241         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2242         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2243         \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2244         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2245     }
2246     { \@@_set_initial_coords_from_anchor:n { south-west } }
2247     \bool_if:NTF \l_@@_final_open_bool
2248     {
2249         \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2250         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2251         \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2252         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2253     }

```

```

2254     { \@@_set_final_coords_from_anchor:n { north-east } }
2255 \bool_if:NT \l_@@_parallelize_diags_bool
2256 {
2257     \int_gincr:N \g_@@_iddots_int
2258     \int_compare:nNnTF \g_@@_iddots_int = 1
2259     {
2260         \dim_gset:Nn \g_@@_delta_x_two_dim
2261             { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2262         \dim_gset:Nn \g_@@_delta_y_two_dim
2263             { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2264     }
2265     {
2266         \dim_set:Nn \l_@@_y_final_dim
2267             {
2268                 \l_@@_y_initial_dim +
2269                 ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2270                 \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2271             }
2272     }
2273 }
2274 \@@_draw_line:
2275 }

```

## 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`

```

2276 \cs_new_protected:Npn \@@_draw_line:
2277 {
2278     \pgfrememberpicturepositiononpagetrue
2279     \pgf@relevantforpicturesizefalse
2280     \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
2281         \@@_draw_standard_dotted_line:
2282         \@@_draw_non_standard_dotted_line:
2283 }

```

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.

```

2284 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2285 {
2286     \begin { scope }
2287     \exp_args:No \@@_draw_non_standard_dotted_line:n
2288         { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2289 }

```

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.

```

2290 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2291 {

```

```

2292 \draw
2293 [
2294     #1 ,
2295     shorten~> = \l_@@_xdots_shorten_dim ,
2296     shorten~< = \l_@@_xdots_shorten_dim ,
2297 ]
2298     ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
2299     -- node [ sloped , above ]
2300         { \c_math_toggle_token \scriptstyle \l_@@_xdots_up_tl \c_math_toggle_token }
2301     node [ sloped , below ]
2302         {
2303             \c_math_toggle_token
2304             \scriptstyle \l_@@_xdots_down_tl
2305             \c_math_toggle_token
2306         }
2307     ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2308 \end { scope }
2309 }

```

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

```

2310 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2311 {

```

First, we put the labels.

```

2312 \bool_lazy_and:nnF
2313 { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2314 { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2315 {
2316     \pgfscope
2317     \pgftransformshift
2318     {
2319         \pgfpointlineattime { 0.5 }
2320         { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2321         { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2322     }
2323     \pgftransformrotate
2324     {
2325         \fp_eval:n
2326         {
2327             atand
2328             (
2329                 \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2330                 \l_@@_x_final_dim - \l_@@_x_initial_dim
2331             )
2332         }
2333     }
2334     \pgfnode
2335     { rectangle }
2336     { south }
2337     {
2338         \c_math_toggle_token
2339         \scriptstyle \l_@@_xdots_up_tl
2340         \c_math_toggle_token
2341     }
2342     { }
2343     { \pgfusepath { } }
2344     \pgfnode
2345     { rectangle }
2346     { north }
2347     {
2348         \c_math_toggle_token
2349         \scriptstyle \l_@@_xdots_down_tl

```



```

2350         \c_math_toggle_token
2351     }
2352     { }
2353     { \pgfusepath { } }
2354 \endpgfscope
2355 }
2356 \pgfrememberpicturepositiononpagetrue
2357 \pgf@relevantforpicturesizefalse
2358 \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.

```

2359     \dim_zero_new:N \l_@@_l_dim
2360     \dim_set:Nn \l_@@_l_dim
2361     {
2362         \fp_to_dim:n
2363         {
2364             sqrt
2365             (
2366                 ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
2367                 +
2368                 ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2369             )
2370         }
2371     }

```

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.

```

2372     \bool_lazy_or:nnF
2373     { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2374     { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2375     \@@_draw_standard_dotted_line_i:
2376 \group_end:
2377 }
2378 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2379 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2380 {

```

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

```

2381     \bool_if:NTF \l_@@_initial_open_bool
2382     {
2383         \bool_if:NTF \l_@@_final_open_bool
2384         {
2385             \int_set:Nn \l_tmpa_int
2386             { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2387         }
2388         {
2389             \int_set:Nn \l_tmpa_int
2390             {
2391                 \dim_ratio:nn
2392                 { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2393                 \l_@@_inter_dots_dim
2394             }
2395         }
2396     }
2397     {
2398         \bool_if:NTF \l_@@_final_open_bool
2399         {
2400             \int_set:Nn \l_tmpa_int
2401             {
2402                 \dim_ratio:nn

```

```

2403         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2404         \l_@@_inter_dots_dim
2405     }
2406 }
2407 {
2408     \int_set:Nn \l_tmpa_int
2409     {
2410         \dim_ratio:nn
2411         { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2412         \l_@@_inter_dots_dim
2413     }
2414 }
2415 }

```

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

```

2416 \dim_set:Nn \l_tmpa_dim
2417 {
2418     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2419     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2420 }
2421 \dim_set:Nn \l_tmpb_dim
2422 {
2423     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2424     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2425 }

```

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`.

```

2426 \int_set:Nn \l_tmpb_int
2427 {
2428     \bool_if:NTF \l_@@_initial_open_bool
2429     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
2430     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
2431 }

```

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.

```

2432 \dim_gadd:Nn \l_@@_x_initial_dim
2433 {
2434     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2435     \dim_ratio:nn
2436     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2437     { 2 \l_@@_l_dim }
2438     * \l_tmpb_int
2439 }
2440 \dim_gadd:Nn \l_@@_y_initial_dim
2441 {
2442     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2443     \dim_ratio:nn
2444     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2445     { 2 \l_@@_l_dim }
2446     * \l_tmpb_int
2447 }
2448 \pgf@relevantforpicturesizefalse
2449 \int_step_inline:nnn 0 \l_tmpa_int
2450 {
2451     \pgfpathcircle
2452     { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2453     { \l_@@_radius_dim }
2454     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2455     \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim

```

```

2456     }
2457     \pgfusepathqfill
2458 }

```

## 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 for 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.

```

2459 \AtBeginDocument
2460 {
2461     \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
2462     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2463     \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
2464     {
2465         \int_compare:nNnTF \c@jCol = 0
2466         { \@@_error:nn { in~first~col } \Ldots }
2467         {
2468             \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2469             { \@@_error:nn { in~last~col } \Ldots }
2470             {
2471                 \@@_instruction_of_type:nn { Ldots }
2472                 { #1 , down = #2 , up = #3 }
2473             }
2474         }
2475         \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ldots }
2476         \bool_gset_true:N \g_@@_empty_cell_bool
2477     }

2478     \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
2479     {
2480         \int_compare:nNnTF \c@jCol = 0
2481         { \@@_error:nn { in~first~col } \Cdots }
2482         {
2483             \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2484             { \@@_error:nn { in~last~col } \Cdots }
2485             {
2486                 \@@_instruction_of_type:nn { Cdots }
2487                 { #1 , down = #2 , up = #3 }
2488             }
2489         }
2490         \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_cdots }
2491         \bool_gset_true:N \g_@@_empty_cell_bool
2492     }

2493     \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
2494     {
2495         \int_compare:nNnTF \c@iRow = 0
2496         { \@@_error:nn { in~first~row } \Vdots }

```

```

2497     {
2498         \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
2499         { \@@_error:nn { in~last~row } \Vdots }
2500         {
2501             \@@_instruction_of_type:nn { Vdots }
2502             { #1 , down = #2 , up = #3 }
2503         }
2504     }
2505     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_vdots }
2506     \bool_gset_true:N \g_@@_empty_cell_bool
2507 }

\exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
{
    \int_case:nnF \c@iRow
    {
2511        {
2512            0 { \@@_error:nn { in~first~row } \Ddots }
2513            \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
2514        }
2515        {
2516            \int_case:nnF \c@jCol
2517            {
2518                0 { \@@_error:nn { in~first~col } \Ddots }
2519                \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
2520            }
2521            {
2522                \@@_instruction_of_type:nn { Ddots }
2523                { #1 , down = #2 , up = #3 }
2524            }
2525        }
2526    }
2527    \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ddots }
2528    \bool_gset_true:N \g_@@_empty_cell_bool
2529 }

\exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
{
2531    \int_case:nnF \c@iRow
2532    {
2533        {
2534            0 { \@@_error:nn { in~first~row } \Iddots }
2535            \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
2536        }
2537        {
2538            \int_case:nnF \c@jCol
2539            {
2540                0 { \@@_error:nn { in~first~col } \Iddots }
2541                \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
2542            }
2543            {
2544                \@@_instruction_of_type:nn { Iddots }
2545                { #1 , down = #2 , up = #3 }
2546            }
2547        }
2548    }
2549    \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_iddots }
2550    \bool_gset_true:N \g_@@_empty_cell_bool
2551 }
2552 }

```

End of the \AtBeginDocument.

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

```

2553 \cs_new_protected:Npn \@@_Hspace:
2554 {
2555   \bool_gset_true:N \g_@@_empty_cell_bool
2556   \hspace
2557 }

```

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

```

2558 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
2559 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
2560 {
2561   \@@_old_multicolumn { #1 } { #2 } { #3 }

```

The `\peek_remove_spaces:n` is mandatory.

```

2562 \peek_remove_spaces:n
2563 {
2564   \int_compare:nNnT #1 > 1
2565   {
2566     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
2567     { \int_use:N \c@iRow - \int_use:N \c@jCol }
2568     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
2569     \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
2570     {
2571       { \int_use:N \c@iRow }
2572       { \int_use:N \c@jCol }
2573       { \int_use:N \c@iRow }
2574       { \int_eval:n { \c@jCol + #1 - 1 } }
2575     }
2576   }
2577   \int_gadd:Nn \c@jCol { #1 - 1 }
2578 }
2579 }

```

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`.

```

2580 \cs_new:Npn \@@_Hdotsfor:
2581 {
2582   \multicolumn { 1 } { C } { }
2583   \@@_Hdotsfor_i
2584 }

```

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:`).

```

2585 \AtBeginDocument
2586 {
2587   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2588   \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.

```

2589 \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
2590 {
2591   \tl_gput_right:Nx \g_@@_HVDotsfor_lines_tl
2592   {
2593     \@@_Hdotsfor:nnnn
2594     { \int_use:N \c@iRow }
2595     { \int_use:N \c@jCol }
2596     { #2 }
2597     {
2598       #1 , #3 ,

```

```

2599         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
2600     }
2601 }
2602 \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
2603 }
2604 }

```

Enf of \AtBeginDocument.

```

2605 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
2606 {
2607     \bool_set_false:N \l_@@_initial_open_bool
2608     \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

2609     \int_set:Nn \l_@@_initial_i_int { #1 }
2610     \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

2611     \int_compare:nNnTF #2 = 1
2612     {
2613         \int_set:Nn \l_@@_initial_j_int 1
2614         \bool_set_true:N \l_@@_initial_open_bool
2615     }
2616     {
2617         \cs_if_exist:cTF
2618         {
2619             pgf @ sh @ ns @ \@@_env:
2620             - \int_use:N \l_@@_initial_i_int
2621             - \int_eval:n { #2 - 1 }
2622         }
2623         { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
2624         {
2625             \int_set:Nn \l_@@_initial_j_int { #2 }
2626             \bool_set_true:N \l_@@_initial_open_bool
2627         }
2628     }
2629     \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
2630     {
2631         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2632         \bool_set_true:N \l_@@_final_open_bool
2633     }
2634     {
2635         \cs_if_exist:cTF
2636         {
2637             pgf @ sh @ ns @ \@@_env:
2638             - \int_use:N \l_@@_final_i_int
2639             - \int_eval:n { #2 + #3 }
2640         }
2641         { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
2642         {
2643             \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2644             \bool_set_true:N \l_@@_final_open_bool
2645         }
2646     }
2647     \group_begin:
2648     \int_compare:nNnTF { #1 } = 0
2649     { \color { nicematrix-first-row } }
2650     {
2651         \int_compare:nNnT { #1 } = \g_@@_row_total_int
2652         { \color { nicematrix-last-row } }
2653     }
2654     \keys_set:nn { NiceMatrix / xdots } { #4 }
2655     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }

```

```

2656 \@@_actually_draw_Ldots:
2657 \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).

```

2658 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
2659 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
2660 }

2661 \AtBeginDocument
2662 {
2663   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2664   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2665   \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
2666   {
2667     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
2668     {
2669       \@@_Vdotsfor:nnnn
2670       { \int_use:N \c@iRow }
2671       { \int_use:N \c@jCol }
2672       { #2 }
2673       {
2674         #1 , #3 ,
2675         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
2676       }
2677     }
2678   }
2679 }

```

Enf of \AtBeginDocument.

```

2680 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
2681 {
2682   \bool_set_false:N \l_@@_initial_open_bool
2683   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it’s easy.

```

2684 \int_set:Nn \l_@@_initial_j_int { #2 }
2685 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

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

```

2686 \int_compare:nNnTF #1 = 1
2687 {
2688   \int_set:Nn \l_@@_initial_i_int 1
2689   \bool_set_true:N \l_@@_initial_open_bool
2690 }
2691 {
2692   \cs_if_exist:cTF
2693   {
2694     pgf @ sh @ ns @ \@@_env:
2695     - \int_eval:n { #1 - 1 }
2696     - \int_use:N \l_@@_initial_j_int
2697   }
2698   { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
2699   {
2700     \int_set:Nn \l_@@_initial_i_int { #1 }
2701     \bool_set_true:N \l_@@_initial_open_bool
2702   }
2703 }
2704 \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
2705 {
2706   \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
2707   \bool_set_true:N \l_@@_final_open_bool

```

```

2708     }
2709     {
2710         \cs_if_exist:cTF
2711         {
2712             pgf @ sh @ ns @ \@@_env:
2713             - \int_eval:n { #1 + #3 }
2714             - \int_use:N \l_@@_final_j_int
2715         }
2716         { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
2717         {
2718             \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
2719             \bool_set_true:N \l_@@_final_open_bool
2720         }
2721     }
2722     \group_begin:
2723     \int_compare:nNnTF { #2 } = 0
2724     { \color { nicematrix-first-col } }
2725     {
2726         \int_compare:nNnT { #2 } = \g_@@_col_total_int
2727         { \color { nicematrix-last-col } }
2728     }
2729     \keys_set:nn { NiceMatrix / xdots } { #4 }
2730     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2731     \@@_actually_draw_Vdots:
2732     \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).

```

2733     \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
2734     { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
2735 }

```

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`.

```

2736 \cs_new_protected:Npn \@@_rotate:
2737 {
2738     \bool_if:NTF \l_@@_NiceTabular_bool
2739     { \group_insert_after:N \@@_rotate_ii: }
2740     { \group_insert_after:N \@@_rotate_i: }
2741 }
2742 \cs_new_protected:Npn \@@_rotate_i: { \group_insert_after:N \@@_rotate_ii: }
2743 \cs_new_protected:Npn \@@_rotate_ii: { \group_insert_after:N \@@_rotate_iii: }
2744 \cs_new_protected:Npn \@@_rotate_iii:
2745 {
2746     \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.

```

2747     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
2748     {
2749         \vbox_set_top:Nn \l_@@_cell_box
2750         {
2751             \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`).

```

2752         \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
2753         \box_use:N \l_@@_cell_box

```



```

2754     }
2755   }
2756 }

```

## 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>39</sup>

```

2757 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
2758   { \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).

```

2759 \AtBeginDocument
2760 {
2761   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
2762   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2763   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
2764     {
2765       \group_begin:
2766       \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
2767       \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2768       \use:x
2769       {
2770         \@@_line_i:nn
2771         { \@@_double_int_eval:n #2 \q_stop }
2772         { \@@_double_int_eval:n #3 \q_stop }
2773       }
2774       \group_end:
2775     }
2776 }
2777 \cs_new_protected:Npn \@@_line_i:nn #1 #2
2778 {
2779   \bool_set_false:N \l_@@_initial_open_bool
2780   \bool_set_false:N \l_@@_final_open_bool
2781   \bool_if:nTF
2782     {
2783       \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
2784       ||
2785       \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
2786     }
2787     {
2788       \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
2789     }
2790     { \@@_draw_line_ii:nn { #1 } { #2 } }
2791 }
2792 \AtBeginDocument
2793 {
2794   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
2795   {

```

---

<sup>39</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`.

