

# The package `nicematrix`\*

F. Pantigny  
fpantigny@wanadoo.fr

April 25, 2021

## Abstract

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

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

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

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

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

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

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

The idea of `nicematrix` is to create PGF nodes under the cells and the positions of the rules of the tabular created by `array` and to use these nodes to develop new features. As usual with PGF, the coordinates of these nodes are written in the `.aux` to be used on the next compilation and that's why `nicematrix` may need **several compilations**.<sup>2</sup>

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

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

\*This document corresponds to the version 5.15 of `nicematrix`, at the date of 2021/04/25.

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

<sup>2</sup>If you use Overleaf, Overleaf will do automatically the right number of compilations.

# 1 The environments of this package

The package `nicematrix` defines the following new environments.

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

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

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

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

**It's recommended to use primarily the classical environments and to use the environments of `nicematrix` only when some feature provided by these environments is used (this will save memory).**

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

## Important

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

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

# 2 The vertical space between the rows

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

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

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

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

The initial value of these parameters is 0 pt in order to have for the environments of `nicematrix` the same behaviour as those of `array` and `amsmath`. However, a value of 1 pt would probably be a good choice and we suggest to set them with `\NiceMatrixOptions`.<sup>3</sup>

```
\NiceMatrixOptions{cell-space-limits = 1pt}
\begin{pNiceMatrix}
\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{3} & \frac{1}{4}
\end{pNiceMatrix}
```

---

<sup>3</sup>One should remark that these parameters apply also to the columns of type `S` of `siunitx` whereas the package `cellspace` is not able to act on such columns of type `S`.

### 3 The vertical position of the arrays

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

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

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

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

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

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

1. an item

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

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

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

1. an item

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

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

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

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

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

## 4 The blocks

### 4.1 General case

In the environments of `nicematrix`, it's possible to use the command `\Block` in order to place an element in the center of a rectangle of merged cells of the array.<sup>4</sup>

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

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

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

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

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

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

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

One may wish to raise the size of the “A” placed in the block of the previous example. Since this element is composed in math mode, it's not possible to use directly a command like `\large`, `\Large` and `\LARGE`. That's why the command `\Block` provides an option between angle brackets to specify some TeX code which will be inserted before the beginning of the math mode.<sup>5</sup>

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

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

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

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

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

In fact, the command `\Block` accepts as first optional argument (between square brackets) a list of couples key-value. The available keys are as follows:

- the keys `l`, `c` and `r` are used to fix the horizontal position of the content of the block, as explained previously;

<sup>4</sup>The spaces after a command `\Block` are deleted.

<sup>5</sup>This argument between angular brackets may also be used to insert a command of font such as `\bfseries` when the command `\\` is used in the content of the block.

- the key `fill` takes in as value a color and fills the block with that color;
- the key `draw` takes in as value a color and strokes the frame of the block with that color (the default value of that key is the current color of the rules of the array);
- the key `color` takes in as value a color and apply that color the content of the block but draws also the frame of the block with that color;
- the key `line-width` is the width (thickness) of the frame (this key should be used only when the key `draw` is in force);
- the key `rounded-corners` requires rounded corners (for the frame drawn by `draw` and the shape drawn by `fill`) with a radius equal to the value of that key (the default value is 4 pt<sup>6</sup>);
- the key `borders` provides the ability to draw only some borders of the blocks; the value of that key is a (comma-separated) list of elements covered by `left`, `right`, `top` and `bottom`;
- **New 5.15** the keys `hvlines` draws all the vertical and horizontal rules in the block;
- **New 5.14** the keys `t` and `b` fix the base line that will be given to the block when it has a multi-line content (the lines are separated by `\\`).

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

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

```
\begin{NiceTabular}{cwc{2cm}wc{3cm}c}
rose      & tulipe & marguerite & dahlia \\
violette  &         &             &         \\
& \Block[draw=red,fill=[RGB]{204,204,255},rounded-corners]{2-2}
&         {\LARGE De très jolies fleurs}
& & souci \\
pervenche & & & lys \\
arum      & iris & jacinthe & muguet
\end{NiceTabular}
```

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

## 4.2 The mono-column blocks

The mono-column blocks have a special behaviour.

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

---

<sup>6</sup>This value is the initial value of the *rounded corners* of Tikz.

```

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

```

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

### 4.3 The mono-row blocks

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

### 4.4 The mono-cell blocks

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

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

- It's possible to use the command `\` in a (mono-cell) block.
- It's possible to use the option of horizontal alignment of the block in derogation of the type of column given in the preamble of the array.
- It's possible to draw a frame around the cell with the key `draw` of the command `\Block` and to fill the background with rounded corners with the keys `fill` and `rounded-corners`.<sup>7</sup>
- It's possible to draw one or several borders of the cell with the key `borders`.

```

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

```

Writer	year of birth
Hugo	1802
Balzac	1799

We recall that if the first mandatory argument of `\Block` is left blank, the block is mono-cell.<sup>8</sup>

### 4.5 A small remark

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

<sup>7</sup>If one simply wishes to color the background of a unique celle, there is no point using the command `\Block`: it's possible to use the command `\cellcolor` (when the key `colortbl-like` is used).

<sup>8</sup>One may consider that the default value of the first mandatory argument of `\Block` is 1-1.

```

\begin{NiceTabular}{@{}c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
& \Block{1-3}{First group} & & \Block{1-3}{Second group} \\
Rank & 1A & 1B & 1C & 2A & 2B & 2C \\
\midrule
1 & 0.657 & 0.913 & 0.733 & 0.830 & 0.387 & 0.893 \\
2 & 0.343 & 0.537 & 0.655 & 0.690 & 0.471 & 0.333 \\
3 & 0.783 & 0.885 & 0.015 & 0.306 & 0.643 & 0.263 \\
4 & 0.161 & 0.708 & 0.386 & 0.257 & 0.074 & 0.336 \\
\bottomrule
\end{NiceTabular}

```

Rank	First group			Second group		
	1A	1B	1C	2A	2B	2C
1	0.657	0.913	0.733	0.830	0.387	0.893
2	0.343	0.537	0.655	0.690	0.471	0.333
3	0.783	0.885	0.015	0.306	0.643	0.263
4	0.161	0.708	0.386	0.257	0.074	0.336

## 5 The rules

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

### 5.1 Some differences with the classical environments

#### 5.1.1 The vertical rules

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

```

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

```

First	Second
Peter	
Mary	George

However, the vertical rules are not drawn in the blocks (created by `\Block`: cf. p. 4) nor in the corners (created by the key `corner`: cf. p. 9).