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:`.

```

2796     \c_@@_pgfortikzpicture_tl
2797     \@@_draw_line_iii:nn { #1 } { #2 }
2798     \c_@@_endpgfortikzpicture_tl
2799   }
2800 }

```

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

```

2801 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
2802 {
2803   \pgfrememberpicturepositiononpagetrue
2804   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
2805   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2806   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2807   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
2808   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2809   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2810   \@@_draw_line:
2811 }

```

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).

## Commands available in the code-before

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

```

2812 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
2813 {
2814   \tl_set:Nn \l_tmpa_tl { #1 }
2815   \tl_set:Nn \l_tmpb_tl { #2 }
2816 }

```

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

```

2817 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
2818 {
2819   \tl_if_blank:nF { #2 }
2820   {
2821     \pgfpicture
2822     \pgf@relevantforpicturesizefalse
2823     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }

```

`\l_tmpa_dim` is the  $x$ -value of the right side of the rows.

```

2824     \@@_qpoint:n { col - 1 }
2825     \int_compare:nNnTF \l_@@_first_col_int = 0
2826     { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
2827     { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
2828     \@@_qpoint:n { col - \@@_succ:n \c@jCol }
2829     \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }
2830     \clist_map_inline:nn { #3 }
2831     {
2832       \tl_set:Nn \l_tmpa_tl { ##1 }
2833       \tl_if_in:NnTF \l_tmpa_tl { - }
2834       { \@@_cut_on_hyphen:w ##1 \q_stop }
2835       { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
2836       \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
2837       \tl_if_empty:NT \l_tmpb_tl
2838       { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }

```

```

2839         \int_compare:nNt \l_tmpb_tl > \c@iRow
2840         { \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`.

```

2841         @@_qpoint:n { row - @@_succ:n \l_tmpb_tl }
2842         \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
2843         @@_qpoint:n { row - \l_tmpa_tl }
2844         \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
2845         \pgfpathrectanglecorners
2846         { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
2847         { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
2848     }
2849     \pgfusepathqfill
2850     \endpgfpicture
2851 }
2852 }

```

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

```

2853 \NewDocumentCommand \@@_columncolor { 0 { } m m }
2854 {
2855     \tl_if_blank:nF { #2 }
2856     {
2857         \pgfpicture
2858         \pgf@relevantforpicturesizefalse
2859         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
2860         @@_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.

```

2861         \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
2862         @@_qpoint:n { row - @@_succ:n \c@iRow }
2863         \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
2864         \clist_map_inline:nn { #3 }
2865         {
2866             \tl_set:Nn \l_tmpa_tl { ##1 }
2867             \tl_if_in:NnTF \l_tmpa_tl { - }
2868             { \@@_cut_on_hyphen:w ##1 \q_stop }
2869             { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
2870             \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
2871             \tl_if_empty:NT \l_tmpb_tl
2872             { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
2873             \int_compare:nNt \l_tmpb_tl > \c@jCol
2874             { \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`.

```

2875         @@_qpoint:n { col - \l_tmpa_tl }
2876         \int_compare:nNtF \l_@@_first_col_int = \l_tmpa_tl
2877         { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
2878         { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
2879         @@_qpoint:n { col - @@_succ:n \l_tmpb_tl }
2880         \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
2881         \pgfpathrectanglecorners
2882         { \pgfpoint \l_tmpc_dim \l_tmpa_dim }
2883         { \pgfpoint \l_tmpd_dim \l_tmpb_dim }
2884     }
2885     \pgfusepathqfill
2886     \endpgfpicture
2887 }
2888 }

```

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

```

2889 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
2890 {
2891     \tl_if_blank:nF { #2 }

```

```

2892 {
2893   \pgfpicture
2894   \pgf@relevantforpicturesizefalse
2895   \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
2896   \clist_map_inline:nn { #3 }
2897   {
2898     \@@_cut_on_hyphen:w ##1 \q_stop
2899     \@@_qpoint:n { row - \l_tmpa_tl }
2900     \bool_lazy_and:nnT
2901       { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
2902       { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
2903       {
2904         \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
2905         \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
2906         \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
2907         \@@_qpoint:n { col - \l_tmpb_tl }
2908         \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
2909           { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
2910           { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
2911         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
2912         \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
2913         \pgfpathrectanglecorners
2914           { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
2915           { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
2916       }
2917     }
2918   \pgfusepathqfill
2919   \endpgfpicture
2920 }
2921 }

```

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

```

2922 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
2923 {
2924   \tl_if_blank:nF { #2 }
2925   {
2926     \pgfpicture
2927     \pgf@relevantforpicturesizefalse
2928     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
2929     \@@_cut_on_hyphen:w #3 \q_stop
2930     \bool_lazy_and:nnT
2931       { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
2932       { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
2933       {
2934         \@@_qpoint:n { row - \l_tmpa_tl }
2935         \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
2936         \@@_qpoint:n { col - \l_tmpb_tl }
2937         \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
2938           { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
2939           { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
2940         \@@_cut_on_hyphen:w #4 \q_stop
2941         \int_compare:nNnT \l_tmpa_tl > \c@iRow
2942           { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
2943         \int_compare:nNnT \l_tmpb_tl > \c@jCol
2944           { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
2945         \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
2946         \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
2947         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
2948         \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
2949         \pgfpathrectanglecorners
2950           { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
2951           { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
2952         \pgfusepathqfill

```

```

2953     }
2954     \endpgfpicture
2955 }
2956 }

```

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`.

```

2957 \NewDocumentCommand \@@_rowcolors { 0 { } m m m }
2958 {
2959   \int_step_inline:nnn { #2 } { \int_use:N \c@iRow }
2960   {
2961     \int_if_odd:nTF { ##1 }
2962     { \@@_rowcolor [ #1 ] { #3 } }
2963     { \@@_rowcolor [ #1 ] { #4 } }
2964     { ##1 }
2965   }
2966 }

```

```

2967 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
2968 {
2969   \int_step_inline:nn { \int_use:N \c@iRow }
2970   {
2971     \int_step_inline:nn { \int_use:N \c@jCol }
2972     {
2973       \int_if_even:nTF { ####1 + ##1 }
2974       { \@@_cellcolor [ #1 ] { #2 } }
2975       { \@@_cellcolor [ #1 ] { #3 } }
2976       { ##1 - ####1 }
2977     }
2978   }
2979 }

```

## 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`.

```

2980 \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`.

```

2981 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
2982 {
2983   \int_compare:nNnTF \l_@@_first_col_int = 0
2984   { \@@_OnlyMainNiceMatrix_i:n { #1 } }
2985   {
2986     \int_compare:nNnTF \c@jCol = 0
2987     {
2988       \int_compare:nNnF \c@iRow = { -1 }
2989       { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
2990     }
2991     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
2992   }
2993 }

```

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.

```

2994 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
2995 {
2996   \int_compare:nNnF \c@iRow = 0
2997     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
2998 }

```

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`).

In fact, independently of `\OnlyMainNiceMatrix`, which is a convenience given to the user, we have to modify the behaviour of the standard specifier “|”.

Remark first that the natural way to do that would be to redefine the specifier “|” with `\newcolumnntype`:

```
\newcolumnntype { | } { ! { \OnlyMainNiceMatrix \vline } }
```

However, this code fails if the user uses `\DefineShortVerb{\|}` of `fancyvrb`. Moreover, it would not be able to deal correctly with two consecutive specifiers “|” (in a preamble like `ccc||ccc`).

That's why we have done a redefinition of the macro `\@arrayrule` of `array` and this redefinition will add `\@@_vline:` instead of `\vline` in the preamble (that definition is in the beginning of `{NiceArrayWithDelims}`).

Here is the definition of `\@@_vline:`. This definition *must* be protected because you don't want that macro expanded during the construction of the preamble (the tests in `\@@_OnlyMainNiceMatrix:n` must be effective in each row and not once for all when the preamble is constructed). 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.

```

2999 \cs_new_protected:Npn \@@_vline:
3000 { \@@_OnlyMainNiceMatrix:n { { \CT@arc@ \vline } } }

```

The command `\@@_draw_vlines` will be executed when the user uses the option `vlines` (which draws all the vlins of the array).

```

3001 \cs_new_protected:Npn \@@_draw_vlines:
3002 {
3003   \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 if `colortbl` is not loaded.

```

3004   \CT@arc@
3005   \pgfpicture
3006   \pgfrememberpicturepositiononpagetrue
3007   \pgf@relevantforpicturesizefalse
3008   \pgfsetlinewidth \arrayrulewidth

```

First, we compute in `\l_tmpa_dim` the height of the rules we have to draw.

```

3009   \@@_qpoint:n { row - 1 }
3010   \dim_set_eq:NN \l_tmpa_dim \pgf@y
3011   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3012   \dim_sub:Nn \l_tmpa_dim \pgf@y

```

We translate vertically to take into account the potential “last row”.

```

3013   \dim_zero:N \l_tmpb_dim
3014   \int_compare:nNnT \l_@@_last_row_int > { -1 }
3015   {
3016     \dim_set_eq:NN \l_tmpb_dim \g_@@_dp_last_row_dim
3017     \dim_add:Nn \l_tmpb_dim \g_@@_ht_last_row_dim

```

We adjust the value of `\l_tmpa_dim` by the width of the horizontal rule just before the “last row”.

```

3018     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3019     \dim_add:Nn \l_tmpa_dim \pgf@y
3020     \@@_qpoint:n { row - \@@_succ:n \g_@@_row_total_int }
3021     \dim_sub:Nn \l_tmpa_dim \pgf@y
3022     \dim_sub:Nn \l_tmpa_dim \l_tmpb_dim
3023   }
3024   \dim_add:Nn \l_tmpa_dim \arrayrulewidth

```

Now, we can draw the vertical rules with a loop.

```

3025   \int_step_inline:nnn
3026   { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3027   { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3028   {
3029     \pgfpathmoveto { \@@_qpoint:n { col - ##1 } }
3030     \pgfpathlineto
3031     {
3032       \pgfpointadd
3033       { \@@_qpoint:n { col - ##1 } }
3034       { \pgfpoint \c_zero_dim { \l_tmpb_dim + \l_tmpa_dim } }
3035     }
3036   }
3037   \pgfusepathqstroke
3038   \endpgfpicture
3039   \group_end:
3040 }

```

## The key `hvlines`

```

3041 \cs_new_protected:Npn \@@_draw_hlines:
3042 {
3043   \pgfpicture
3044   \CT@arc@
3045   \pgfrememberpicturepositiononpagetrue
3046   \pgf@relevantforpicturesizefalse
3047   \pgfsetlinewidth \arrayrulewidth
3048   \int_step_inline:nnn
3049   { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3050   { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3051   {
3052     \@@_qpoint:n { row - ##1 }
3053     \dim_set_eq:NN \l_tmpa_dim \pgf@y
3054     \pgfpathmoveto { \pgfpoint \pgf@x \pgf@y }
3055     \@@_qpoint:n { col - \@@_succ:n { \c@jCol } }
3056     \dim_set:Nn \l_tmpb_dim { \pgf@x + 0.5 \arrayrulewidth }
3057     \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3058   }
3059   \pgfusepathqstroke
3060   \endpgfpicture
3061 }

```

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.).

```

3062 \cs_new_protected:Npn \@@_draw_hvlines:
3063 {
3064   \bool_lazy_and:nnTF
3065   { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
3066   { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
3067   \@@_draw_hvlines_i:
3068   \@@_draw_hvlines_ii:

```

```
3069 }
```

This version is only for efficiency. The general case (in `\@@_draw_hvlines_ii:`) does the job in all case (but slower).

```
3070 \cs_new_protected:Npn \@@_draw_hvlines_i:
3071 {
3072   \@@_draw_hlines:
3073   \@@_draw_vlines:
3074 }
```

Now, the general case, where there are blocks or dots in the array.

```
3075 \cs_new_protected:Npn \@@_draw_hvlines_ii:
3076 {
3077   \group_begin:
3078   \CT@arc@
```

First, the exterior rectangle of the array (only in `{NiceArray}` and `{NiceTabular}`).

```
3079   \bool_if:NT \l_@@_NiceArray_bool
3080   {
3081     \pgfpicture
3082     \pgfrememberpicturepositiononpagetrue
3083     \pgf@relevantforpicturesizefalse
3084     \pgfsetlinewidth \arrayrulewidth
3085     \pgfsetrectcap
3086     \@@_qpoint:n { col - 1 }
3087     \dim_set_eq:NN \l_tmpa_dim \pgf@x
3088     \@@_qpoint:n { row - 1 }
3089     \dim_set_eq:NN \l_tmpb_dim \pgf@y
3090     \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3091     \dim_set_eq:NN \l_tmpc_dim \pgf@x
3092     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3093     \pgfpathrectanglecorners
3094     { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3095     { \pgfpoint \l_tmpc_dim \pgf@y }
3096     \pgfusepathqstroke
3097     \endpgfpicture
3098   }
```

First, the horizontal rules in the interior of the array.

```
3099   \int_step_variable:nnNn 2 \c@iRow \l_tmpa_tl
3100   {
3101     \int_step_variable:nnNn \c@jCol \l_tmpb_tl
3102     {
```

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.

```
3103     \bool_gset_true:N \g_tmpa_bool
3104     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3105     { \@@_test_if_hline_in_block:nnnn ##1 }
3106     \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3107     { \@@_test_if_hline_in_block:nnnn ##1 }
3108     \bool_if:NT \g_tmpa_bool
3109     {
3110       \pgfpicture
3111       \pgfrememberpicturepositiononpagetrue
3112       \pgf@relevantforpicturesizefalse
3113       \pgfsetlinewidth \arrayrulewidth
3114       \pgfsetrectcap
3115       \@@_qpoint:n { row - \l_tmpa_tl }
3116       \dim_set_eq:NN \l_tmpb_dim \pgf@y
3117       \@@_qpoint:n { col - \l_tmpb_tl }
3118       \dim_set_eq:NN \l_tmpa_dim \pgf@x
```



```

3119         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3120         \dim_set_eq:NN \l_tmpc_dim \pgf@x
3121         \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3122         \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3123         \pgfusepathqstroke
3124         \endpgfpicture
3125     }
3126 }
3127 }

```

Now, the vertical rules in the interior of the array.

```

3128     \int_step_variable:nNn \c@iRow \l_tmpa_tl
3129     {
3130         \int_step_variable:nnNn 2 \c@jCol \l_tmpb_tl
3131         {

```

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.

```

3132         \bool_gset_true:N \g_tmpa_bool
3133         \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3134         { \@@_test_if_vline_in_block:nnnn ##1 }
3135         \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3136         { \@@_test_if_vline_in_block:nnnn ##1 }
3137         \bool_if:NT \g_tmpa_bool
3138         {
3139             \pgfpicture
3140             \pgfrememberpicturepositiononpagetrue
3141             \pgf@relevantforpicturesizefalse
3142             \pgfsetlinewidth \arrayrulewidth
3143             \pgfsetrectcap
3144             \@@_qpoint:n { row - \l_tmpa_tl }
3145             \dim_set_eq:NN \l_tmpb_dim \pgf@y
3146             \@@_qpoint:n { col - \l_tmpb_tl }
3147             \dim_set_eq:NN \l_tmpa_dim \pgf@x
3148             \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3149             \dim_set_eq:NN \l_tmpc_dim \pgf@y
3150             \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3151             \pgfpathlineto { \pgfpoint \l_tmpa_dim \l_tmpc_dim }
3152             \pgfusepathqstroke
3153             \endpgfpicture
3154         }
3155     }
3156 }

```

The group was for the color of the rules.

```

3157     \group_end:
3158     \seq_gclear:N \g_@@_pos_of_xdots_seq
3159 }

```

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`.

```

3160 \cs_set_protected:Npn \@@_test_if_hline_in_block:nnnn #1 #2 #3 #4
3161 {
3162     \int_compare:nNnT \l_tmpa_tl > { #1 }
3163     {
3164         \int_compare:nNnT \l_tmpa_tl < { #3 + 1 }
3165         {
3166             \int_compare:nNnT \l_tmpb_tl > { #2 - 1 }
3167             {
3168                 \int_compare:nNnT \l_tmpb_tl < { #4 + 1 }
3169                 { \bool_gset_false:N \g_tmpa_bool }
3170             }

```

```

3171     }
3172   }
3173 }

```

The same for vertical rules.

```

3174 \cs_set_protected:Npn \@@_test_if_vline_in_block:nnnn #1 #2 #3 #4
3175 {
3176   \int_compare:nNnT \l_tmpa_tl > { #1 - 1 }
3177   {
3178     \int_compare:nNnT \l_tmpa_tl < { #3 + 1 }
3179     {
3180       \int_compare:nNnT \l_tmpb_tl > { #2 }
3181       {
3182         \int_compare:nNnT \l_tmpb_tl < { #4 + 1 }
3183         { \bool_gset_false:N \g_tmpa_bool }
3184       }
3185     }
3186   }
3187 }

```

## 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`.

```

3188 \cs_new:Npn \@@_hdottedline:
3189 {
3190   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
3191   \@@_hdottedline_i:
3192 }

```

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

```

3193 \cs_new_protected:Npn \@@_hdottedline_i:
3194 {

```

We write in the code-after the instruction that will eventually 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).

```

3195   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3196   { \@@_hdottedline:n { \int_use:N \c_iRow } }
3197 }

```

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.

```

3198 \AtBeginDocument
3199 {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”.

```

3200   \cs_new_protected:Npx \@@_hdottedline:n #1
3201   {
3202     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
3203     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
3204     \c_@@_pgfortikzpicture_tl
3205     \@@_hdottedline_i:n { #1 }
3206     \c_@@_endpgfortikzpicture_tl
3207   }
3208 }

```

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

```

3209 \cs_new_protected:Npn \@@_hdottedline_i:n #1
3210 {
3211   \pgfrememberpicturepositiononpagetrue
3212   \@@_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).

```

3213   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3214   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
3215   \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

The dotted line will be extended if the user uses `margin` (or `left-margin` and `right-margin`).

The aim is that, by standard the dotted line fits between square brackets (`\hline` doesn’t).

```

\begin{bNiceMatrix}
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & 4 \\
\hdottedline
1 & 2 & 3 & 4
\end{bNiceMatrix}

```

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

But, if the user uses `margin`, the dotted line extends to have the same width as a `\hline`.

```

\begin{bNiceMatrix}[margin]
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & 4 \\
\hdottedline
1 & 2 & 3 & 4
\end{bNiceMatrix}

```

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

```

3216   \@@_qpoint:n { col - 1 }
3217   \dim_set:Nn \l_@@_x_initial_dim
3218   {
3219     \pgf@x +
3220     \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep
3221     - \l_@@_left_margin_dim
3222   }
3223   \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3224   \dim_set:Nn \l_@@_x_final_dim
3225   {
3226     \pgf@x -
3227     \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep
3228     + \l_@@_right_margin_dim
3229   }

```

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.

```

3230   \tl_set:Nn \l_tmpa_tl { ( }
3231   \tl_if_eq:NNF \l_@@_left_delim_tl \l_tmpa_tl
3232   { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
3233   \tl_set:Nn \l_tmpa_tl { ) }
3234   \tl_if_eq:NNF \l_@@_right_delim_tl \l_tmpa_tl
3235   { \dim_gsub:Nn \l_@@_x_final_dim { 0.5 \l_@@_inter_dots_dim } }

```

As for now, we have no option to control the style of the lines drawn by `\hdottedline` and the specifier “.” in the preamble. That’s why we impose the style `standard`.

```

3236   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
3237   \@@_draw_line:
3238 }

```

## Vertical dotted lines

```

3239 \cs_new_protected:Npn \@@_vdottedline:n #1
3240 {
3241     \bool_set_true:N \l_@@_initial_open_bool
3242     \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”.

```

3243     \bool_if:NTF \c_@@_tikz_loaded_bool
3244     {
3245         \tikzpicture
3246         \@@_vdottedline_i:n { #1 }
3247         \endtikzpicture
3248     }
3249     {
3250         \pgfpicture
3251         \@@_vdottedline_i:n { #1 }
3252         \endpgfpicture
3253     }
3254 }

```

```

3255 \cs_new_protected:Npn \@@_vdottedline_i:n #1
3256 {

```

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.

```

3257     \CT@arc@
3258     \pgfrememberpicturepositiononpagetrue
3259     \@@_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).

```

3260     \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
3261     \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
3262     \@@_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.

```

3263     \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
3264     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3265     \dim_set:Nn \l_@@_y_final_dim { \pgf@y + 0.5 \l_@@_inter_dots_dim }

```

As for now, we have no option to control the style of the lines drawn by `\hdottedline` and the specifier “:” in the preamble. That’s why we impose the style `standard`.

```

3266     \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
3267     \@@_draw_line:
3268 }

```

## 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.

```

3269 \bool_new:N \l_@@_block_auto_columns_width_bool

```

As of now, there is only one option available for the environment `{NiceMatrixBlock}`.

```

3270 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
3271 {
3272     auto-columns-width .code:n =
3273     {

```

```

3274     \bool_set_true:N \l_@@_block_auto_columns_width_bool
3275     \dim_gzero_new:N \g_@@_max_cell_width_dim
3276     \bool_set_true:N \l_@@_auto_columns_width_bool
3277   }
3278 }

3279 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
3280 {
3281   \int_gincr:N \g_@@_NiceMatrixBlock_int
3282   \dim_zero:N \l_@@_columns_width_dim
3283   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
3284   \bool_if:NT \l_@@_block_auto_columns_width_bool
3285   {
3286     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
3287     {
3288       \exp_args:Nnc \dim_set:Nn \l_@@_columns_width_dim
3289       { @@_max_cell_width _ \int_use:N \g_@@_NiceMatrixBlock_int }
3290     }
3291   }
3292 }

```

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).

```

3293 {
3294   \bool_if:NT \l_@@_block_auto_columns_width_bool
3295   {
3296     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
3297     \iow_shipout:Nx \@mainaux
3298     {
3299       \cs_gset:cpn
3300       { @@_max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }

```

For technical reasons, we have to include the width of an eventual rule on the right side of the cells.

```

3301       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
3302     }
3303     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
3304   }
3305 }

```

## The extra nodes

First, two variants of the functions \dim\_min:nn and \dim\_max:nn.

```

3306 \cs_generate_variant:Nn \dim_min:nn { v n }
3307 \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 `\c_max_dim` or `-\c_max_dim`.

```

3308 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
3309 {
3310   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3311   {
3312     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
3313     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
3314     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
3315     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
3316   }
3317   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3318   {
3319     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
3320     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
3321     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
3322     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
3323   }

```

We begin the two nested loops over the rows and the columns of the array.

```

3324   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3325   {
3326     \int_step_variable:nnNn
3327     \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.

```

3328     {
3329       \cs_if_exist:cT
3330       { 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`.

```

3331     {
3332       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
3333       \dim_set:cn { l_@@_row_\@@_i: _min_dim }
3334       { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
3335       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
3336       {
3337         \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
3338         { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
3339       }

```

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`.

```

3340       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
3341       \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
3342       { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
3343       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
3344       {
3345         \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
3346         { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
3347       }
3348     }
3349   }
3350 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

3351   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3352   {
3353     \dim_compare:nNnT
3354     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim

```

```

3355     {
3356         \@@_qpoint:n { row - \@@_i: - base }
3357         \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
3358         \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
3359     }
3360 }
3361 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3362 {
3363     \dim_compare:nNnT
3364     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
3365     {
3366         \@@_qpoint:n { col - \@@_j: }
3367         \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
3368         \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
3369     }
3370 }
3371 }

```

Here is the command `\@@_create_medium_nodes:`. When this command is used, the “medium nodes” are created.

```

3372 \cs_new_protected:Npn \@@_create_medium_nodes:
3373 {
3374     \pgfpicture
3375     \pgfrememberpicturepositiononpagetrue
3376     \pgf@relevantforpicturesizefalse
3377     \@@_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”.

```

3378     \tl_set:Nn \l_@@_suffix_tl { -medium }
3379     \@@_create_nodes:
3380     \endpgfpicture
3381 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones<sup>40</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:`.

```

3382 \cs_new_protected:Npn \@@_create_large_nodes:
3383 {
3384     \pgfpicture
3385     \pgfrememberpicturepositiononpagetrue
3386     \pgf@relevantforpicturesizefalse
3387     \@@_computations_for_medium_nodes:
3388     \@@_computations_for_large_nodes:
3389     \tl_set:Nn \l_@@_suffix_tl { - large }
3390     \@@_create_nodes:
3391     \endpgfpicture
3392 }
3393 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
3394 {
3395     \pgfpicture
3396     \pgfrememberpicturepositiononpagetrue
3397     \pgf@relevantforpicturesizefalse
3398     \@@_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”.

```

3399     \tl_set:Nn \l_@@_suffix_tl { - medium }

```

---

<sup>40</sup>If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

```

3400 \@@_create_nodes:
3401 \@@_computations_for_large_nodes:
3402 \tl_set:Nn \l_@@_suffix_tl { - large }
3403 \@@_create_nodes:
3404 \endpgfpicture
3405 }

```

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.

```

3406 \cs_new_protected:Npn \@@_computations_for_large_nodes:
3407 {
3408   \int_set:Nn \l_@@_first_row_int 1
3409   \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`.

```

3410   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
3411   {
3412     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
3413     {
3414       (
3415         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
3416         \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
3417       )
3418       / 2
3419     }
3420     \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
3421     { l_@@_row _ \@@_i: _ min _ dim }
3422   }
3423   \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
3424   {
3425     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
3426     {
3427       (
3428         \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
3429         \dim_use:c
3430         { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
3431       )
3432       / 2
3433     }
3434     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
3435     { l_@@_column _ \@@_j: _ max _ dim }
3436   }

```

Here, we have to use `\dim_sub:cn` because of the number 1 in the name.

```

3437   \dim_sub:cn
3438   { l_@@_column _ 1 _ min _ dim }
3439   \l_@@_left_margin_dim
3440   \dim_add:cn
3441   { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
3442   \l_@@_right_margin_dim
3443 }

```

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).

```

3444 \cs_new_protected:Npn \@@_create_nodes:
3445 {
3446   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:

```



```

3447 {
3448   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3449   {

```

We draw the rectangular node for the cell ( $\@@_i$ - $\@@_j$ ).

```

3450     \@@_pgf_rect_node:nnnnn
3451     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3452     { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
3453     { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
3454     { \dim_use:c { l_@@_column_ \@@_j: _max_dim } }
3455     { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
3456     \str_if_empty:NF \l_@@_name_str
3457     {
3458       \pgfnodealias
3459       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
3460       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3461     }
3462   }
3463 }

```

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  $n$ .

```

3464   \seq_mapthread_function:NNN
3465   \g_@@_multicolumn_cells_seq
3466   \g_@@_multicolumn_sizes_seq
3467   \@@_node_for_multicolumn:nn
3468 }

```

```

3469 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
3470 {
3471   \cs_set:Npn \@@_i: { #1 }
3472   \cs_set:Npn \@@_j: { #2 }
3473 }

```

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”).

```

3474 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
3475 {
3476   \@@_extract_coords_values: #1 \q_stop
3477   \@@_pgf_rect_node:nnnnn
3478   { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3479   { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
3480   { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
3481   { \dim_use:c { l_@@_column_ \int_eval:n { \@@_j: +#2-1 } _max_dim } }
3482   { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
3483   \str_if_empty:NF \l_@@_name_str
3484   {
3485     \pgfnodealias
3486     { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
3487     { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
3488   }
3489 }

```

## Block matrices

The code in this section is for the construction of *block matrices*. It 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)

```
3490 \NewDocumentCommand \@@_Block: { 0 { } m D < > { } m }
3491 { \@@_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.

```
3492 \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.

```
3493 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
3494 {
```

```
3495     \tl_set:Nx \l_tmpa_tl
3496     {
3497         { \int_use:N \c@iRow }
3498         { \int_use:N \c@jCol }
3499         { \int_eval:n { \c@iRow + #1 - 1 } }
3500         { \int_eval:n { \c@jCol + #2 - 1 } }
3501     }
```

Now, `\l_tmpa_tl` contains a “object” corresponding to the position of the block with four components surrounded by brackets:

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

We store this information in the sequence `\g_@@_pos_of_blocks_seq`.

```
3502     \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 “objet” with six components:

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

```
3503     \seq_gput_left:Nx \g_@@_blocks_seq
3504     {
3505         \l_tmpa_tl
3506         { #3 }
3507         \exp_not:n { { #4 \@@_math_toggle_token: #5 \@@_math_toggle_token: } }
3508     }
3509 }
```

The key `tikz` is for Tikz options used when the PGF node of the block is created.

```
3510 \keys_define:nn { NiceMatrix / Block }
3511 {
3512     tikz .tl_set:N = \l_@@_tikz_tl ,
3513     tikz .value_required:n = true ,
3514 }
```

The command `\@@_draw_blocks:` will draw all the blocks. This command is used after the construction of the array.

```
3515 \cs_new_protected:Npn \@@_draw_blocks:
3516 { \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iii:nnnnn ##1 } }
3517 % \end{macrocode}
3518 %
3519 % \begin{macrocode}
3520 \cs_new_protected:Npn \@@_Block_iii:nnnnn #1 #2 #3 #4 #5 #6
3521 {
```

The group is for the keys.

```
3522     \group_begin:
3523     \keys_set:nn { NiceMatrix / Block } { #5 }
```

```

3524 \bool_lazy_or:nnTF
3525 { \int_compare_p:nNn { #3 } > \c@iRow }
3526 { \int_compare_p:nNn { #4 } > \c@jCol }
3527 { \msg_error:nnnn { nicematrix } { Block-too~large } { #1 } { #2 } }
3528 {

```

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.

```

3529 \hbox_set:Nn \l_@@_cell_box { #6 }

```

The construction of the node corresponding to the merged cells.

```

3530 \pgfpicture
3531 \pgfrememberpicturepositiononpagetrue
3532 \pgf@relevantforpicturesizefalse
3533 \@@_qpoint:n { row - #1 }
3534 \dim_set_eq:NN \l_tmpa_dim \pgf@y
3535 \@@_qpoint:n { col - #2 }
3536 \dim_set_eq:NN \l_tmpb_dim \pgf@x
3537 \@@_qpoint:n { row - \@@_succ:n { #3 } }
3538 \dim_set_eq:NN \l_tmpc_dim \pgf@y
3539 \@@_qpoint:n { col - \@@_succ:n { #4 } }
3540 \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 as arguments the name of the node and the four coordinates of two opposite corner points of the rectangle.

```

3541 \begin { pgfscope }
3542 \exp_args:Nx \pgfset { \l_@@_tikz_tl }
3543 \@@_pgf_rect_node:nnnnn
3544 { \@@_env: - #1 - #2 - block }
3545 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
3546 \end { pgfscope }

```

If the creation of the “medium nodes” is required, we create a “medium node” for the block. The function `\@@_pgf_rect_node:nnnnn` takes as arguments the name of the node and two PGF points.

```

3547 \bool_if:NT \l_@@_medium_nodes_bool
3548 {
3549 \@@_pgf_rect_node:nnn
3550 { \@@_env: - #1 - #2 - block - medium }
3551 { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
3552 { \pgfpointanchor { \@@_env: - #3 - #4 - medium } { south-east } }
3553 }

```

Now, we will put the label of the block.

```

3554 \int_compare:nNnTF { #1 } = { #3 }
3555 {

```

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`.

```

3556 \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

We retrieve (in `\pgf@x`) the  $x$ -value of the center of the block.