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

```

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

```

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

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

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

However, in this case, it is probably more clever to add a command `\OnlyMainNiceMatrix` (cf. p. 38):

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

### 5.1.2 The command `\cline`

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

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

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

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

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

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

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

## 5.2 The thickness and the color of the rules

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

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

With `nicematrix`, it’s possible to specify the color of the rules even when `colortbl` is not loaded. For sake of compatibility, the command is also named `\arrayrulecolor`. The environments of `nicematrix` also provide a key `rules/color` to fix the color of the rules in the current environment. This key sets the value locally (whereas `\arrayrulecolor` acts globally).

```
\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 38.

## 5.3 The tools of `nicematrix` for the rules

Here are the tools provided by `nicematrix` for the rules.

- the keys `hlines`, `vlines` and `hvlines`;
- the specifier “|” in the preamble (for the environments with preamble);
- the command `\Hline`.



All these tools don't draw the rules in the blocks nor in the empty corners (when the key `corners` is used).

- These blocks are:
  - the blocks created by the command `\Block`<sup>9</sup> presented p. 4;
  - the blocks implicitly delimited by the continuous dotted lines created by `\Cdots`, `Vdots`, etc. (cf. p. 18).
- The corners are created by the key `corners` explained below (see p. 9).

In particular, this remark explains the difference between the standard command `\hline` and the command `\Hline` provided by `nicematrix`.

### 5.3.1 The keys `hlines` and `vlines`

The keys `hlines` and `vlines` (which draw, of course, horizontal and vertical rules) take in as value a list of numbers which are the numbers of the rules to draw.

In fact, for the environments with delimiters (such as `{pNiceMatrix}` or `{bNiceArray}`), the key `vlines` don't draw the exterior rules (this is certainly the expected behaviour).

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

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

### 5.3.2 The key `hvlines`

The key `hvlines` (no value) is the conjunction of the keys `hlines` and `vlines`.

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

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

### 5.3.3 The (empty) corners

The four `corners` of an array will be designed by `NW`, `SW`, `NE` and `SE` (*north west*, *south west*, *north east* and *south east*).

For each of these corners, we will call *empty corner* (or simply *corner*) the reunion of all the empty rectangles starting from the cell actually in the corner of the array.<sup>10</sup>

However, it's possible, for a cell without content, to require `nicematrix` to consider that cell as not empty with the key `\NotEmpty`.

<sup>9</sup>And also the command `\multicolumn` also it's recommended to use instead `\Block` in the environments of `nicematrix`.

<sup>10</sup>For sake of completeness, we should also say that a cell contained in a block (even an empty cell) is not taken into account for the determination of the corners. That behaviour is natural.

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

```
\NiceMatrixOptions{cell-space-top-limit=3pt}
\begin{NiceTabular}{*{6}{c}}[corners,hvlines]
& & & & A & \\
& & A & A & A & \\
& & & A & & \\
& & A & A & A & A & \\
A & A & A & A & A & A & A & \\
A & A & A & A & A & A & A & \\
& A & A & A & & & \\
& \Block{2-2}{B} & & A & \\
& & & A & \\
\end{NiceTabular}
```

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

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

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

## 5.4 The command `\diagbox`

```

\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}$

```

$x \backslash y$	$e$	$a$	$b$	$c$
$e$	$e$	$a$	$b$	$c$
$a$	$a$	$e$	$c$	$b$
$b$	$b$	$c$	$e$	$a$
$c$	$c$	$b$	$a$	$e$

---

<sup>11</sup>The author of this document considers that type of construction as graphically poor.

## 5.5 Dotted rules

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

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

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

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

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

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

It's possible to change in `nicematrix` the letter used to specify a vertical dotted line with the option `letter-for-dotted-lines` available in `\NiceMatrixOptions`. Thus released, the letter “:” can be used otherwise (for example by the package `arydshln`<sup>12</sup>).

*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>13</sup>. In `nicematrix`, the dotted lines drawn by `\hdottedline` and “:” do likewise.

## 6 The color of the rows and columns

### 6.1 Use of `colortbl`

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

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

However, there is two drawbacks:

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

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

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

<sup>12</sup>However, one should remark that the package `arydshln` is not fully compatible with `nicematrix`.

<sup>13</sup>In fact, with `array`, this is true only for `\hline` and “|” but not for `\cline`: cf p. 8

## 6.2 The tools of `nicematrix` in the `\CodeBefore`

The package `nicematrix` provides some tools (independent of `colortbl`) to draw the colored panels first, and, then, the content of the cells and the rules. This strategy is more conform to the “painting model” of the formats PostScript and PDF and is more suitable for the PDF viewers. However, it requires several compilations.<sup>14</sup>

The extension `nicematrix` provides a key `code-before` for some code that will be executed before the drawing of the tabular.

An alternative syntax is provided: it’s possible to put the content of that `code-before` between the keywords `\CodeBefore` and `\Body` at the beginning of the environment.

```
\begin{pNiceArray}{preamble}
\CodeBefore
instructions of the code-before
\Body
contents of the environnement
\end{pNiceArray}
```

New commands are available in that `\CodeBefore`: `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors`, `\chessboardcolors` and `arraycolor`.<sup>15</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.

**New 5.15** These commands don’t color the cells which are in the “corners” if the key `corners` is used. This key has been described p. 9.

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

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

```
\begin{NiceTabular}{|c|c|c|}
\CodeBefore
\cellcolor[HTML]{FFFF88}{3-1,2-2,1-3}
\Body
\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|}
\CodeBefore
\rectanglecolor{blue!15}{2-2}{3-3}
\Body
\hline
a & b & c \\ \hline
e & f & g \\ \hline
h & i & j \\ \hline
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

<sup>14</sup>If you use Overleaf, Overleaf will do automatically the right number of compilations.

<sup>15</sup>Remark that, in the `\CodeBefore`, PGF/Tikz nodes of the form “(i-|j)” are also available to indicate the position to the potential rules: cf. p. 36.

- The command `\arraycolor` takes in as mandatory argument a color and color the whole tabular with that color (excepted the potential exterior rows and columns: cf. p. 17). It's only a particular case of `\rectanglecolor`.
- The command `\chessboardcolors` takes in as mandatory arguments two colors and it colors the cells of the tabular in quincunx with these colors.

```

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

```

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

We have used the key `r` which aligns all the columns rightwards (cf. p. 31).

- 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).

```

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

```

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

- The command `\columncolor` takes its name from the command `\columncolor` of `colortbl`. Its syntax is similar to the syntax of `\rowcolor`.
- The command `\rowcolors` (with a *s*) takes its name from the command `\rowcolors` of `xcolor`<sup>16</sup>. The *s* emphasizes the fact that there is *two* colors. This command colors alternately the rows of the tabular with the tow colors (provided in second and third argument), beginning with the row whose number is given in first (mandatory) argument.

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

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

<sup>16</sup>The command `\rowcolors` of `xcolor` is available when `xcolor` is loaded with the option `table`. That option also loads the package `colortbl`.