```

3557 \@@_qpoint:n { #1 - #2 - block }

```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```

3558 \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
3559 \pgfnode { rectangle } { base }
3560 { \box_use_drop:N \l_@@_cell_box } { } { }
3561 }

```

If the number of rows is different of 1, we put the label of the block in the center of the node (the label of the block has been composed in `\l_@@_cell_box`).

```

3562     {
3563         \pgftransformshift { \l_@@_qpoint:n { #1 - #2 - block } }
3564         \pgfnode { rectangle } { center }
3565         { \box_use_drop:N \l_@@_cell_box } { } { }
3566     }
3567     \endpgfpicture
3568 }
3569 \group_end:
3570 }

```

## How to draw the dotted lines transparently

```

3571 \cs_set_protected:Npn \l_@@_renew_matrix:
3572 {
3573     \RenewDocumentEnvironment { pmatrix } { } { }
3574     { \pNiceMatrix }
3575     { \endpNiceMatrix }
3576     \RenewDocumentEnvironment { vmatrix } { } { }
3577     { \vNiceMatrix }
3578     { \endvNiceMatrix }
3579     \RenewDocumentEnvironment { Vmatrix } { } { }
3580     { \VNiceMatrix }
3581     { \endVNiceMatrix }
3582     \RenewDocumentEnvironment { bmatrix } { } { }
3583     { \bNiceMatrix }
3584     { \endbNiceMatrix }
3585     \RenewDocumentEnvironment { Bmatrix } { } { }
3586     { \BNiceMatrix }
3587     { \endBNiceMatrix }
3588 }

```

## Automatic arrays

```

3589 \cs_new_protected:Npn \l_@@_set_size:n #1-#2 \q_stop
3590 {
3591     \int_set:Nn \l_@@_nb_rows_int { #1 }
3592     \int_set:Nn \l_@@_nb_cols_int { #2 }
3593 }
3594 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
3595 {
3596     \int_zero_new:N \l_@@_nb_rows_int
3597     \int_zero_new:N \l_@@_nb_cols_int
3598     \l_@@_set_size:n #4 \q_stop
3599     \begin { NiceArrayWithDelims } { #1 } { #2 }
3600         { * { \l_@@_nb_cols_int } { C } } [ #3 , #5 , #7 ]
3601     \int_compare:nNnT \l_@@_first_row_int = 0
3602     {
3603         \int_compare:nNnT \l_@@_first_col_int = 0 { & }
3604         \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
3605         \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
3606     }
3607     \prg_replicate:nn \l_@@_nb_rows_int
3608     {
3609         \int_compare:nNnT \l_@@_first_col_int = 0 { & }

```

You put `{ }` before `#6` to avoid a hasty expansion of an eventual `\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`).

```

3610     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
3611     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\\
3612   }
3613   \int_compare:nNnT \l_@@_last_row_int > { -2 }
3614   {
3615     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
3616     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
3617     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\\
3618   }
3619   \end { NiceArrayWithDelims }
3620 }
3621 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
3622 {
3623   \cs_set_protected:cpn { #1 AutoNiceMatrix }
3624   {
3625     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
3626     \AutoNiceMatrixWithDelims { #2 } { #3 }
3627   }
3628 }
3629 \@@_define_com:nnn p ( )
3630 \@@_define_com:nnn b [ ]
3631 \@@_define_com:nnn v | |
3632 \@@_define_com:nnn V \| \|
3633 \@@_define_com:nnn B \{ \}

```

We define also an command `\AutoNiceMatrix` similar to the environment `{NiceMatrix}`.

```

3634 \NewDocumentCommand \AutoNiceMatrix { 0 { } m 0 { } m ! 0 { } }
3635 {
3636   \group_begin:
3637     \bool_set_true:N \l_@@_NiceArray_bool
3638     \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
3639   \group_end:
3640 }

```

## The redefinition of the command `\dotfill`

```

3641 \cs_set_eq:NN \@@_dotfill \dotfill
3642 \cs_new_protected:Npn \@@_dotfill:
3643 {

```

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}`).

```

3644   \@@_dotfill
3645   \bool_if:NT \l_@@_NiceTabular_bool
3646   { \group_insert_after:N \@@_dotfill_ii: }
3647   { \group_insert_after:N \@@_dotfill_i: }
3648 }
3649 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
3650 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }

```

Now, if the box is not empty (unfortunately, we can’t actually test whether the box is empty and that’s why we only consider it’s width), we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in the cell of the array, and it will extend, since it is no longer in `\l_@@_cell_box`.

```

3651 \cs_new_protected:Npn \@@_dotfill_iii:
3652 { \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } = \c_zero_dim \@@_dotfill }

```

## The command `\diagbox`

```

3653 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
3654 {
3655   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3656   {
3657     \@@_actually_diagbox:nnnn
3658     { \int_use:N \c_iRow } { \int_use:N \c_jCol } { #1 } { #2 }
3659   }

```

```
3660 }
```

The two arguments of `\@@_actually_diagbox:nn` are the number of row and the number of column of the cell to slash. The two other are the elements to draw below and above the diagonal line.

```
3661 \cs_new_protected:Npn \@@_actually_diagbox:nnnn #1 #2 #3 #4
3662 {
3663   \pgfpicture
3664   \pgf@relevantforpicturesizefalse
3665   \pgfrememberpicturepositiononpagetrue
3666   \@@_qpoint:n { row - #1 }
3667   \dim_set_eq:NN \l_tmpa_dim \pgf@y
3668   \@@_qpoint:n { col - #2 }
3669   \dim_set_eq:NN \l_tmpb_dim \pgf@x
3670   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3671   \@@_qpoint:n { row - \@@_succ:n { #1 } }
3672   \dim_set_eq:NN \l_tmpc_dim \pgf@y
3673   \@@_qpoint:n { col - \@@_succ:n { #2 } }
3674   \dim_set_eq:NN \l_tmpd_dim \pgf@x
3675   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
3676   {
```

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.

```
3677   \CT@arc@
3678   \pgfsetroundcap
3679   \pgfusepathqstroke
3680 }
3681 \pgfset { inner~sep = 1 pt }
3682 \pgfscope
3683 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3684 \pgfnode { rectangle } { south-west }
3685   { \@@_math_toggle_token: #3 \@@_math_toggle_token: } { } { }
3686 \endpgfscope
3687 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3688 \pgfnode { rectangle } { north-east }
3689   { \@@_math_toggle_token: #4 \@@_math_toggle_token: } { } { }
3690 \endpgfpicture
3691 }
```

## The command `\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. 66.

The command `\CodeAfter` catches everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```
3692 \cs_new_protected:Npn \@@_CodeAfter:n #1 \end
3693 {
3694   \tl_gput_right:Nn \g_@@_code_after_tl { #1 }
3695   \@@_CodeAfter_i:n
3696 }
```

We catch the argument of the command `\end` (in `#1`).

```
3697 \cs_new_protected:Npn \@@_CodeAfter_i:n #1
3698 {
```

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.

```
3699   \str_set:NV \l_tmpa_str \@currenvir
3700   \bool_if:NTF { \str_if_eq_p:Vn \l_tmpa_str { #1 } }
3701     { \end { #1 } }
```

If this is not the `\end` we are looking for, we put those tokens in `\g_@@_code_after_tl` and we go on searching for the next command `\end` with a recursive call to the command `\@@_CodeAfter:n`.

```
3702   {
```

```

3703     \tl_gput_right:Nn \g_@@_code_after_tl { \end { #1 } }
3704     \@@_CodeAfter:n
3705   }
3706 }

```

## 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.

```

3707 \keys_define:nn { NiceMatrix / Package }
3708 {
3709   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
3710   renew-dots .value_forbidden:n = true ,
3711   renew-matrix .code:n = \@@_renew_matrix: ,
3712   renew-matrix .value_forbidden:n = true ,
3713   transparent .meta:n = { renew-dots , renew-matrix } ,
3714   transparent .value_forbidden:n = true,
3715 }
3716 \ProcessKeysOptions { NiceMatrix / Package }

```

## Error messages of the package

The following command converts all the elements of a sequence (which are token lists) into strings.

```

3717 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
3718 {
3719   \seq_clear:N \l_tmpa_seq
3720   \seq_map_inline:Nn #1
3721   {
3722     \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
3723   }
3724   \seq_set_eq:NN #1 \l_tmpa_seq
3725 }

```

The following command creates a sequence of strings (`str`) from a `clist`.

```

3726 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
3727 {
3728   \seq_set_from_clist:Nn #1 { #2 }
3729   \@@_convert_to_str_seq:N #1
3730 }
3731 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
3732 {
3733   NiceMatrix ,
3734   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
3735 }

```

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`.

```

3736 \cs_new_protected:Npn \@@_error_too_much_cols:
3737 {
3738   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
3739   {
3740     \int_compare:nNnTF \l_@@_last_col_int = { -2 }

```

```

3741     { \@@_fatal:n { too-much-cols-for-matrix } }
3742     {
3743       \bool_if:NF \l_@@_last_col_without_value_bool
3744       { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
3745     }
3746   }
3747   { \@@_fatal:n { too-much-cols-for-array } }
3748 }

```

The following command must *not* be protected since it's used in an error message.

```

3749 \cs_new:Npn \@@_message_hdotsfor:
3750 {
3751   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
3752   { ~Maybe-your-use-of-\token_to_str:N \Hdotsfor\ is-incorrect.}
3753 }
3754 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
3755 {
3756   You-try-to-use-more-columns-than-allowed-by-your~
3757   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of~
3758   columns-is-\int_eval:n { \l_@@_last_col_int - 1 }~(plus-the-potential~
3759   exterior-ones).~This-error-is-fatal.
3760 }
3761 \@@_msg_new:nn { too-much-cols-for-matrix }
3762 {
3763   You-try-to-use-more-columns-than-allowed-by-your~
3764   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall-that-the-maximal~
3765   number-of-columns-for-a-matrix-is-fixed-by-the-LaTeX-counter~
3766   'MaxMatrixCols'.~Its-actual-value-is-\int_use:N \c@MaxMatrixCols.~
3767   This-error-is-fatal.
3768 }

```

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`.

```

3769 \@@_msg_new:nn { too-much-cols-for-array }
3770 {
3771   You-try-to-use-more-columns-than-allowed-by-your~
3772   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of-columns-is~
3773   \int_eval:n { \c@jCol - 1 }~(plus-the-potential-exterior-ones).~
3774   This-error-is-fatal.
3775 }
3776 \@@_msg_new:nn { in-first-col }
3777 {
3778   You-can't-use-the-command~#1 in-the-first-column-(number~0)-of-the-array.\\
3779   If-you-go-on,~this-command-will-be-ignored.
3780 }
3781 \@@_msg_new:nn { in-last-col }
3782 {
3783   You-can't-use-the-command~#1 in-the-last-column-(exterior)-of-the-array.\\
3784   If-you-go-on,~this-command-will-be-ignored.
3785 }
3786 \@@_msg_new:nn { in-first-row }
3787 {
3788   You-can't-use-the-command~#1 in-the-first-row-(number~0)-of-the-array.\\
3789   If-you-go-on,~this-command-will-be-ignored.
3790 }
3791 \@@_msg_new:nn { in-last-row }
3792 {
3793   You-can't-use-the-command~#1 in-the-last-row-(exterior)-of-the-array.\\
3794   If-you-go-on,~this-command-will-be-ignored.
3795 }

```



```

3796 \@@_msg_new:nn { option-S~without~siunitx }
3797 {
3798   You~can't~use~the~option~'S'~in~your~environment~\@@_full_name_env:
3799   because~you~have~not~loaded~siunitx.\
3800   If~you~go~on,~this~option~will~be~ignored.
3801 }
3802 \@@_msg_new:nn { bad-option-for~line-style }
3803 {
3804   Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line-style'~
3805   is~'standard'.~If~you~go~on,~this~option~will~be~ignored.
3806 }
3807 \@@_msg_new:nn { Unknown-option-for~xdots }
3808 {
3809   As~for~now~there~is~only~three~options~available~here:~'color',~'line-style'~
3810   and~'shorten'~(and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
3811   this~option~will~be~ignored.
3812 }
3813 \@@_msg_new:nn { ampersand~in~light-syntax }
3814 {
3815   You~can't~use~an~ampersand~(\token_to_str &)~to~separate~columns~because
3816   ~you~have~used~the~option~'light-syntax'.~This~error~is~fatal.
3817 }
3818 \@@_msg_new:nn { double-backslash~in~light-syntax }
3819 {
3820   You~can't~use~\token_to_str:N \~to~separate~rows~because~you~have~used~
3821   the~option~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
3822   (set~by~the~option~'end-of-row').~This~error~is~fatal.
3823 }
3824 \@@_msg_new:nn { standard-cline~in~document }
3825 {
3826   The~key~'standard-cline'~is~available~only~in~the~preamble.\
3827   If~you~go~on~this~command~will~be~ignored.
3828 }
3829 \@@_msg_new:nn { bad-value~for~baseline }
3830 {
3831   The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
3832   valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
3833   \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\
3834   If~you~go~on,~a~value~of~1~will~be~used.
3835 }
3836 \@@_msg_new:nn { empty~environment }
3837 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }
3838 \@@_msg_new:nn { unknown~cell~for~line~in~code~after }
3839 {
3840   Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
3841   can't~be~executed~because~a~cell~doesn't~exist.\
3842   If~you~go~on~this~command~will~be~ignored.
3843 }
3844 \@@_msg_new:nn { last-col~non-empty~for~NiceArray }
3845 {
3846   In~the~\@@_full_name_env:,~you~must~use~the~option~
3847   'last-col'~without~value.\
3848   However,~you~can~go~on~for~this~time~
3849   (the~value~'\l_keys_value_tl'~will~be~ignored).
3850 }
3851 \@@_msg_new:nn { last-col~non-empty~for~NiceMatrixOptions }
3852 {
3853   In~\NiceMatrixoptions,~you~must~use~the~option~
3854   'last-col'~without~value.\

```

```

3855     However,~you~can~go~on~for~this~time~
3856     (the~value~'\l_keys_value_tl'~will~be~ignored).
3857 }

3858 \@@_msg_new:nn { Block-too-large }
3859 {
3860     You~try~to~draw~a~block~in~the~cell~#1~#2~of~your~matrix~but~the~matrix~is~
3861     too~small~for~that~block. \\
3862     If~you~go~on,~this~command~will~be~ignored.
3863 }

3864 \@@_msg_new:nn { Wrong-last-row }
3865 {
3866     You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
3867     \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
3868     If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
3869     last-row.~You~can~avoid~this~problem~by~using~'last-row'~
3870     without~value~(more~compilations~might~be~necessary).
3871 }

3872 \@@_msg_new:nn { Yet-in-env }
3873 {
3874     Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~nested.\\
3875     This~error~is~fatal.
3876 }

3877 \@@_msg_new:nn { Outside-math-mode }
3878 {
3879     The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
3880     (and~not~in~\token_to_str:N \vcenter).\\
3881     This~error~is~fatal.
3882 }

3883 \@@_msg_new:nn { Bad-value-for-letter-for-dotted-lines }
3884 {
3885     The~value~of~key~'\tl_use:N \l_keys_key_str'~must~be~of~length~1.\\
3886     If~you~go~on,~it~will~be~ignored.
3887 }

3888 \@@_msg_new:nnn { Unknown-key-for-NiceMatrixOptions }
3889 {
3890     The~key~'\tl_use:N \l_keys_key_str'~is~unknown~for~the~command~
3891     \token_to_str:N \NiceMatrixOptions. \\
3892     If~you~go~on,~it~will~be~ignored. \\
3893     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
3894 }
3895 {
3896     The~available~options~are~(in~alphabetic~order):~
3897     allow-duplicate-names,~
3898     code-for-first-col,~
3899     cell-space-bottom-limit,~
3900     cell-space-top-limit,~
3901     code-for-first-row,~
3902     code-for-last-col,~
3903     code-for-last-row,~
3904     create-extra-nodes,~
3905     create-medium-nodes,~
3906     create-large-nodes,~
3907     end-of-row,~
3908     first-col,~
3909     first-row,~
3910     hlines,~
3911     hvlines,~
3912     last-col,~
3913     last-row,~
3914     left-margin,~
3915     letter-for-dotted-lines,~

```

```

3916     light-syntax,~
3917     nullify-dots,~
3918     renew-dots,~
3919     renew-matrix,~
3920     right-margin,~
3921     small,~
3922     transparent,~
3923     vl原因,~
3924     xdots/color,~
3925     xdots/shorten~and~
3926     xdots/line-style.
3927 }

3928 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
3929 {
3930     The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~environment~
3931     \{NiceArray\}. \\
3932     If~you~go~on,~it~will~be~ignored. \\
3933     For~a~list~of~the~*principal*~available~options,~type~H~<return>.
3934 }
3935 {
3936     The~available~options~are~(in~alphabetic~order):~
3937     b,~
3938     baseline,~
3939     c,~
3940     cell-space-bottom-limit,~
3941     cell-space-top-limit,~
3942     code-after,~
3943     code-for-first-col,~
3944     code-for-first-row,~
3945     code-for-last-col,~
3946     code-for-last-row,~
3947     columns-width,~
3948     create-extra-nodes,~
3949     create-medium-nodes,~
3950     create-large-nodes,~
3951     extra-left-margin,~
3952     extra-right-margin,~
3953     first-col,~
3954     first-row,~
3955     hlines,~
3956     hvlines,~
3957     last-col,~
3958     last-row,~
3959     left-margin,~
3960     light-syntax,~
3961     name,~
3962     nullify-dots,~
3963     renew-dots,~
3964     right-margin,~
3965     rules/color,~
3966     rules/width,~
3967     small,~
3968     t,~
3969     vl原因,~
3970     xdots/color,~
3971     xdots/shorten~and~
3972     xdots/line-style.
3973 }

```

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).

```

3974 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
3975 {

```

```

3976 The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~
3977 \@@_full_name_env:. \\\
3978 If~you~go~on,~it~will~be~ignored. \\\
3979 For~a~list~of~the~*principal*~available~options,~type~H~<return>.
3980 }
3981 {
3982 The~available~options~are~(in~alphabetic~order):~
3983 b,~
3984 baseline,~
3985 c,~
3986 cell-space-bottom-limit,~
3987 cell-space-top-limit,~
3988 code-after,~
3989 code-for-first-col,~
3990 code-for-first-row,~
3991 code-for-last-col,~
3992 code-for-last-row,~
3993 columns-width,~
3994 create-extra-nodes,~
3995 create-medium-nodes,~
3996 create-large-nodes,~
3997 extra-left-margin,~
3998 extra-right-margin,~
3999 first-col,~
4000 first-row,~
4001 hlines,~
4002 hvlines,~
4003 l~(=L),~
4004 last-col,~
4005 last-row,~
4006 left-margin,~
4007 light-syntax,~
4008 name,~
4009 nullify-dots,~
4010 r~(=R),~
4011 renew-dots,~
4012 right-margin,~
4013 rules/color,~
4014 rules/width,~
4015 S,~
4016 small,~
4017 t,~
4018 vlines,~
4019 xdots/color,~
4020 xdots/shorten~and~
4021 xdots/line-style.
4022 }
4023 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
4024 {
4025 The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~environment~
4026 \{NiceTabular\}. \\\
4027 If~you~go~on,~it~will~be~ignored. \\\
4028 For~a~list~of~the~*principal*~available~options,~type~H~<return>.
4029 }
4030 {
4031 The~available~options~are~(in~alphabetic~order):~
4032 b,~
4033 baseline,~
4034 c,~
4035 cell-space-bottom-limit,~
4036 cell-space-top-limit,~
4037 code-after,~
4038 code-for-first-col,~

```

```

4039 code-for-first-row,~
4040 code-for-last-col,~
4041 code-for-last-row,~
4042 columns-width,~
4043 create-extra-nodes,~
4044 create-medium-nodes,~
4045 create-large-nodes,~
4046 extra-left-margin,~
4047 extra-right-margin,~
4048 first-col,~
4049 first-row,~
4050 hlines,~
4051 hvlines,~
4052 last-col,~
4053 last-row,~
4054 left-margin,~
4055 light-syntax,~
4056 name,~
4057 nullify-dots,~
4058 renew-dots,~
4059 right-margin,~
4060 rules/color,~
4061 rules/width,~
4062 t,~
4063 vlines,~
4064 xdots/color,~
4065 xdots/shorten-and~
4066 xdots/line-style.
4067 }

4068 \@@_msg_new:nnn { Duplicate-name }
4069 {
4070   The-name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
4071   the~same~environment~name~twice.~You~can~go~on,~but,~
4072   maybe,~you~will~have~incorrect~results~especially~
4073   if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
4074   message~again,~use~the~option~'allow-duplicate-names'.\\
4075   For~a~list~of~the~names~already~used,~type~H~<return>. \\
4076 }
4077 {
4078   The~names~already~defined~in~this~document~are:~
4079   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
4080 }

4081 \@@_msg_new:nn { Option-auto-for-columns-width }
4082 {
4083   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
4084   If~you~go~on,~the~option~will~be~ignored.
4085 }