- The key `cols` describes a set of columns. The command `\rowcolors` will color only the cells of these columns. The value is a comma-separated list of intervals of the form  $i$ - $j$  (where  $i$  or  $j$  may be replaced by  $*$ ).
- With the key `restart`, each interval of rows (specified by the first mandatory argument) begins with the same color.<sup>17</sup>
- With the key `respect-blocks` the “rows” alternately colored may extend over several rows if they have to incorporate blocks (created with the command `\Block`: cf. p. 4).

```
\begin{NiceTabular}{clr}[hvlines]
\CodeBefore
  \rowcolors{gray}{2}{0.8}{}[cols=2-3,restart]
\Body
\Block{1-*}{Results} \\\
John & 12 \\\
Stephen & 8 \\\
Sarah & 18 \\\
Ashley & 20 \\\
Henry & 14 \\\
Madison & 15
\end{NiceTabular}
```

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

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

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

We recall that all the color commands we have described don’t color the cells which are in the “corners”. In the following example, we use the key `corners` to require the determination of the corner *north east* (NE).

```
\begin{NiceTabular}{cccccc}[corners=NE,margin,hvlines,first-row,first-col]
\CodeBefore
  \rowcolors{1}{blue!15}{}
\Body
  & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\\
0 & 1 \\\
1 & 1 & 1 \\\
2 & 1 & 2 & 1 \\\
3 & 1 & 3 & 3 & 1 \\\
4 & 1 & 4 & 6 & 4 & 1 \\\
5 & 1 & 5 & 10 & 10 & 5 & 1 \\\
6 & 1 & 6 & 15 & 20 & 15 & 6 & 1 \\\
\end{NiceTabular}
```

	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

<sup>17</sup>Otherwise, the color of a given row relies only upon the parity of its absolute number.

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

```
\begin{NiceTabular}[c]{lSSSS}
\CodeBefore
  \rowcolor{red!15}{1-2}
  \rowcolors{3}{blue!15}{}
\Body
\toprule
\Block{2-1}{Product} &
\Block{1-3}{dimensions (cm)} & & &
\Block{2-1}{\rotate Price} \\
\cmidrule{2-4}
& L & l & h & \\
\midrule
small & 3 & 5.5 & 1 & 30 \\
standard & 5.5 & 8 & 1.5 & 50.5 \\
premium & 8.5 & 10.5 & 2 & 80 \\
extra & 8.5 & 10 & 1.5 & 85.5 \\
special & 12 & 12 & 0.5 & 70 \\
\bottomrule
\end{NiceTabular}
```

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

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

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

It's possible to access the preceding tools with a syntax close to the syntax of `colortbl`. For that, one must use the key `colortbl-like` in the current environment.<sup>18</sup>

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

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

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

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

<sup>18</sup>As for now, this key is *not* available in `\NiceMatrixOptions`.

## 7 The width of the columns

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

```
\begin{NiceTabular}{Wc{2cm}cc}[hvlines]
Paris & New York & Madrid & \\
Berlin & London & Roma & \\
Rio & Tokyo & Oslo & \\
\end{NiceTabular}
```

Paris	New York	Madrid
Berlin	London	Roma
Rio	Tokyo	Oslo

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

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

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

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

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.<sup>19</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>20</sup>. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 4).

<sup>19</sup>The result is achieved with only one compilation (but PGF/Tikz will have written informations in the `.aux` file and a message requiring a second compilation will appear).

<sup>20</sup>At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.



```

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

```

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

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

## 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”.

```

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

```

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

The dotted lines have been drawn with the tools presented p. 18.

We have several remarks to do.

- For the environments with an explicit preamble (i.e. `{NiceTabular}`, `{NiceArray}` and its variants), no letter must be given in that preamble for the potential first column and the potential last column: they will automatically (and necessarily) be of type `r` for the first column and `l` for the last one.<sup>21</sup>
- One may wonder how `nicematrix` determines the number of rows and columns which are needed for the composition of the “last row” and “last column”.
  - For the environments with explicit preamble, like `{NiceTabular}` and `{pNiceArray}`, the number of columns can obviously be computed from the preamble.
  - When the option `light-syntax` (cf. p. 33) is used, `nicematrix` has, in any case, to load the whole body of the environment (and that’s why it’s not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows (but not the number of columns).

<sup>21</sup>The users wishing exterior columns with another type of alignment should consider the command `\SubMatrix` available in the `\CodeAfter` (cf. p. 24).

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

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

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

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

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

#### Remarks

- As shown in the previous example, the horizontal and vertical rules doesn't extend in the exterior rows and columns.
- However, if one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 38.
- 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. 16) doesn't apply to the "first column" and "last column".
- For technical reasons, it's not possible to use the option of the command `\` after the "first row" or before the "last row". The placement of the delimiters would be wrong. If you are looking for a workaround, consider the command `\SubMatrix` in the `\CodeAfter` described p. 24.

## 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>22</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>23</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>24</sup>

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

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

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

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

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

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

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

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

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

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

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.<sup>25</sup>

<sup>22</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>23</sup>The precise definition of a “non-empty cell” is given below (cf. p. 39).

<sup>24</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. 22.

<sup>25</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. 16

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

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

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

## 9.1 The option `nullify-dots`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```


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


```

Remark: Unlike the command `\hdotsfor` of `amsmath`, the command `\Hdotsfor` may be used even when the package `colortbl`<sup>26</sup> is loaded (but you might have problem if you use `\rowcolor` on the same row as `\Hdotsfor`).

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

```

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

```

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

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

Imagine you have a document with a great number of mathematical matrices with ellipsis. You may wish to use the dotted lines of `nicematrix` without having to modify the code of each matrix. It's possible with the keys. `renew-dots` and `renew-matrix`.<sup>27</sup>

<sup>26</sup>We recall that when `xcolor` is loaded with the option `table`, the package `colortbl` is loaded.

<sup>27</sup>The options `renew-dots`, `renew-matrix` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage`. There is also a key `transparent` which is an alias for the conjunction of `renew-dots` and `renew-matrix` but it must be considered as obsolete.

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>22</sup> and `\hdotsfor` are redefined within the environments provided by `nicematrix` and behave like `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor`; the command `\dots` (“automatic dots” of `amsmath`) is also redefined to behave like `\Ldots`.

- The option `renew-matrix`

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

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

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

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

## 9.4 The labels of the dotted lines

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and the command `\line` in the `\CodeAfter` which is described p. 24) accept two optional arguments specified by the tokens `_` and `^` for labels positionned below and above the line. The arguments are composed in math mode with `\scriptstyle`.