4086 \@@_msg_new:nn { Zero~row }
4087 {
4088   There~is~a~problem.~Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~
4089   and~R~in~the~preamble~of~your~environment. \\
4090   This~error~is~fatal.
4091 }

```

## 16 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>41</sup>, Tikz externalization is now deactivated in the environments of the package `nicematrix`.<sup>42</sup>

---

<sup>41</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>42</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 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\left( \begin{array}{ccc} & C_j & \\ 0 & \vdots & 0 \\ & a \cdots \cdots & \\ 0 & & 0 \end{array} \right)_{L_i}$$

## Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See [www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)

## Changes between version 2.1.4 and 2.1.5

Compatibility with the classes `revtex4-1` and `revtex4-2`.

Option `allow-duplicate-names`.

## Changes between version 2.1.5 and 2.2

Possibility to draw horizontal dotted lines to separate rows with the command `\hdottedline` (similar to the classical command `\hline` and the command `\hdashline` of `arydshln`).

Possibility to draw vertical dotted lines to separate columns with the specifier “:” in the preamble (similar to the classical specifier “|” and the specifier “:” of `arydshln`).

## Changes between version 2.2 and 2.2.1

Improvement of the vertical dotted lines drawn by the specifier “:” in the preamble.

Modification of the position of the dotted lines drawn by `\hdottedline`.

## Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.

Option `hlines`.

## Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of “|”) as `\hdotsfor` does.

Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

## 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>43</sup>, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

## Changes between version 3.4 and 3.5

Correction on a bug on the two previous versions where the `code-after` was not executed.

## Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate-name` in the environments of `amsmath`.

## Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

---

<sup>43</sup>cf. [tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize](https://tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize)



## Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`. An error is raised when an obsolete environment is used.

## Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.  
New options `create-medium-nodes` and `create-large-nodes`.

## Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).  
New option `dotted-lines-margin` for fine tuning of the dotted lines.

## Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

## Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.  
Options `vlines`, `hlines` and `hvlines`.  
Option `baseline` pour `{NiceArray}` (not for the other environments).  
The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell  $i-j$ , the name is  $i-j$ -block and, if the creation of the “medium nodes” is required, a node  $i-j$ -block-medium is created.  
If the user 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](https://stackoverflow.com)).  
Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.

## Changes between versions 3.14 and 3.15

It's possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the code-after with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New command `\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 `hylvline` don't draw rules in the blocks (commands `\Block`) and in the virtual blocks corresponding to the dotted lines.

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols		
@@ commands:		\@@_Vdots . . . . . 809, 827, 2493
\@@_Block: . . . . .	817, 3490	\g_@@_Vdots_lines_tl . . . . . 899, 1800
\@@_Block_i . . . . .	3491, 3492	\@@_Vdotsfor: . . . . . 815, 2665
\@@_Block_ii:nmnnn . . . . .	3492, 3493	\@@_Vdotsfor:nmnn . . . . . 2669, 2680
\@@_Block_iii:nmnnnn . . . . .	3516, 3520	\@@_actually_diagbox:nmnn . . . . . 3657, 3661
\@@_Cdots . . . . .	808, 826, 2478	\@@_actually_draw_Cdots: . . . . . 2055, 2059
\g_@@_Cdots_lines_tl . . . . .	897, 1803	\@@_actually_draw_Ddots: . . . . . 2182, 2186
\@@_Cell: 177, 554, 744, 839, 840, 841, 849, 863		\@@_actually_draw_Iddots: . . . . . 2233, 2237
\@@_CodeAfter:n . . . . .	821, 3692, 3704	\@@_actually_draw_Ldots: . 2011, 2015, 2656
\@@_CodeAfter_i:n . . . . .	3695, 3697	\@@_actually_draw_Vdots: . 2109, 2113, 2731
\@@_Ddots . . . . .	810, 828, 2508	\@@_adapt_S_column: . . . . . 152, 167, 911
\g_@@_Ddots_lines_tl . . . . .	900, 1801	\@@_after_array: . . . . . 1219, 1658
\g_@@_HVdotsfor_lines_tl . . . . .		\@@_analyze_end:Nn . . . . . 1299, 1344
. . . . .	902, 1799, 2591, 2667, 3751	\l_@@_argspec_tl . . . . . 2461,
\@@_Hdotsfor: . . . . .	814, 831, 2580	2462, 2463, 2478, 2493, 2508, 2530, 2587,
\@@_Hdotsfor:nmnn . . . . .	2593, 2605	2588, 2589, 2663, 2664, 2665, 2761, 2762, 2763
\@@_Hdotsfor_i . . . . .	2583, 2589	\@@_array: . . . . . 673, 1300, 1327
\@@_Hspace: . . . . .	813, 2553	\l_@@_auto_columns_width_bool . . . . .
\@@_Iddots . . . . .	811, 829, 2530	. . . . . 315, 424, 1408, 1412, 3276
\g_@@_Iddots_lines_tl . . . . .	901, 1802	\l_@@_baseline_str . . . . .
\@@_Ldots . . . . .	807, 825, 830, 2463	. . . . . 306, 307, 417, 418, 419, 420, 684,
\g_@@_Ldots_lines_tl . . . . .	898, 1804	1128, 1144, 1147, 1148, 1149, 1225, 1231, 1236
\l_@@_NiceArray_bool . . . . .		\@@_begin_of_NiceMatrix:nn . . . . 1627, 1631
. . . . .	194, 930, 1041, 1059, 1071,	\@@_begin_of_row: . . . . . 558, 582, 1494
. . . . .	1126, 1574, 3026, 3027, 3049, 3050, 3079, 3637	\l_@@_block_auto_columns_width_bool . .
\g_@@_NiceMatrixBlock_int . . . . .		. . . . . 925, 1413, 3269, 3274, 3284, 3294
. . . . .	190, 3281, 3286, 3289, 3300	\g_@@_blocks_seq . 230, 928, 1762, 3503, 3516
\l_@@_NiceTabular_bool . . . . .		\c_@@_booktabs_loaded_bool . . . 29, 34, 754
. . . . .	138, 195, 561, 771, 912, 985,	\l_@@_cell_box . . . . . 560,
. . . . .	987, 1060, 1072, 1080, 1197, 1201, 1654,	606, 608, 614, 622, 623, 624, 625, 627, 630,
. . . . .	2022, 2032, 2066, 2074, 2738, 3220, 3227, 3645	632, 634, 651, 756, 848, 856, 862, 871, 996,
\@@_OnlyMainNiceMatrix:n . .	819, 2981, 3000	998, 1495, 1517, 1520, 1522, 1537, 1558,
\@@_OnlyMainNiceMatrix_i:n	2984, 2991, 2994	1562, 2746, 2749, 2753, 3529, 3560, 3565, 3652

\l_@@_cell_space_bottom_limit_dim ...	295, 359, 625
\l_@@_cell_space_top_limit_dim	294, 357, 623
\@@_cellcolor	979, 2889, 2974, 2975
\g_@@_cells_seq	1338, 1339, 1340, 1342
\@@_chessboardcolors	984, 2967
\@@_cline	120, 806
\@@_cline_i:nn	121, 122, 130
\@@_cline_i:w	122, 123
\g_@@_code_after_tl	213, 437, 1321, 1779, 1780, 3694, 3703
\l_@@_code_before_bool	219, 414, 691, 936, 1358, 1372, 1390, 1421, 1447, 1470, 1704
\l_@@_code_before_tl	413, 986
\l_@@_code_for_first_col_tl	371, 1506
\l_@@_code_for_first_row_tl	375, 570
\l_@@_code_for_last_col_tl	373, 1546
\l_@@_code_for_last_row_tl	377, 577
\g_@@_col_total_int	559, 837, 1117, 1440, 1441, 1472, 1477, 1482, 1483, 1536, 1662, 1665, 1670, 1677, 1721, 2146, 2726, 3317, 3327, 3361, 3448
\c_@@_colortbl_loaded_bool	84, 88, 773
\@@_columncolor	983, 2853
\l_@@_columns_width_dim	191, 425, 491, 1409, 1415, 3282, 3288
\g_@@_com_or_env_str	205, 206, 209
\@@_computations_for_large_nodes:	3388, 3401, 3406
\@@_computations_for_medium_nodes:	3308, 3377, 3387, 3398
\@@_convert_to_str_seq:N	3717, 3729
\@@_create_col_nodes:	1303, 1331, 1350
\@@_create_large_nodes:	1741, 3382
\@@_create_medium_and_large_nodes:	1738, 3393
\@@_create_medium_nodes:	1739, 3372
\@@_create_nodes:	3379, 3390, 3400, 3403, 3444
\@@_create_row_node:	687, 719, 755
\@@_cut_on_hyphen:w	2812, 2834, 2835, 2868, 2869, 2898, 2929, 2940
\g_@@_ddots_int	1728, 2206, 2207
\@@_define_columntype:nn	739, 842, 843, 844
\@@_define_com:nnn	3621, 3629, 3630, 3631, 3632, 3633
\@@_define_env:n	1620, 1644, 1645, 1646, 1647, 1648, 1649
\g_@@_delta_x_one_dim	1730, 2209, 2219
\g_@@_delta_x_two_dim	1732, 2260, 2270
\g_@@_delta_y_one_dim	1731, 2211, 2219
\g_@@_delta_y_two_dim	1733, 2262, 2270
\@@_diagbox:nn	822, 3653
\@@_dotfill	3641, 3644, 3652
\@@_dotfill:	820, 3642
\@@_dotfill_i:	3647, 3649
\@@_dotfill_ii:	3646, 3649, 3650
\@@_dotfill_iii:	3650, 3651
\@@_double_int_eval:n	2757, 2771, 2772
\g_@@_dp_ante_last_row_dim	585, 789
\g_@@_dp_last_row_dim	585, 586, 792, 793, 997, 998, 1185, 3016
\g_@@_dp_row_zero_dim	605, 606, 783, 784, 1139, 1168, 1178
\@@_draw_Cdots:nnn	2041
\@@_draw_Ddots:nnn	2174
\@@_draw_Iddots:nnn	2225
\@@_draw_Ldots:nnn	1997
\@@_draw_Vdots:nnn	2095
\@@_draw_blocks:	1762, 3515
\@@_draw_dotted_lines:	1754, 1788
\@@_draw_dotted_lines_i:	1791, 1795
\@@_draw_hlines:	1758, 3041, 3072
\@@_draw_hvlines:	1756, 3062
\@@_draw_hvlines_i:	3067, 3070
\@@_draw_hvlines_ii:	3068, 3075
\@@_draw_line:	2039, 2093, 2172, 2223, 2274, 2276, 2810, 3237, 3267
\@@_draw_line_ii:nn	2790, 2794
\@@_draw_line_iii:nn	2797, 2801
\@@_draw_non_standard_dotted_line:	2282, 2284
\@@_draw_non_standard_dotted_line:n	2287, 2290
\@@_draw_standard_dotted_line:	2281, 2310
\@@_draw_standard_dotted_line_i:	2375, 2379
\@@_draw_vlines:	1759, 3001, 3073
\g_@@_empty_cell_bool	226, 629, 636, 2476, 2491, 2506, 2528, 2550, 2555
\@@_end_Cell:	179, 618, 749, 839, 840, 841, 853, 867
\l_@@_end_of_row_tl	323, 324, 365, 1323, 1324, 3821
\c_@@_endpgfortikzpicture_tl	40, 44, 1792, 2798, 3206
\@@_env:	185, 189, 591, 597, 652, 658, 696, 702, 708, 950, 951, 957, 958, 965, 966, 976, 1359, 1362, 1364, 1377, 1383, 1386, 1395, 1401, 1404, 1426, 1432, 1435, 1452, 1458, 1464, 1472, 1477, 1483, 1773, 1869, 1937, 1979, 1990, 2619, 2637, 2694, 2712, 2783, 2785, 2804, 2807, 3330, 3332, 3340, 3451, 3460, 3478, 3544, 3550, 3551, 3552
\g_@@_env_int	184, 185, 187, 924, 938, 942, 945, 954, 955, 962, 963, 1006, 1009, 1024, 1027, 1669, 1690, 1708, 1711, 3487
\@@_error:n	20, 335, 344, 479, 490, 499, 502, 522, 525, 532, 534, 540, 545, 550, 552, 1112, 1156, 1243
\@@_error:nn	21, 432, 2466, 2469, 2481, 2484, 2496, 2499, 2512, 2513, 2518, 2519, 2534, 2535, 2540, 2541
\@@_error:nnn	22, 2788
\@@_error_too_much_cols:	1078, 3736
\@@_everycr:	713, 778, 781
\@@_everycr_i:	713, 714
\l_@@_exterior_arraycolsep_bool	308, 487, 1062, 1074
\l_@@_extra_left_margin_dim	321, 401, 1091, 1525
\l_@@_extra_right_margin_dim	322, 402, 1103, 1566, 2149
\@@_extract_coords_values:	3469, 3476
\@@_fatal:n	23, 199, 915, 1308, 1312, 1314, 1347, 1353, 3741, 3744, 3747
\@@_fatal:nn	24

<code>\l_@@_final_i_int</code> .....	<code>\g_@@_internal_code_after_tl</code> .....
..... 1744, 1816, 1821, 1824, 1849,	..... 214, 888, 1763, 1764, 3195, 3655
1857, 1861, 1870, 1878, 1960, 1991, 2033,	<code>\@@_j:</code> .....
2131, 2198, 2249, 2610, 2638, 2706, 2716, 2718	3317, 3319,
<code>\l_@@_final_j_int</code> .....	3320, 3321, 3322, 3327, 3330, 3332, 3335,
1745, 1817, 1822, 1829, 1834, 1840, 1850,	3337, 3338, 3340, 3343, 3345, 3346, 3361,
1858, 1862, 1871, 1879, 1961, 1992, 2029,	3364, 3366, 3367, 3368, 3423, 3425, 3428,
2071, 2200, 2251, 2631, 2641, 2643, 2685, 2714	3430, 3434, 3435, 3448, 3451, 3452, 3454,
<code>\l_@@_final_open_bool</code> .....	3459, 3460, 3472, 3478, 3479, 3481, 3486, 3487
1747, 1823,	<code>\l_@@_l_dim</code> .....
1827, 1830, 1837, 1843, 1847, 1863, 2027,	..... 2359, 2360, 2373, 2374, 2386, 2392,
2069, 2079, 2090, 2116, 2129, 2137, 2158,	2403, 2411, 2419, 2424, 2436, 2437, 2444, 2445
2196, 2247, 2383, 2398, 2429, 2430, 2608,	<code>\l_@@_large_nodes_bool</code> 318, 392, 1737, 1741
2632, 2644, 2683, 2707, 2719, 2780, 3203, 3242	<code>\g_@@_last_col_found_bool</code> .....
<code>\@@_find_extremities_of_line:nnnn</code> ...	896, 1118, 1214, 1439, 1468, 1534, 1661, 1718
..... 1811, 2001, 2045, 2100, 2178, 2229	<code>\l_@@_last_col_int</code> .....
<code>\l_@@_first_col_int</code> .....	..... 241, 242, 480, 513, 515, 533, 541,
112, 121,	551, 948, 1020, 1026, 1033, 1066, 1637,
235, 236, 367, 538, 558, 1054, 1121, 1354,	1639, 1662, 1665, 1717, 2105, 2144, 2468,
1370, 1715, 2825, 2876, 2908, 2937, 2983,	2483, 2519, 2541, 3605, 3611, 3617, 3740, 3758
3317, 3327, 3361, 3409, 3448, 3603, 3609, 3615	<code>\l_@@_last_col_without_value_bool</code> ...
<code>\l_@@_first_row_int</code> .....	..... 240, 512, 1663, 3743
233, 234, 368,	<code>\l_@@_last_row_int</code> ..
542, 835, 1136, 1152, 1165, 1176, 1239,	237, 238, 369, 575,
1713, 3310, 3324, 3351, 3408, 3446, 3601, 3832	724, 882, 946, 991, 1001, 1008, 1015, 1106,
<code>\@@_full_name_env:</code> .....	1110, 1113, 1120, 1182, 1325, 1326, 1502,
207, 3757,	1503, 1543, 1544, 1684, 2006, 2050, 2498,
3764, 3772, 3798, 3837, 3846, 3867, 3879, 3977	2513, 2535, 2747, 2989, 2997, 3014, 3613, 3866
<code>\@@_hdottedline:</code> .....	<code>\l_@@_last_row_without_value_bool</code> ...
812, 3188	..... 239, 1003, 1108, 1682
<code>\@@_hdottedline:n</code> .....	<code>\g_@@_last_vdotted_col_int</code> 885, 887, 893, 895
3196, 3200	<code>\l_@@_left_delim_dim</code> .....
<code>\@@_hdottedline_i:</code> .....	..... 1039, 1043, 1048, 1290, 1523
3191, 3193	<code>\l_@@_left_delim_tl</code> .....
<code>\@@_hdottedline_i:n</code> .....	906, 3231
3205, 3209	<code>\l_@@_left_margin_dim</code> .....
<code>\l_@@_hlines_bool</code> ..	..... 319, 395, 1090, 1524, 3221, 3439
311, 379, 385, 720, 1758	<code>\l_@@_letter_for_dotted_lines_str</code> ...
<code>\g_@@_ht_last_row_dim</code> .....	..... 498, 504, 505, 875, 876
..... 587, 790, 791, 995, 996, 1184, 3017	<code>\l_@@_light_syntax_bool</code> .....
<code>\g_@@_ht_row_one_dim</code> ....	..... 305, 363, 1093, 1098, 1685
613, 614, 787, 788	<code>\@@_light_syntax_i</code> .....
<code>\g_@@_ht_row_zero_dim</code> .....	1316, 1319
..... 607, 608, 785, 786, 1139, 1168, 1179	<code>\@@_line</code> .....
<code>\l_@@_hvlines_bool</code> ...	1777, 2763
313, 383, 1755, 1954	<code>\@@_line_i:nn</code> .....
<code>\@@_i:</code> .....	2770, 2777
3310, 3312,	<code>\@@_line_with_light_syntax:n</code> ...
3313, 3314, 3315, 3324, 3330, 3332, 3333,	1330, 1334
3334, 3335, 3340, 3341, 3342, 3343, 3351,	<code>\@@_line_with_light_syntax_i:n</code> .....
3354, 3356, 3357, 3358, 3410, 3412, 3415,	..... 1329, 1335, 1336
3416, 3420, 3421, 3446, 3451, 3453, 3455,	<code>\@@_math_toggle_token:</code> .....
3459, 3460, 3471, 3478, 3480, 3482, 3486, 3487	137,
<code>\g_@@_iddots_int</code> .....	620, 1496, 1513, 1538, 1554, 3507, 3685, 3689
1729, 2257, 2258	<code>\g_@@_max_cell_width_dim</code> .....
<code>\l_@@_in_env_bool</code> .....	..... 626, 627, 926, 1414, 3275, 3301
193, 915, 916	<code>\l_@@_max_delimiter_width_bool</code> .....
<code>\l_@@_initial_i_int</code> .....	..... 326, 362, 1210
1742,	<code>\c_@@_max_l_dim</code> .....
1814, 1889, 1892, 1917, 1925, 1929, 1938,	2373, 2378
1946, 1958, 1980, 2023, 2081, 2083, 2125,	<code>\l_@@_medium_nodes_bool</code> 317, 391, 1735, 3547
2190, 2241, 2609, 2610, 2620, 2688, 2698, 2700	<code>\@@_message_hdotsfor:</code> 3749, 3757, 3764, 3772
<code>\l_@@_initial_j_int</code> .....	<code>\@@_msg_new:nn</code> ....
..... 1743, 1815, 1890, 1897,	25, 3754, 3761, 3769,
1902, 1908, 1918, 1926, 1930, 1939, 1947,	3776, 3781, 3786, 3791, 3796, 3802, 3807,
1959, 1981, 2019, 2063, 2139, 2141, 2146,	3813, 3818, 3824, 3829, 3836, 3838, 3844,
2192, 2243, 2613, 2623, 2625, 2684, 2685, 2696	3851, 3858, 3864, 3872, 3877, 3883, 4081, 4086
<code>\l_@@_initial_open_bool</code> .....	<code>\@@_msg_new:nnn</code> .....
..... 1746, 1891, 1895, 1898, 1905, 1911,	..... 26, 3888, 3928, 3974, 4023, 4068
1915, 1931, 2017, 2061, 2078, 2088, 2116,	<code>\@@_msg_redirect_name:nn</code> .....
2123, 2135, 2188, 2239, 2381, 2428, 2607,	27, 493
2614, 2626, 2682, 2689, 2701, 2779, 3202, 3241	<code>\@@_multicolumn:nnn</code> .....
<code>\@@_instruction_of_type:nn</code> .....	816, 2559
..... 662, 2471, 2486, 2501, 2522, 2544	<code>\g_@@_multicolumn_cells_seq</code> .....
<code>\l_@@_inter_dots_dim</code> .....	..... 833, 2566, 3335, 3343, 3465
..... 296, 297, 1751, 2386, 2393, 2404, 2412,	
2419, 2424, 2436, 2444, 3232, 3235, 3263, 3265	

<code>\g_@@multicolumn_sizes_seq</code>	834, 2568, 3466	<code>\l_@@radius_dim</code>	300, 301, 883,
<code>\g_@@name_env_str</code>	204, 210, 211, 909, 910, 1346, 1575, 1576,		1750, 2037, 2038, 2453, 3190, 3214, 3260, 3261
	1582, 1583, 1590, 1591, 1598, 1599, 1606,	<code>\l_@@real_left_delim_dim</code>	1261, 1276, 1291
	1607, 1614, 1615, 1624, 1652, 1782, 3625, 3738	<code>\l_@@real_right_delim_dim</code>	1262, 1288, 1294
<code>\l_@@name_str</code>	316, 434, 593, 596,	<code>\@@rectanglecolor</code>	980, 2922
	654, 657, 704, 707, 1004, 1013, 1016, 1022,	<code>\@@renew_NC@rewrite@S:</code>	170, 894
	1031, 1034, 1363, 1364, 1385, 1386, 1403,	<code>\l_@@renew_dots_bool</code>	388, 823, 3709
	1404, 1434, 1435, 1460, 1463, 1479, 1482,	<code>\@@renew_matrix:</code>	483, 3571, 3711
	1672, 1676, 1693, 1697, 3456, 3459, 3483, 3486	<code>\@@restore_iRow_jCol:</code>	1783, 1806
<code>\g_@@names_seq</code>	192, 431, 433, 4079	<code>\c_@@revtex_bool</code>	47, 49, 52, 675
<code>\l_@@nb_cols_int</code>	3592, 3597, 3600, 3604, 3610, 3616	<code>\l_@@right_delim_dim</code>	1040, 1044, 1050, 1293, 1564
<code>\l_@@nb_rows_int</code>	3591, 3596, 3607	<code>\l_@@right_delim_tl</code>	907, 3234
<code>\@@newcolumnmtype</code>	730, 741, 839, 840, 841, 845, 859	<code>\l_@@right_margin_dim</code>	320, 397, 1102, 1565, 2148, 3228, 3442
<code>\@@node_for_multicolumn:nn</code>	3467, 3474	<code>\@@rotate:</code>	818, 2736
<code>\@@node_for_the_cell:</code>	633, 638, 1521, 1567	<code>\@@rotate_i:</code>	2740, 2742
<code>\@@node_position:</code>	957, 959, 965, 967	<code>\@@rotate_ii:</code>	2739, 2742, 2743
<code>\l_@@nullify_dots_bool</code>	314, 390, 2475, 2490, 2505, 2527, 2549	<code>\@@rotate_iii:</code>	2743, 2744
<code>\@@old_CT@arc@</code>	917, 1784	<code>\g_@@row_of_col_done_bool</code>	218, 717, 908, 1369
<code>\@@old_arraycolsep_dim</code>	227, 772, 1082	<code>\g_@@row_total_int</code>	836, 1119, 1153, 1240, 1684, 1691, 1698,
<code>\@@old_cdots</code>	798, 2490		1714, 2651, 3020, 3310, 3324, 3351, 3446, 3833
<code>\@@old_ddots</code>	800, 2527	<code>\@@rowcolor</code>	981, 2817, 2962, 2963
<code>\l_@@old_iRow_int</code>	215, 758, 1808	<code>\@@rowcolors</code>	982, 2957
<code>\@@old_ialign:</code>	686, 794, 1761	<code>\g_@@rows_seq</code>	1322, 1324, 1326, 1328, 1330
<code>\@@old_iddots</code>	801, 2549	<code>\l_@@rules_color_tl</code>	217, 348, 934, 935
<code>\l_@@old_jCol_int</code>	216, 761, 1809	<code>\@@set_CT@arc@:</code>	139, 935
<code>\@@old_ldots</code>	797, 2475	<code>\@@set_CT@arc@_i:</code>	140, 141
<code>\@@old_multicolumn</code>	2558, 2561	<code>\@@set_CT@arc@_ii:</code>	140, 143
<code>\@@old_pgful@check@rerun</code>	76, 81	<code>\@@set_final_coords:</code>	1970, 1995
<code>\@@old_vdots</code>	799, 2505	<code>\@@set_final_coords_from_anchor:n</code>	1986, 2036, 2076, 2119, 2134, 2203, 2254
<code>\l_@@parallelize_diags_bool</code>	309, 310, 387, 1726, 2204, 2255	<code>\@@set_initial_coords:</code>	1965, 1984
<code>\@@pgf_rect_node:nnn</code>	269, 3549	<code>\@@set_initial_coords_from_anchor:n</code>	1975, 2026, 2068, 2118, 2128, 2195, 2246
<code>\@@pgf_rect_node:nnnnn</code>	244, 3450, 3477, 3543	<code>\@@set_seq_of_str_from_clist:Nn</code>	3726, 3731
<code>\c_@@pgfortikzpicture_tl</code>	39, 43, 1790, 2796, 3204	<code>\@@set_size:n</code>	3589, 3598
<code>\@@picture_position:</code>	951, 959, 967	<code>\c_@@siunitx_loaded_bool</code>	145, 149, 154, 520, 894
<code>\g_@@pos_of_blocks_seq</code>	231, 929, 2569, 3065, 3104, 3133, 3502	<code>\l_@@small_bool</code>	481,
<code>\g_@@pos_of_xdots_seq</code>	232, 1956, 3066, 3106, 3135, 3158		523, 529, 543, 564, 764, 1497, 1539, 1748
<code>\@@pre_array:</code>	752, 1038	<code>\@@standard_cline</code>	109, 805
<code>\c_@@preamble_first_col_tl</code>	1055, 1490	<code>\@@standard_cline:w</code>	109, 110
<code>\c_@@preamble_last_col_tl</code>	1067, 1530	<code>\l_@@standard_cline_bool</code>	293, 355, 804
<code>\@@pred:n</code>	113, 136, 1639	<code>\c_@@standard_tl</code>	303, 304, 2280, 3236, 3266
<code>\@@put_box_in_flow:</code>	1212, 1221, 1292	<code>\l_@@stop_loop_bool</code>	1818, 1819,
<code>\@@put_box_in_flow_bis:nn</code>	1211, 1259		1851, 1864, 1873, 1886, 1887, 1919, 1932, 1941
<code>\@@put_box_in_flow_i:</code>	1227, 1229	<code>\@@succ:n</code>	130,
<code>\@@qpoint:n</code>	188, 1131, 1133, 1160, 1162,		135, 696, 702, 1249, 1452, 1458, 1463,
	1247, 1249, 1252, 2019, 2023, 2029, 2033,		1464, 1472, 1477, 1482, 1483, 1719, 2029,
	2063, 2071, 2081, 2083, 2125, 2131, 2139,		2071, 2083, 2131, 2141, 2198, 2200, 2243,
	2141, 2190, 2192, 2198, 2200, 2241, 2243,		2249, 2828, 2841, 2862, 2879, 2905, 2911,
	2249, 2251, 2804, 2807, 2824, 2828, 2841,		2945, 2947, 3011, 3018, 3020, 3027, 3050,
	2843, 2860, 2862, 2875, 2879, 2899, 2905,		3055, 3090, 3092, 3119, 3148, 3223, 3264,
	2907, 2911, 2934, 2936, 2945, 2947, 3009,		3416, 3420, 3430, 3434, 3537, 3539, 3671, 3673
	3011, 3018, 3020, 3029, 3033, 3052, 3055,	<code>\l_@@suffix_tl</code>	3378, 3389,
	3086, 3088, 3090, 3092, 3115, 3117, 3119,		3399, 3402, 3451, 3459, 3460, 3478, 3486, 3487
	3144, 3146, 3148, 3212, 3216, 3223, 3259,	<code>\c_@@table_collect_begin_tl</code>	162, 164, 177
	3262, 3264, 3356, 3366, 3533, 3535, 3537,	<code>\c_@@table_print_tl</code>	165, 166, 179
	3539, 3556, 3557, 3563, 3666, 3668, 3671, 3673	<code>\@@test_if_hline_in_block:nnnn</code>	3105, 3107, 3160

\@@_test_if_math_mode: . . . . .	196, 914, 1584, 1592, 1600, 1608, 1616
\@@_test_if_vline_in_block:nnnn . . . . .	3134, 3136, 3174
\l_@@_the_array_box . . . . .	1052, 1079, 1141, 1145, 1171, 1200
\c_@@_tikz_loaded_bool . . . . .	30, 38, 971, 1765, 3243
\l_@@_tikz_tl . . . . .	3512, 3542
\l_@@_type_of_col_tl . . . . .	516, 517, 518, 519, 521, 1625, 1627
\c_@@_types_of_matrix_seq . . . . .	3731, 3738
\@@_update_for_first_and_last_row: . . . . .	601, 628, 993, 1515, 1556
\@@_vdottedline:n . . . . .	889, 3239
\@@_vdottedline_i:n . . . . .	3246, 3251, 3255
\@@_vline: . . . . .	927, 2999
\l_@@_vlines_bool . . . . .	312, 380, 384, 1061, 1073, 1085, 1104, 1759
\g_@@_width_first_col_dim . . . . .	229, 1124, 1516, 1517
\g_@@_width_last_col_dim . . . . .	228, 1216, 1557, 1558
\l_@@_x_final_dim . . . . .	222, 1972, 2030, 2031, 2072, 2073, 2121, 2143, 2151, 2155, 2159, 2161, 2166, 2168, 2201, 2210, 2218, 2252, 2261, 2269, 2307, 2321, 2330, 2366, 2418, 2434, 2808, 3224, 3235, 3261
\l_@@_x_initial_dim . . . . .	220, 1967, 2020, 2021, 2064, 2065, 2121, 2142, 2143, 2150, 2155, 2159, 2161, 2163, 2166, 2168, 2193, 2210, 2218, 2244, 2261, 2269, 2298, 2320, 2330, 2366, 2418, 2432, 2434, 2452, 2454, 2805, 3217, 3232, 3260
\l_@@_xdots_color_tl . . . . .	325, 338, 2010, 2054, 2097, 2181, 2232, 2288, 2655, 2730, 2767
\l_@@_xdots_down_tl . . . . .	342, 2304, 2314, 2349
\l_@@_xdots_line_style_tl . . . . .	302, 304, 334, 2280, 2288, 3236, 3266
\l_@@_xdots_shorten_dim . . . . .	298, 299, 340, 1752, 2295, 2296, 2392, 2403, 2411
\l_@@_xdots_up_tl . . . . .	343, 2300, 2313, 2339
\l_@@_y_final_dim . . . . .	223, 1973, 2034, 2038, 2085, 2089, 2091, 2132, 2199, 2212, 2215, 2250, 2263, 2266, 2307, 2321, 2329, 2368, 2423, 2442, 2809, 3215, 3265
\l_@@_y_initial_dim . . . . .	221, 1968, 2024, 2037, 2084, 2085, 2089, 2091, 2126, 2191, 2212, 2217, 2242, 2263, 2268, 2298, 2320, 2329, 2368, 2423, 2440, 2442, 2452, 2455, 2806, 3213, 3214, 3215, 3263
\ \ . . . . .	1313, 1335, 3605, 3611, 3617, 3778, 3783, 3788, 3793, 3799, 3820, 3826, 3833, 3841, 3847, 3854, 3861, 3874, 3880, 3885, 3891, 3892, 3931, 3932, 3977, 3978, 4026, 4027, 4074, 4075, 4089
\{ . . . . .	211, 1601, 3633, 3840, 3874, 3931, 4026
\} . . . . .	211, 1601, 3633, 3840, 3874, 3931, 4026
\  . . . . .	1617, 3632
\_ . . . . .	3752, 3757, 3764, 3772, 3832, 3833, 3837, 3867, 3868, 3879
<b>A</b>	
\array . . . . .	683
\arraycolsep . . . . .	396, 398, 400, 767, 772, 1043, 1044, 1082, 1083, 1087, 1123, 1199, 1203, 1217, 2022, 2032, 2066, 2074, 3220, 3227
\arrayrulecolor . . . . .	91
\arrayrulewidth . . . . .	115, 118, 128, 350, 592, 695, 697, 703, 725, 1087, 1088, 1104, 1376, 1378, 1384, 1394, 1396, 1402, 1425, 1427, 1433, 1451, 1453, 1459, 2826, 2827, 2829, 2842, 2844, 2861, 2863, 2877, 2878, 2880, 2904, 2906, 2909, 2910, 2912, 2935, 2938, 2939, 2946, 2948, 3008, 3024, 3047, 3056, 3084, 3113, 3142, 3301
\arraystretch . . . . .	766
\AtBeginDocument . . . . .	31, 68, 85, 146, 1786, 2459, 2585, 2661, 2759, 2792, 3198
\AutoNiceMatrix . . . . .	3634
\AutoNiceMatrixWithDelims . . . . .	3594, 3626, 3638
<b>B</b>	
\baselineskip . . . . .	94
\Block . . . . .	817
\BNiceMatrix . . . . .	3586
\bNiceMatrix . . . . .	3583
bool commands:	
\bool_do_until:Nn . . . . .	1819, 1887
\bool_gset_false:N . . . . .	636, 896, 908, 3169, 3183
\bool_gset_true:N . . . . .	1369, 1534, 2476, 2491, 2506, 2528, 2550, 2555, 3103, 3132
\bool_if:NTF . . . . .	138, 154, 561, 564, 691, 717, 720, 754, 764, 771, 823, 894, 915, 925, 936, 971, 985, 987, 1080, 1085, 1104, 1108, 1214, 1358, 1372, 1390, 1408, 1421, 1447, 1468, 1470, 1497, 1539, 1661, 1663, 1682, 1685, 1704, 1726, 1741, 1748, 1758, 1759, 1765, 1954, 2088, 2090, 2204, 2255, 2475, 2490, 2505, 2527, 2549, 3079, 3108, 3137, 3284, 3294, 3547, 3645, 3743
\bool_if:nTF . . . . .	1118, 1150, 1237, 1735, 2781
\bool_lazy_all:nTF . . . . .	1057, 1069
\bool_lazy_and:nnTF . . . . .	1411, 1498, 1716, 2077, 2312, 2900, 2930, 3064
\bool_lazy_or:nnTF . . . . .	331, 1542, 2116, 2372, 3524
\bool_lazy_or_p:nn . . . . .	1501
\bool_not_p:n . . . . .	1060, 1061, 1062, 1072, 1073, 1074, 1413, 1718
\bool_set:Nn . . . . .	2120
\g_tmpa_bool . . . . .	3103, 3108, 3132, 3137, 3169, 3183
box commands:	
\box_clear_new:N . . . . .	756, 1052
\box_dp:N . . . . .	586, 606, 625, 784, 793, 998, 1224, 1270, 1283
\box_ht:N . . . . .	587, 608, 614, 623, 786, 788, 791, 996, 1223, 1270, 1283, 2752
\box_move_up:nn . . . . .	59, 61, 63, 1141, 1170, 1256
\box_rotate:Nn . . . . .	2746
\box_set_dp:Nn . . . . .	624, 1224
\box_set_ht:Nn . . . . .	622, 1223
\box_use:N . . . . .	746, 749, 2753
\box_use_drop:N . . . . .	630, 634, 651, 856, 871, 1141, 1145, 1171, 1200, 1226, 1256, 1257, 1522, 3560, 3565



<code>\box_wd:N</code> . . . . .	627, 632, 1048, 1050, 1277, 1289, 1517, 1520, 1558, 1562, 3652
<code>\l_tmpa_box</code> . . . . .	1047, 1048, 1049, 1050, 1188, 1223, 1224, 1226, 1256, 1257, 1270, 1283
<code>\l_tmpb_box</code> . . . . .	1263, 1277, 1278, 1289
<b>C</b>	
<code>\Cdots</code> . . . . .	808, 2481, 2484
<code>\cdots</code> . . . . .	798, 826
<code>\cellcolor</code> . . . . .	979
<code>\chessboardcolors</code> . . . . .	984
<code>\cline</code> . . . . .	129, 805, 806
clist commands:	
<code>\clist_map_inline:nn</code> . . . . .	2830, 2864, 2896
<code>\CodeAfter</code> . . . . .	821, 1316, 1319, 1778
<code>\color</code> . . . . .	95, 142, 144, 2004, 2007, 2010, 2048, 2051, 2054, 2097, 2103, 2106, 2181, 2232, 2649, 2652, 2655, 2724, 2727, 2730, 2767, 2823, 2859, 2895, 2928
<code>\colorlet</code> . . . . .	202, 203, 571, 578, 1507, 1547
<code>\columncolor</code> . . . . .	983
<code>\cr</code> . . . . .	117, 131, 1488
<code>\crcr</code> . . . . .	1352
cs commands:	
<code>\cs_generate_variant:Nn</code> . . . . .	3306, 3307
<code>\cs_gset:Npn</code> . . . . .	95, 1669, 1676, 1690, 1697, 3299
<code>\cs_gset_eq:NN</code> . . . . .	167, 777, 917, 1784
<code>\cs_if_exist:NTF</code> . . . . .	15, 757, 760, 918, 921, 1006, 1013, 1024, 1031, 1808, 1809, 1854, 1867, 1922, 1935, 2617, 2635, 2692, 2710, 3286, 3329
<code>\cs_if_exist_p:N</code> . . . . .	332
<code>\cs_if_free:NTF</code> . . . . .	732, 1999, 2043, 2098, 2176, 2227
<code>\cs_if_free_p:N</code> . . . . .	2783, 2785
<code>\cs_new_protected:Npx</code> . . . . .	1788, 2794, 3200
<code>\cs_set:Npn</code> . . . . .	91, 92, 97, 109, 110, 120, 122, 123, 142, 144, 680, 734, 766, 769, 1813, 1875, 1943, 2659, 2734, 3471, 3472
<code>\cs_set_protected:Npn</code> . . . . .	3623
<b>D</b>	
<code>\Ddots</code> . . . . .	810, 2512, 2513, 2518, 2519
<code>\ddots</code> . . . . .	800, 828
<code>\diagbox</code> . . . . .	822
dim commands:	
<code>\dim_add:Nn</code> . . . . .	3440
<code>\dim_compare:nNnTF</code> . . . . .	94, 1409, 1562, 2161, 3353, 3363, 3652
<code>\dim_max:nn</code> . . . . .	3342, 3346
<code>\dim_min:nn</code> . . . . .	3334, 3338
<code>\dim_ratio:nn</code> . . . . .	2219, 2270, 2386, 2391, 2402, 2410, 2419, 2424, 2435, 2443
<code>\dim_set:Nn</code> . . . . .	3315, 3322, 3333, 3337, 3341, 3345, 3357, 3358, 3367, 3368, 3412, 3425
<code>\dim_set_eq:NN</code> . . . . .	3313, 3320, 3420, 3434
<code>\dim_sub:Nn</code> . . . . .	3437