```
$\begin{bNiceMatrix}
1 & & \hspace*{1cm} & & & & 0 \\
& & \Ddots^{\text{times}} & & & & \\
0 & & & & & & 1 \\
\end{bNiceMatrix}$
```

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

## 9.5 Customisation of the dotted lines

The dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and by the command `\line` in the `\CodeAfter` which is described p. 24) may be customized by three options (specified between square brackets after the command):

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

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

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

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

### The option `xdots/color`

The option `xdots/color` fixes the color of the dotted line. However, one should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. p. 17.

### 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>28</sup>

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

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

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

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

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

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

## 9.6 The dotted lines and the rules

The dotted lines determine virtual blocks which have the same behaviour regarding the rules (the rules specified by the specifier `|` in the preamble, by the command `\Hline` and by the keys `hlines`, `vlines` and `hvlines` are not drawn within the blocks).<sup>29</sup>

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

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

<sup>28</sup>The first reason of this behaviour is that the PDF format includes a description for dashed lines. The lines specified with this descriptor are displayed very efficiently by the PDF readers. It’s easy, starting from these dashed lines, to create a line composed by square dots whereas a line of rounded dots needs a specification of each dot in the PDF file.

<sup>29</sup>On the other side, the command `\line` in the `\CodeAfter` (cf. p. 24) does *not* create block.

## 10 The `\CodeAfter`

The option `code-after` may be used to give some code that will be executed *after* the construction of the matrix.<sup>30</sup>

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

The experienced users may, for instance, use the PGF/Tikz nodes created by `nicematrix` in the `\CodeAfter`. These nodes are described further beginning on p. 33.

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

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

The command `\line` draws directly dotted lines between nodes. It takes in two arguments for the two cells to link, both of the form  $i-j$  where  $i$  is the number of the row and  $j$  is the number of the column. The options available for the customisation of the dotted lines created by `\Cdots`, `\Vdots`, etc. are also available for this command (cf. p. 22).

This command may be used, for example, to draw a dotted line between two adjacent cells.

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

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

It can also be used to draw a diagonal line not parallel to the other diagonal lines (by default, the dotted lines drawn by `\Ddots` are “parallelized”: cf. p. 38).

```
\begin{bNiceMatrix}
1      & \Cdots & 1      & 2      & \Cdots & 2      & \\
0      & \Ddots & \Vdots & \Vdots & \hspace*{2.5cm} & \Vdots & \\
\Vdots & \Ddots & & & & & \\
0      & \Cdots & 0 & 1      & 2      & \Cdots & 2 \\
\CodeAfter \line[shorten=6pt]{1-5}{4-7}
\end{bNiceMatrix}
```

$$\left[ \begin{array}{cccccc} 1 & \cdots & 1 & 2 & \cdots & 2 \\ 0 & \ddots & \vdots & \vdots & \hspace{2.5cm} & \vdots \\ \vdots & \ddots & & & & \\ 0 & \cdots & 0 & 1 & 2 & \cdots & 2 \end{array} \right]$$

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

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

- the first argument is the left delimiter, which may be any extensible delimiter provided by LaTeX : `(`, `[`, `\{`, `\langle`, `\lgroup`, `\lfloor`, etc. but also the null delimiter `.`;

<sup>30</sup>There is also a key `code-before` described p. 12.



- the second argument is the upper-left corner of the submatrix with the syntax  $i-j$  where  $i$  the number of row and  $j$  the number of column;
- the third argument is the lower-right corner with the same syntax;
- the fourth argument is the right delimiter;
- the last argument, which is optional, is a list of key-value pairs.<sup>31</sup>

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

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

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

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

- `left-xshift` and `right-shift` shift horizontally the delimiters (there exists also the key `xshift` which fixes both parameters);
- `extra-height` adds a quantity to the total height of the delimiters (height `\ht` + depth `\dp`);
- `delimiters/color` fixes the color of the delimiters (also available in `\NiceMatrixOptions`, in the environments with delimiters and as option of the keyword `\CodeAfter`);
- `slim` is a boolean key: when that key is in force, the horizontal position of the delimiters is computed by using only the contents of the cells of the submatrix whereas, in the general case, the position is computed by taking into account the cells of the whole columns implied in the submatrix (see example below). ;
- `vlines` contents a list of numbers of vertical rules that will be drawn in the sub-matrix (if this key is used without value, all the vertical rules of the sub-matrix are drawn);
- `hlines` is similar to `vlines` but for the horizontal rules;
- `hvlines`, which must be used without value, draws all the vertical and horizontal rules.

One should remark that these keys add their rules after the construction of the main matrix: no space is added between the rows and the columns of the array for theses rules.

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

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

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


---

<sup>31</sup>There is no optional argument between square brackets in first position because a square bracket just after `\SubMatrix` must be interpreted as the first (mandatory) argument of the command `\SubMatrix`: that bracket is the left delimiter of the sub-matrix to construct (eg.: `\SubMatrix[{2-2}{4-7}]`).

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

```


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


```

There is also a key `name` which gives a name to the submatrix created by `\SubMatrix`. That name is used to create PGF/Tikz nodes: cf p. 37.

**New 5.15** It's also possible to specify some delimiters<sup>32</sup> by placing them in the preamble of the environment (for the environments with a preamble: `{NiceArray}`, `{pNiceArray}`, etc.). This syntax is inspired by the extension `blkarray`.

When there are two successive delimiters (necessarily a closing one following by an opening one for another submatrix), a space equal to `\enskip` is automatically inserted.

```


$$\begin{pNiceArray}{(c)(c)(c)} \\ a_{11} & a_{12} & a_{13} \\ a_{21} & \displaystyle \int_0^1 \frac{1}{x^2+1} dx & a_{23} \\ a_{31} & a_{32} & a_{33} \\ \end{pNiceArray}$$


```

$$\begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} \begin{pmatrix} a_{12} \\ \int_0^1 \frac{1}{x^2+1} dx \\ a_{32} \end{pmatrix} \begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}$$

## 11 The notes in the tabulars

### 11.1 The footnotes

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

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

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

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

<sup>32</sup>Those delimiters are `(`, `[`, `\{` and the closing ones. Of course, it's also possible to put `|` and `||` in the preamble of the environment.

## 11.2 The notes of tabular

The package `nicematrix` also provides a command `\tabularnote` which gives the ability to specify notes that will be composed at the end of the array with a width of line equal to the width of the array (excepted the potential exterior columns). With no surprise, that command is available only in the environments without delimiters, that is to say `{NiceTabular}`, `{NiceArray}` and `{NiceMatrix}`. In fact, this command is available only if the extension `enumitem` has been loaded (before or after `nicematrix`). Indeed, the notes are composed at the end of the array with a type of list provided by the package `enumitem`.

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

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

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

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

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

For an illustration of some of those remarks, see table 1, p. 28. This table has been composed with the following code.

```

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

```

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

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

Some text before the notes.

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

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

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

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

### 11.3 Customisation of the tabular notes

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

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

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

```

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

```

We detail these keys.

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

Initial value: `false`

That key is also available within a given environment.

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

Initial value: `false`

That key is also available within a given environment.

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

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

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

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

Initial value: `\textsuperscript{#1}`

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

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

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

Initial value: `\textsuperscript{#1}`

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

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

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

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

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

This initial value contains the specification `align = left` which requires a composition of the label leftwards in the box affected to that label. With that tuning, the notes are composed flush left, which is pleasant when composing tabulars in the spirit of `booktabs` (see for example the table 1, p. 28).

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

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

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

Initial value: `empty`

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

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

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

For an example of customisation of the tabular notes, see p. 40.

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

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

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

## 12 Other features

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

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

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

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

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

## 12.2 Alignment option in {NiceMatrix}

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

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

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

## 12.3 The command \rotate

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

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

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

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

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

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

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

## 12.4 The option small

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

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

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

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

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

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

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

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

In the cells of the array, it's possible to use the LaTeX counters `iRow` and `jCol` which represent the number of the current row and the number of the current column<sup>33</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. 12) and in the `\CodeAfter` (cf. p. 24), `iRow` represents the total number of rows (excepted the potential exterior rows) and `jCol` represents the total number of columns (excepted the potential exterior columns).

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

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

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

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

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

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

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

---

<sup>33</sup>We recall that the exterior “first row” (if it exists) has the number 0 and that the exterior “first column” (if it exists) has also the number 0.



## 12.6 The option `light-syntax`

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

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

```


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


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


```

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

When the option `light-syntax` is used, it is not possible to put verbatim material (for example with the command `\verb`) in the cells of the array.<sup>34</sup>

## 12.7 Color of the delimiters

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

```


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


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


```

## 12.8 The environment `{NiceArrayWithDelims}`

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

```


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


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


```

# 13 Use of Tikz with `nicematrix`

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

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

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

However, it's possible to impose the creation of a node with the command `\NotEmpty`.<sup>35</sup>

<sup>34</sup>The reason is that, when the option `light-syntax` is used, the whole content of the environment is loaded as a TeX argument to be analyzed. The environment doesn't behave in that case as a standard environment of LaTeX which only put TeX commands before and after the content.

<sup>35</sup>One should note that, with that command, the cell is considered as non-empty, which has consequences for the continuous dotted lines (cf. p. 18) and the computation of the "corners" (cf. p. 9).

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

In the environment with the number  $n$ , the node of the row  $i$  and column  $j$  has for name `nm-n-i-j`.

The command `\NiceMatrixLastEnv` provides the number of the last environment of `nicematrix` (for LaTeX, it’s a “fully expandable” command and not a counter).

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

```

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

```

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

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

```

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

```

In the following example, we have underlined all the nodes of the matrix (we explain below the technic used : cf. p. 45).

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

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

In fact, the package `nicematrix` can create “extra nodes”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.<sup>36</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.

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

<sup>36</sup>There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

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>37</sup>

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

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>38</sup>

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

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

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

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

Here is an array composed with the following code:

```
\large
\begin{NiceTabular}{\w{2cm}\l{1}}[hvlines]
fraise & amande & abricot \\
prune & pêche & poire \\
noix & noisette & brugnon
\end{NiceTabular}
```

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Here are the “large nodes” of this array (with-out use of `margin` nor `extra-margin`).

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

<sup>37</sup>There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 17).

<sup>38</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.

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

The package `nicematrix` creates a PGF/Tikz node merely called  $i$  (with the classical prefix) at the intersection of the horizontal rule of number  $i$  and the vertical rule of number  $i$  (more specifically the potential position of those rules because maybe there are not actually drawn). The last node has also an alias called `last`.

**New 5.14** There is also a node called  $i.5$  midway between the node  $i$  and the node  $i + 1$ . These nodes are available in the `\CodeBefore` and the `\CodeAfter`.

	tulipe	lys
arum		violette mauve
muguet	dahlia	

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

```
\begin{NiceMatrix}
\CodeBefore
\tikz \draw [fill=red!15] (7-|4) |- (8-|5) |- (9-|6) |- cycle ;
\Body
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

The nodes of the form  $i.5$  may be used, for example to cross a row of a matrix (if Tikz is loaded).

```
$\begin{pNiceArray}{ccc|c}
2 & 1 & 3 & 0 \\\
3 & 3 & 1 & 0 \\\
3 & 3 & 1 & 0
\CodeAfter
\tikz \draw [red] (3.5-|1) -- (3.5-|last) ;
\end{pNiceArray}$
```

$$\left(\begin{array}{ccc|c} 2 & 1 & 3 & 0 \\ 3 & 3 & 1 & 0 \\ \hline 3 & 3 & 1 & 0 \end{array}\right)$$

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

The command `\SubMatrix` available in the `\CodeAfter` has been described p. 24.

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

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

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

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

## 14 API for the developpers

The package `nicematrix` provides two variables which are internal but public<sup>39</sup>:

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

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

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

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

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

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

Here is an example of use:

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

Tokyo	Paris	London
Lima	Oslo	Miami
Los Angeles	Madrid	Roma

<sup>39</sup>According to the LaTeX3 conventions, each variable with name beginning with `\g_nicematrix` ou `\l_nicematrix` is public and each variable with name beginning with `\g__nicematrix` or `\l__nicematrix` is private.

## 15 Technical remarks

### 15.1 Definition of new column types

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

For example, one may wish to define a new column type `?` in order to draw a (black) heavy rule of width 1 pt. The following definition will do the job<sup>40</sup>:

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

The heavy vertical rule won't extend in the exterior rows.<sup>41</sup>

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

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

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

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

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

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

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

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

### 15.2 Diagonal lines

By default, all the diagonal lines<sup>42</sup> of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That's why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

<sup>40</sup>The command `\vrule` is a TeX (and not LaTeX) command.

<sup>41</sup>Of course, such rule is defined by the classical technics of `nicematrix` and, for this reason, won't cross the double rules of `\hline\hline`.

<sup>42</sup>We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

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

### 15.3 The “empty” cells

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

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

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

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

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

### 15.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea<sup>43</sup>. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`<sup>44</sup>. The package `nicematrix` does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` of `array` (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of `nicematrix` are not affected).

### 15.5 Incompatibilities

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

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

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

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

<sup>43</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>44</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.

## 16 Examples

### 16.1 Notes in the tabulars

The tools provided by `nicematrix` for the composition of the tabular notes have been presented in the section 11 p. 26.

Let's consider that we wish to number the notes of a tabular with stars.<sup>45</sup>

First, we write a command `\stars` similar the well-known commands `\arabic`, `\alph`, `\Alph`, etc. which produces a number of stars equal to its argument <sup>46</sup>

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

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

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

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

---

<sup>45</sup>Of course, it's realistic only when there is very few notes in the tabular.