<code>\dim_use:N</code> . . . . .	3354, 3364, 3415, 3416, 3428, 3429, 3452, 3453, 3454, 3455, 3479, 3480, 3481, 3482
<code>\dim_zero_new:N</code> . . . . .	3312, 3314, 3319, 3321
<code>\c_max_dim</code> . . . . .	3313, 3315, 3320, 3322, 3354, 3364
<code>\l_tmpc_dim</code> . . . . .	224, 2826, 2827, 2846, 2877, 2878, 2882, 2909, 2910, 2914, 2938, 2939, 2950, 3091, 3095, 3120, 3122, 3149, 3151, 3538, 3545, 3672, 3675, 3683
<code>\l_tmpd_dim</code> . . . . .	225, 2844, 2846, 2880, 2883, 2912, 2915, 2948, 2951, 3540, 3545, 3674, 3675, 3687
<code>\dotfill</code> . . . . .	820, 3641
<code>\dots</code> . . . . .	830
<code>\draw</code> . . . . .	2292
<b>E</b>	
else commands:	
<code>\else:</code> . . . . .	198
<code>\endarray</code> . . . . .	1304, 1332
<code>\endBNiceMatrix</code> . . . . .	3587
<code>\endbNiceMatrix</code> . . . . .	3584
<code>\endNiceArray</code> . . . . .	1657
<code>\endNiceArrayWithDelims</code> . . . . .	1579, 1587, 1595, 1603, 1611, 1619
<code>\endpgfscope</code> . . . . .	2354, 3686
<code>\endpNiceMatrix</code> . . . . .	3575
<code>\endVNiceMatrix</code> . . . . .	3581
<code>\endvNiceMatrix</code> . . . . .	3578
<code>\everycr</code> . . . . .	116, 131, 781
exp commands:	
<code>\exp_after:wN</code> . . . . .	174, 935
<code>\exp_args:Ne</code> . . . . .	121, 130
<code>\exp_args:NNc</code> . . . . .	3288
<code>\exp_args:NNV</code> . . . . .	1324, 2463, 2478, 2493, 2508, 2530, 2589, 2665, 2763
<code>\exp_args:Nnx</code> . . . . .	1627
<code>\exp_args:No</code> . . . . .	2287
<code>\exp_args:NV</code> . . . . .	876, 1300, 1327, 1329
<code>\exp_args:Nx</code> . . . . .	3542
<code>\exp_not:N</code> . . . . .	39, 40, 43, 44, 3202, 3203
<code>\exp_not:n</code> . . . . .	670, 2599, 2675, 3507
<code>\expandafter</code> . . . . .	733
<code>\ExplSyntaxOff</code> . . . . .	1680, 1701, 1724, 3303
<code>\ExplSyntaxOn</code> . . . . .	1666, 1687, 1706, 3296
<b>F</b>	
<code>\fi</code> . . . . .	99
fi commands:	
<code>\fi:</code> . . . . .	200
<code>\firstline</code> . . . . .	802
<code>\fontdimen</code> . . . . .	1254
fp commands:	
<code>\fp_eval:n</code> . . . . .	2325
<code>\fp_to_dim:n</code> . . . . .	2362
<code>\futurelet</code> . . . . .	104
<b>G</b>	
group commands:	
<code>\group_insert_after:N</code> . . . . .	2739, 2740, 2742, 2743, 3646, 3647, 3649, 3650
<b>H</b>	
<code>\halign</code> . . . . .	795
<code>\hbox</code> . . . . .	689, 1195, 1374, 1392, 1419, 1423, 1449
hbox commands:	
<code>\hbox:n</code> . . . . .	59, 61, 64
<code>\hbox_overlap_left:n</code> . . . . .	1518
<code>\hbox_overlap_right:n</code> . . . . .	1560
<code>\hbox_set:Nn</code> . . . . .	1047, 1049, 1188, 1263, 1278, 3529
<code>\hbox_set:Nw</code> . . . . .	560, 848, 862, 1079, 1495, 1537

`\hbox_set_end:` 621, 854, 868, 1105, 1514, 1555  
`\hbox_to_wd:nn` ..... 262, 287  
`\Hdotsfor` ..... 814, 3752  
`\hdotsfor` ..... 831  
`\hdottedline` ..... 812  
`\hfil` ..... 869  
`\hfill` ..... 115, 128  
`\hline` ..... 97, 802, 803  
`\hrule` ..... 101, 115, 128, 725  
`\hskip` ..... 100  
`\Hspace` ..... 813  
`\hspace` ..... 2556  
`\hss` ..... 869

## I

`\ialign` ..... 686, 769, 794, 1761  
`\iddots` ..... 811, 2534, 2535, 2540, 2541  
`\iddots` ..... 54, 801, 829  
if commands:  
`\if_mode_math:` ..... 198  
`\ifnum` ..... 99  
`\ifstandalone` ..... 921

int commands:

`\int_case:nnTF` ..... 2510, 2516, 2532, 2538  
`\int_compare:nNnTF` ..... 112, 113, 125,  
557, 558, 568, 575, 611, 722, 724, 880, 882,  
885, 946, 948, 991, 1001, 1020, 1106, 1110,  
1120, 1121, 1136, 1165, 1325, 1353, 1354,  
1540, 1829, 1836, 1840, 1842, 1897, 1904,  
1908, 1910, 2006, 2050, 2105, 2144, 2146,  
2564, 2651, 2726, 2747, 2839, 2873, 2941,  
2943, 2988, 2989, 2996, 2997, 3014, 3162,  
3164, 3166, 3168, 3176, 3178, 3180, 3182,  
3601, 3603, 3605, 3609, 3611, 3613, 3615, 3617  
`\int_compare_p:n` .... 2901, 2902, 2931, 2932  
`\int_gadd:Nn` ..... 2577  
`\int_gincr:N` .....  
. 556, 584, 924, 1445, 1535, 2206, 2257, 3281  
`\int_if_even:nTF` ..... 2973  
`\int_step_inline:nn` ..... 2969, 2971

iow commands:

`\iow_now:Nn` ..... 71, 1706, 1707, 1709, 1724  
`\iow_shipout:Nn` ..... 1666, 1667, 1674,  
1680, 1687, 1688, 1695, 1701, 3296, 3297, 3303

## K

`\kern` ..... 64

keys commands:

`\keys_define:nn` ..... 327, 346, 353, 407,  
440, 476, 508, 527, 536, 547, 3270, 3510, 3707  
`\l_keys_key_str` .....  
.... 15, 3810, 3885, 3890, 3930, 3976, 4025  
`\keys_set:nn` ..... 361,  
507, 931, 932, 1626, 1653, 2009, 2053,  
2108, 2180, 2231, 2654, 2729, 2766, 3283, 3523  
`\l_keys_value_tl` ..... 3849, 3856, 4070

## L

`\lasthline` ..... 803  
`\Ldots` ..... 807, 2466, 2469  
`\ldots` ..... 797, 825  
`\leaders` ..... 115, 128  
`\left` ..... 1191, 1266, 1281

legacy commands:

`\legacy_if:nTF` ..... 428  
`\line` ..... 1777, 3840

## M

`\makebox` ..... 855, 870  
`\mathinner` ..... 56

mode commands:

`\mode_leave_vertical:` ..... 746, 913

msg commands:

`\msg_error:nn` ..... 16, 20  
`\msg_error:nnn` ..... 21  
`\msg_error:nnnn` ..... 22, 3527  
`\msg_fatal:nn` ..... 23  
`\msg_fatal:nnn` ..... 24  
`\msg_new:nnn` ..... 9, 25  
`\msg_new:nnnn` ..... 26  
`\msg_redirect_name:nnn` ..... 28

`\multicolumn` ..... 816, 2558, 2582, 2602  
`\multispan` ..... 113, 114, 126, 127  
`\myfiledate` ..... 6  
`\myfileversion` ..... 7

## N

`\newcolumnntype` ..... 876  
`\NewDocumentCommand` ..... 506, 2463, 2478,  
2493, 2508, 2530, 2589, 2665, 2763, 2817,  
2853, 2889, 2922, 2957, 2967, 3490, 3594, 3634  
`\NewDocumentEnvironment` .....  
..... 904, 1296, 1306, 1572,  
1580, 1588, 1596, 1604, 1612, 1622, 1650, 3279  
`\NewExpandableDocumentCommand` ..... 186  
`\NiceArray` ..... 1655  
`\NiceArrayWithDelims` .....  
..... 1577, 1585, 1593, 1601, 1609, 1617  
`\NiceMatrixLastEnv` ..... 186  
`\NiceMatrixOptions` ..... 506, 3891  
`\NiceMatrixoptions` ..... 3853  
`\noalign` ..... 94, 99, 118, 713, 777, 3190  
`\normalbaselines` ..... 763  
`\nulldelimiterspace` ..... 1277, 1289  
`\numexpr` ..... 135, 136

## O

`\omit` ..... 112, 1356, 1368, 1444  
`\OnlyMainNiceMatrix` ..... 819, 2980

## P

peek commands:

`\peek_meaning:NTF` ..... 140, 735  
`\peek_meaning_ignore_spaces:NTF` .... 1298  
`\peek_meaning_remove_ignore_spaces:NTF` 129  
`\peek_remove_spaces:n` ..... 2562  
`\pgfextracty` ..... 3556  
`\pgfgetlastxy` ..... 280  
`\pgfpathcircle` ..... 2451  
`\pgfpathlineto` .... 3030, 3057, 3122, 3151, 3675  
`\pgfpathmoveto` .... 3029, 3054, 3121, 3150, 3670  
`\pgfpathrectanglecorners` .....  
..... 2845, 2881, 2913, 2949, 3093  
`\pgfpointhead` ..... 278, 3032  
`\pgfpointheadanchor` .....  
..... 189, 1977, 1988, 3332, 3340, 3551, 3552  
`\pgfpointdiff` ..... 279, 959, 967



<code>\pgfpointlineattime</code> .....	2319
<code>\pgfpointorigin</code> .....	1362, 1478
<code>\pgfpointscale</code> .....	278
<code>\pgfpointshapeborder</code> .....	2804, 2807
<code>\pgfrememberpicturepositiononpagetrue</code> .....	589, 642, 701, 1361, 1382, 1400, 1431, 1457, 1476, 1797, 2278, 2356, 2803, 3006, 3045, 3082, 3111, 3140, 3211, 3258, 3375, 3385, 3396, 3531, 3665
<code>\pgfscope</code> .....	2316, 3682
<code>\pgfset</code> .....	247, 272, 643, 3542, 3681
<code>\pgfsetbaseline</code> .....	641
<code>\pgfsetlinewidth</code> ..	3008, 3047, 3084, 3113, 3142
<code>\pgfsetrectcap</code> .....	3085, 3114, 3143
<code>\pgfsetroundcap</code> .....	3678
<code>\pgftransformrotate</code> .....	2323
<code>\pgftransformshift</code> .....	253, 278, 2317, 3558, 3563, 3683, 3687
<code>\pgfusepath</code> .....	2343, 2353
<code>\pgfusepathqfill</code> ..	2457, 2849, 2885, 2918, 2952
<code>\pgfusepathqstroke</code> .....	3037, 3059, 3096, 3123, 3152, 3679
<code>\phantom</code> .....	2475, 2490, 2505, 2527, 2549
<code>\pNiceMatrix</code> .....	3574
prg commands:	
<code>\prg_do_nothing:</code> .....	158, 167, 777, 1778
<code>\prg_replicate:nn</code> .....	1440, 1441, 2602, 3604, 3607, 3610, 3616
<code>\ProcessKeysOptions</code> .....	3716
<code>\ProvideDocumentCommand</code> .....	54
<code>\ProvidesExplPackage</code> .....	4

## Q

quark commands:	
<code>\q_stop</code> .....	109, 110, 122, 123, 141, 143, 935, 1316, 1319, 2757, 2771, 2772, 2812, 2834, 2835, 2868, 2869, 2898, 2929, 2940, 3469, 3476, 3491, 3492, 3589, 3598

## R

<code>\rectanglecolor</code> .....	980
<code>\relax</code> .....	135, 136
<code>\renewcommand</code> .....	172
<code>\RenewDocumentEnvironment</code> .....	3573, 3576, 3579, 3582, 3585
<code>\RequirePackage</code> .....	1, 3, 17, 18, 19
<code>\right</code> .....	1207, 1273, 1285
<code>\rotate</code> .....	818
<code>\rowcolor</code> .....	981
<code>\rowcolors</code> .....	982

## S

<code>\scriptstyle</code> .....	564, 1497, 1539, 2300, 2304, 2339, 2349
seq commands:	
<code>\seq_clear:N</code> .....	928, 929, 3719
<code>\seq_clear_new:N</code> .....	1708
<code>\seq_count:N</code> .....	1326
<code>\seq_gclear:N</code> .....	3158
<code>\seq_gclear_new:N</code> ....	833, 834, 1322, 1338
<code>\seq_gpop_left:NN</code> .....	1328, 1340
<code>\seq_gput_left:Nn</code> 433, 2566, 2568, 3502, 3503	
<code>\seq_gput_right:Nn</code> .....	1956, 2569
<code>\seq_gset_from_clist:Nn</code> .....	1711

<code>\seq_gset_split:Nnn</code> .....	1324, 1339
<code>\seq_if_empty:NTF</code> .....	1762
<code>\seq_if_empty_p:N</code> .....	3065, 3066
<code>\seq_if_exist:NTF</code> .....	938
<code>\seq_if_in:NnTF</code> .....	431, 3335, 3343, 3738
<code>\seq_item:Nn</code> ....	942, 945, 954, 955, 962, 963
<code>\seq_map_function:NN</code> .....	1330
<code>\seq_map_inline:Nn</code> .....	1342, 3104, 3106, 3133, 3135, 3516, 3720
<code>\seq_mapthread_function:NNN</code> .....	3464
<code>\seq_new:N</code> .....	192, 230, 231, 232
<code>\seq_put_left:Nn</code> .....	3722
<code>\seq_set_eq:NN</code> .....	3724
<code>\seq_set_from_clist:Nn</code> .....	3728
<code>\seq_use:Nnnn</code> .....	4079
<code>\l_tmpa_seq</code> .....	3719, 3722, 3724
skip commands:	
<code>\skip_gadd:Nn</code> .....	1416
<code>\skip_gset:Nn</code> .....	1407
<code>\skip_gset_eq:NN</code> .....	1414, 1415
<code>\skip_horizontal:N</code> 883, 1088, 1090, 1091, 1102, 1103, 1104, 1123, 1124, 1198, 1199, 1202, 1203, 1216, 1217, 1290, 1291, 1293, 1294, 1357, 1376, 1378, 1394, 1396, 1418, 1425, 1427, 1446, 1451, 1453, 1474, 1486, 1523, 1524, 1525, 1527, 1559, 1564, 1565, 1566	
<code>\skip_vertical:N</code> .....	118, 695, 697, 1194, 1205, 3190
<code>\skip_vertical:n</code> .....	2752
<code>\g_tmpa_skip</code> 1407, 1414, 1415, 1416, 1418, 1446	
<code>\c_zero_skip</code> .....	782
<code>\space</code> .....	210, 211

str commands:

<code>\c_backslash_str</code> .....	210
<code>\c_colon_str</code> .....	505
<code>\str_case:nnTF</code> .....	1231
<code>\str_gclear:N</code> .....	1782
<code>\str_gset:Nn</code> .....	910, 1576, 1583, 1591, 1599, 1607, 1615, 1624, 1652, 3625
<code>\str_if_empty:NTF</code> .....	593, 654, 704, 909, 1004, 1022, 1363, 1385, 1403, 1434, 1460, 1479, 1575, 1582, 1590, 1598, 1606, 1614, 1672, 1693, 3456, 3483
<code>\str_if_eq:nnTF</code> .....	80, 209, 423, 489, 684, 1128, 1144, 1147, 1225, 1346
<code>\str_if_eq_p:nn</code> .....	333, 3700
<code>\str_lowercase:n</code> .....	855, 870
<code>\str_new:N</code> .....	204, 205, 306, 316, 504
<code>\str_set:Nn</code> .....	79, 206, 307, 417, 418, 419, 430, 498, 1148, 3699
<code>\str_set_eq:NN</code> .....	434, 505
<code>\l_tmpa_str</code> .....	79, 80, 430, 431, 433, 434, 3699, 3700

## T

<code>\tabcolsep</code> .....	1083, 1198, 1202, 2022, 2032, 2066, 2074, 3220, 3227
<code>\tabskip</code> .....	782
T <sub>E</sub> X and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
<code>\@BTnormal</code> .....	755
<code>\@acol</code> .....	679
<code>\@acoll</code> .....	677
<code>\@acolr</code> .....	678
<code>\@addtopreamble</code> .....	927



# Contents

<b>1</b>	<b>The environments of this package</b>	<b>2</b>
<b>2</b>	<b>The vertical space between the rows</b>	<b>2</b>
<b>3</b>	<b>The vertical position of the arrays</b>	<b>3</b>
<b>4</b>	<b>The blocks</b>	<b>3</b>
<b>5</b>	<b>The rules</b>	<b>4</b>
5.1	The thickness and the color of the rules . . . . .	4
5.2	A remark about <code>\cline</code> . . . . .	5
5.3	The keys <code>hlines</code> and <code>vlines</code> . . . . .	5
5.4	The key <code>hvlines</code> . . . . .	6
5.5	La commande <code>\diagbox</code> . . . . .	6
5.6	Dotted rules . . . . .	6
<b>6</b>	<b>The color of the rows and columns</b>	<b>7</b>
<b>7</b>	<b>The width of the columns</b>	<b>8</b>
<b>8</b>	<b>The exterior rows and columns</b>	<b>10</b>
<b>9</b>	<b>The continuous dotted lines</b>	<b>11</b>
9.1	The option <code>nullify-dots</code> . . . . .	12
9.2	The command <code>\Hdotsfor</code> . . . . .	13
9.3	How to generate the continuous dotted lines transparently . . . . .	13
9.4	The labels of the dotted lines . . . . .	14
9.5	Customization of the dotted lines . . . . .	14
9.6	The dotted lines and the key <code>hvlines</code> . . . . .	15
<b>10</b>	<b>The code-after</b>	<b>16</b>
<b>11</b>	<b>Other features</b>	<b>16</b>
11.1	Use of the column type <code>S</code> of <code>siunitx</code> . . . . .	16
11.2	Alignement option in <code>{NiceMatrix}</code> . . . . .	16
11.3	The command <code>\rotate</code> . . . . .	17
11.4	The option <code>small</code> . . . . .	17
11.5	The counters <code>iRow</code> and <code>jCol</code> . . . . .	18
11.6	The option <code>light-syntax</code> . . . . .	18
11.7	The environment <code>{NiceArrayWithDelims}</code> . . . . .	19
<b>12</b>	<b>Utilisation of Tikz with <code>nicematrix</code></b>	<b>19</b>
12.1	The nodes corresponding to the contents of the cells . . . . .	19
12.2	The “medium nodes” and the “large nodes” . . . . .	20
12.3	The “row-nodes” and the “col-nodes” . . . . .	21
<b>13</b>	<b>Technical remarks</b>	<b>22</b>
13.1	Definition of new column types . . . . .	22
13.2	Diagonal lines . . . . .	22
13.3	The “empty” cells . . . . .	23
13.4	The option <code>exterior-arraycolsep</code> . . . . .	23
13.5	Incompatibilities . . . . .	24

<b>14</b>	<b>Examples</b>	<b>24</b>
14.1	Dotted lines . . . . .	24
14.2	Dotted lines which are no longer dotted . . . . .	26
14.3	Width of the columns . . . . .	26
14.4	How to highlight cells of the matrix . . . . .	27
14.5	Direct use of the Tikz nodes . . . . .	30
<b>15</b>	<b>Implementation</b>	<b>31</b>
<b>16</b>	<b>History</b>	<b>126</b>
	<b>Index</b>	<b>130</b>