<sup>46</sup>In fact: the value of its argument.



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

\*Achard is an old family of the Poitou.

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

## 16.2 Dotted lines

An example with the resultant of two polynoms:

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

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

An example for a linear system:

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

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

### 16.3 Dotted lines which are no longer dotted

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

```

\NiceMatrixOptions[code-for-first-row = \scriptstyle,code-for-first-col = \scriptstyle }
\setcounter{MaxMatrixCols}{12}
\newcommand{\blue}{\color{blue}}
\[ \begin{pNiceMatrix}[last-row,last-col,nullify-dots,xdots/line-style={dashed,blue}]
1& & & \Vdots & & & \Vdots & \\
& \Ddots[line-style=standard] & \\
& & 1 & \\
& \Cdots[color=blue,line-style=dashed]& & & \blue 0 & \\
& \Cdots & & & \blue 1 & & & \Cdots & \blue \leftarrow i & \\
& & & & 1 & \\
& & & \Vdots & & \Ddots[line-style=standard] & & \Vdots & \\
& & & & & & & 1 & \\
& \Cdots & & & \blue 1 & \Cdots & & \Cdots & \blue 0 & & \Cdots & \blue \leftarrow j & \\
& & & & & & & & & & 1 & \\
& & & & & & & & \Ddots[line-style=standard] & \\
& & & \Vdots & & & & \Vdots & & & 1 & \\
& & & \blue \overset{\uparrow}{i} & & & & \blue \overset{\uparrow}{j} & \\
\end{pNiceMatrix} \]

```

$$\left( \begin{array}{ccc|ccc|ccc} 1 & \dots & & & & & & & \\ & & 1 & & & & & & \\ \hline & & & 0 & & & 1 & & \\ & & & & 1 & \dots & & & \\ & & & & & & & 1 & \\ \hline & & & & & & & & 0 \\ & & & 1 & & & & & \\ \hline & & & & & & & & \\ & & & & & & & 1 & \dots & 1 \end{array} \right) \begin{array}{l} \\ \\ \leftarrow i \\ \\ \\ \leftarrow j \\ \\ \end{array}$$

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

```

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

```

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

## 16.4 Stacks of matrices

We often need to compose mathematical matrices on top on each other (for example for the resolution of linear systems).

In order to have the columns aligned one above the other, it's possible to fix a width for all the columns. That's what is done in the following example with the environment `{NiceMatrixBlock}` and its option `auto-columns-width`.

```
\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
{
  light-syntax,
  last-col, code-for-last-col = \color{blue} \scriptstyle,
}
\setlength{\extrarowheight}{1mm}

$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 {} ;
3 -18 12 1 4 ;
-3 -46 29 -2 -15 ;
9 10 -5 4 7
\end{pNiceArray}$

\smallskip
$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 ;
0 64 -41 1 19 { L_2 \gets L_1-4L_2 } ;
0 -192 123 -3 -57 { L_3 \gets L_1+4L_3 } ;
0 -64 41 -1 -19 { L_4 \gets 3L_1-4L_4 } ;
\end{pNiceArray}$

\smallskip
$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 ;
0 64 -41 1 19 ;
0 0 0 0 0 { L_3 \gets 3 L_2 + L_3 }
\end{pNiceArray}$

\smallskip
$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 {} ;
0 64 -41 1 19 ;
\end{pNiceArray}$

\end{NiceMatrixBlock}
```

$$\left( \begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array} \right)$$

$$\left( \begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array} \right) \begin{array}{l} L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right)_{L_3 \leftarrow 3L_2 + L_3}$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

However, one can see that the last matrix is not perfectly aligned with others. That's why, in LaTeX, the parenthesis have not exactly the same width (smaller parenthesis are a bit slimer).

In order to solve that problem, it's possible to require the delimiters to be composed with the maximal width, thanks to the boolean key `delimiters/max-width`.

```
\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
{
  delimiters/max-width,
  light-syntax,
  last-col, code-for-last-col = \color{blue}\scriptstyle,
}
\setlength{\extrarowheight}{1mm}

$\begin{pNiceArray}{rrrr|r}
12 -8 7 5 3 {} ;
3 -18 12 1 4 ;
-3 -46 29 -2 -15 ;
9 10 -5 4 7
\end{pNiceArray}$

...
\end{NiceMatrixBlock}
```

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right)_{\begin{array}{l} L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}}$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right)_{L_3 \leftarrow 3L_2 + L_3}$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

If you wish an alignment of the different matrices without the same width for all the columns, you can construct a unique array and place the parenthesis with commands `\SubMatrix` in the `\CodeAfter`. Of course, that array can't be broken by a page break.

```
\setlength{\extrarowheight}{1mm}
\[\begin{NiceMatrix}[ r, last-col=6, code-for-last-col = \scriptstyle \color{blue} ]
12 & -8 & & 7 & 5 & 3 \\
3 & -18 & & 12 & 1 & 4 \\
-3 & -46 & & 29 & -2 & -15 \end{NiceMatrix}]
```

```

9 & 10 & -5 & 4 & 7 \\[1mm]
12 & -8 & 7 & 5 & 3 \\
0 & 64 & -41 & 1 & 19 & L_2 \gets L_1-4L_2 \\
0 & -192 & 123 & -3 & -57 & L_3 \gets L_1+4L_3 \\
0 & -64 & 41 & -1 & -19 & L_4 \gets 3L_1-4L_4 \\[1mm]
12 & -8 & 7 & 5 & 3 \\
0 & 64 & -41 & 1 & 19 \\
0 & 0 & 0 & 0 & 0 & L_3 \gets 3L_2+L_3 \\[1mm]
12 & -8 & 7 & 5 & 3 \\
0 & 64 & -41 & 1 & 19 \\
\CodeAfter [sub-matrix/vlines=4]
\SubMatrix({1-1}{4-5})
\SubMatrix({5-1}{8-5})
\SubMatrix({9-1}{11-5})
\SubMatrix({12-1}{13-5})
\end{NiceMatrix}\}

```

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{pmatrix}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{pmatrix}
\begin{array}{l} L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\begin{array}{l} L_3 \leftarrow 3L_2 + L_3 \end{array}$$

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{pmatrix}$$

## 16.5 How to highlight cells of a matrix

In order to highlight a cell of a matrix, it's possible to “draw” that cell with the key `draw` of the command `\Block` (this is one of the uses of a mono-cell block<sup>47</sup>).

```

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

```

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

<sup>47</sup>We recall that, if the first mandatory argument of the command `\Block` is left empty, that means that the block is a mono-cell block

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the command `\hline`, the specifier “|” and the options `hlines`, `vlines` and `hvlines` spread the cells.<sup>48</sup>

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

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

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

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

**Caution** : Some PDF readers are not able to show transparency.<sup>49</sup>

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

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

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

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

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

<sup>48</sup>For the command `\cline`, see the remark p. 8.

<sup>49</sup>In Overleaf, the “built-in” PDF viewer does not show transparency. You can switch to the “native” viewer in that case.

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

```

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

```

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

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

```

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

```

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

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

```

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

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

```

We obtain the following matrix.

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

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

```

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

```

We obtain the following matrix.

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

## 16.6 Utilisation of `\SubMatrix` in the `\CodeBefore`

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

The whole figure is an environment `{NiceArray}` and the three pairs of parenthesis have been added with `\SubMatrix` in the `code-before`.

You will find the LaTeX code of that figure in the source file of this document.

$$\begin{matrix} & & C_j \\ & \begin{pmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\ \vdots & & \vdots & & \vdots \\ b_{k1} & \cdots & b_{kj} & \cdots & b_{kn} \\ \vdots & & \vdots & & \vdots \\ b_{n1} & \cdots & b_{nj} & \cdots & b_{nn} \end{pmatrix} & & \\ & \swarrow & & & \\ L_i & \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{ik} & \cdots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} & \begin{pmatrix} \vdots \\ \vdots \\ \cdots c_{ij} \vdots \\ \vdots \end{pmatrix} \end{matrix}$$

## 17 Implementation

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

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

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

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

## Declaration of the package and packages loaded

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

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

<@@=nicematrix>

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

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

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

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



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

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

```

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

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

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

```

## Technical definitions

```

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

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

```

92     { \bool_set_true:N \c_@@_colortbl_loaded_bool }
93     {

```

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

```

94     \cs_set_protected:Npn \CT@arc@ { }
95     \cs_set:Npn \arrayrulecolor #1 # { \CT@arc@ { #1 } }
96     \cs_set:Npn \CT@arc@ #1 #2
97     {
98         \dim_compare:nNnT \baselineskip = \c_zero_dim \noalign
99         { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }
100    }

```

Idem for `\CT@drs@`.

```

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

```

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

```

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

```

The following `\skip_horizontal:N \c_zero_dim` is to prevent a potential `\unskip` to delete the `\leaders`<sup>50</sup>

```

129     \skip_horizontal:N \c_zero_dim
130 }

```

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

```

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

```

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

```

135 \cs_set:Npn \@@_cline

```

---

<sup>50</sup>See question 99041 on TeX StackExchange.

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

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

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

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

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

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

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

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

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

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

The following command is a small shortcut.

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

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

163 \cs_set_eq:NN \@@_old_pgfpointanchor \pgfpointanchor
```

## 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`.

```
164 \bool_new:N \c_@@_siunitx_loaded_bool
165 \AtBeginDocument
166 {
167 \ifpackageloaded { siunitx }
168 { \bool_set_true:N \c_@@_siunitx_loaded_bool }
169 { }
170 }
```

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

```
\renewcommand*{\NC@rewrite@S}[1] []
{
\temptokena \exp_after:wN
```

```

{
  \tex_the:D \@temptokena
  > { \__siunitx_table_collect_begin: S {#1} }
  c
  < { \__siunitx_table_print: }
}
\NC@find
}

```

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

```

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

```

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

```

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

```

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

```

177     \cs_set_eq:NN \NC@find \prg_do_nothing:
178     \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`).

```

179     \tl_gset:NV \g_tmpa_tl \@temptokena
180     \group_end:
181     \tl_new:N \c_@@_table_collect_begin_tl
182     \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
183     \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
184     \tl_new:N \c_@@_table_print_tl
185     \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).

```

186     \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
187   }
188 }

```

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

```

189 \AtBeginDocument

```

```

190 {
191   \bool_if:nTF { ! \c_@@_siunitx_loaded_bool }
192   { \cs_set_eq:NN \@@_renew_NC@rewrite@S: \prg_do_nothing: }
193   {
194     \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
195     {
196       \renewcommand*{\NC@rewrite@S}[1] []
197       {
198         \@temptokena \exp_after:wN
199         {
200           \tex_the:D \@temptokena
201           > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }
\@@_true_c: will be replaced statically by c at the end of the construction of the preamble.
202           \@@_true_c:
203           < { \c_@@_table_print_tl \@@_end_Cell: }
204         }
205         \NC@find
206       }
207     }
208   }
209 }

```

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

```

210 \regex_const:Nn \c_@@_columncolor_regex { \c { columncolor } }

```

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

```

211 \cs_new_protected:Npn \@@_provide_pgfsyspdfmark:
212 {
213   \iow_now:Nn \@mainaux
214   {
215     \ExplSyntaxOn
216     \cs_if_free:NT \pgfsyspdfmark
217     { \cs_set_eq:NN \pgfsyspdfmark \@gobblethree }
218     \ExplSyntaxOff
219   }
220   \cs_gset_eq:NN \@@_provide_pgfsyspdfmark: \prg_do_nothing:
221 }

```

## Parameters

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

```

222 \bool_new:N \c_@@_define_L_C_R_bool
223 \cs_new_protected:Npn \@@_define_L_C_R:
224 {
225   \newcolumntype L l
226   \newcolumntype C c
227   \newcolumntype R r
228 }

```

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.

```

229 \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).

```
230 \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.

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

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

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

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

```
235 \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).

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

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

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

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

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

Idem pour the mono-row blocks.

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

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

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

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

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

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

```
244 \bool_new:N \g_@@_NiceArray_bool
```

In fact, if there is delimiters in the preamble of `{NiceArray}` (eg: `[cccc]`), this boolean will be set to false.

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

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

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

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

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

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

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

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

```
249 \cs_new_protected:Npn \@@_test_if_math_mode:
250 {
251   \if_mode_math: \else:
252     \@@_fatal:n { Outside-math-mode }
253   \fi:
254 }
```

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

```
255 \tl_new:N \l_@@_letter_vlism_tl
```

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

```
256 \seq_new:N \g_@@_cols_vlism_seq
```

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

```
257 \colorlet { nicematrix-last-col } { . }
258 \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`).

```
259 \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*.

```
260 \tl_set:Nn \g_@@_com_or_env_str { environment }
```

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

```
261 \cs_new:Npn \@@_full_name_env:
262 {
263   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
264     { command \space \c_backslash_str \g_@@_name_env_str }
265     { environment \space \{ \g_@@_name_env_str \} }
266 }
```

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

```
267 \tl_new:N \g_nicematrix_code_after_tl
```

For the key code of the command `\SubMatrix` (itself in the main `\CodeAfter`), we will use the following token list.

```
268 \tl_new:N \l_@@_code_tl
```



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

```
269 \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.

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

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

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

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

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

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

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

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

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

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

```
276 \tl_new:N \l_@@_code_before_tl
277 \bool_new:N \l_@@_code_before_bool
```

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

```
278 \dim_new:N \l_@@_x_initial_dim
279 \dim_new:N \l_@@_y_initial_dim
280 \dim_new:N \l_@@_x_final_dim
281 \dim_new:N \l_@@_y_final_dim
```

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

```
282 \dim_zero_new:N \l_tmpc_dim
283 \dim_zero_new:N \l_tmpd_dim
```

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

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

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

```
285 \dim_new:N \g_@@_width_last_col_dim
286 \dim_new:N \g_@@_width_first_col_dim
```

The following sequence will contain the characteristics of the blocks of the array, specified by the command `\Block`. Each block is represented by 6 components surrounded by curly braces:

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

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

```
287 \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}`.

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

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

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

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

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

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

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

If the user has used the key `corners` (or the key `hvlines-except-corners`), all the cells which are in an (empty) corner will be stored in the following sequence.

```
291 \seq_new:N \l_@@_corners_cells_seq
```

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

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

The following counters will be used when searching the extremities of a dotted line (we need these counters because of the potential “open” lines in the `\SubMatrix`—the `\SubMatrix` in the `code-before`).

```
293 \int_new:N \l_@@_row_min_int
294 \int_new:N \l_@@_row_max_int
295 \int_new:N \l_@@_col_min_int
296 \int_new:N \l_@@_col_max_int
```

The following sequence will be used when the command `\SubMatrix` is used in the `code-before` (and not in the `\CodeAfter`). It will contain the position of all the sub-matrices specified in the `code-before`. Each sub-matrix is represented by an “object” of the forme `{i}{j}{k}{l}` where *i* and *j* are the number of row and column of the upper-left cell and *k* and *l* the number of row and column of the lower-right cell.

```
297 \seq_new:N \g_@@_submatrix_seq
```

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

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

The following parameters correspond to the keys `fill`, `draw`, `borders` and `rounded-corners` of the command `\Block`.

```
299 \tl_new:N \l_@@_fill_tl
300 \tl_new:N \l_@@_draw_tl
301 \clist_new:N \l_@@_borders_clist
302 \dim_new:N \l_@@_rounded_corners_dim
```

The last parameter has no direct link with the [empty] corners of the array (which are computed and taken into account by `nicematrix` when the key `corners` is used).

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

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

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

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

The parameters of position of the label of a block. For the horizontal position, the possible values are `c`, `r` and `l`. For the vertical position, the possible values are `c`, `t` and `b`. Of course, it would be interesting to program a key `T` and a key `B`.

```
305 \tl_new:N \l_@@_hpos_of_block_tl
306 \tl_set:Nn \l_@@_hpos_of_block_tl { c }
307 \tl_new:N \l_@@_vpos_of_block_tl
308 \tl_set:Nn \l_@@_vpos_of_block_tl { c }
```

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

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

The following flag corresponds to the key `hvlines` of the command `\Block`.

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

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

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

312 \dim_new:N \l_@@_submatrix_extra_height_dim
313 \dim_new:N \l_@@_submatrix_left_xshift_dim
314 \dim_new:N \l_@@_submatrix_right_xshift_dim
315 \clist_new:N \l_@@_hlines_clist
316 \clist_new:N \l_@@_vlines_clist
317 \clist_new:N \l_@@_submatrix_hlines_clist
318 \clist_new:N \l_@@_submatrix_vlines_clist
```

## Variables for the exterior rows and columns

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

- **First row**

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

```
319 \int_new:N \l_@@_first_row_int
320 \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.

```
321 \int_new:N \l_@@_first_col_int
322 \int_set:Nn \l_@@_first_col_int 1
```

- **Last row**

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

```
323 \int_new:N \l_@@_last_row_int
324 \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>51</sup>

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

Idem for `\l_@@_last_col_without_value_bool`

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

- **Last column**

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

```
327 \int_new:N \l_@@_last_col_int
328 \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.

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

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

---

<sup>51</sup>We can’t use `\l_@@_last_row_int` for this usage because, if `nicematrix` has read its value from the `aux` file, the value of the counter won’t be `-1` any longer.

## The command `\tabularnote`

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

```
330 \newcounter { tabularnote }
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
338 \AtBeginDocument
339 {
340   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
341   {
342     \NewDocumentCommand \tabularnote { m }
343     { \@@_error:n { enumitem-not-loaded } }
344   }
345   {
```

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

```
346     \newlist { tabularnotes } { enumerate } { 1 }
347     \setlist [ tabularnotes ]
348     {
349       topsep = Opt ,
350       noitemsep ,
351       leftmargin = * ,
```

```

352         align = left ,
353         labelsep = Opt ,
354         label =
355         \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
356     }
357     \newlist { tabularnotes* } { enumerate* } { 1 }
358     \setlist [ tabularnotes* ]
359     {
360         afterlabel = \nobreak ,
361         itemjoin = \quad ,
362         label =
363         \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
364     }

```

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

Unfortunately, if the package `caption` is loaded, the command `\caption` evaluates its argument twice and since it is not aware (of course) of `\tabularnote`, the command `\tabularnote` is, in fact, not usable in `\caption` when `caption` is loaded.<sup>52</sup>

```

365     \NewDocumentCommand \tabularnote { m }
366     {
367         \bool_if:nTF { ! \g_@@_NiceArray_bool && \l_@@_in_env_bool }
368         { \@@_error:n { tabularnote~forbidden } }
369         {

```

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

```

370         \int_incr:N \l_@@_number_of_notes_int

```

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

```

371         \seq_gput_right:Nn \g_@@_tabularnotes_seq { #1 }
372         \peek_meaning:NF \tabularnote
373         {

```

If the following token is *not* a `\tabularnote`, we have finished the sequence of successive commands `\tabularnote` and we have to format the labels of these tabular notes (in the array). We compose those labels in a box `\l_tmpa_box` because we will do a special construction in order to have this box in a overlapping position if we are at the end of a cell.

```

374         \hbox_set:Nn \l_tmpa_box
375         {

```

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

```

376         \@@_notes_label_in_tabular:n
377         {
378             \stepcounter { tabularnote }
379             \@@_notes_style:n { tabularnote }
380             \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
381             {
382                 ,
383                 \stepcounter { tabularnote }
384                 \@@_notes_style:n { tabularnote }
385             }
386         }
387     }

```

---

<sup>52</sup>We should try to find a solution to that problem.

We use `\refstepcounter` in order to have the (last) tabular note referenceable (with the standard command `\label`) and that's why we have to go back with a decrementation of the counter `tabularnote` first.

```

388             \addtocounter { tabularnote } { -1 }
389             \refstepcounter { tabularnote }
390             \int_zero:N \l_@@_number_of_notes_int
391             \hbox_overlap_right:n { \box_use:N \l_tmpa_box }

```

If the command `\tabularnote` is used exactly at the end of the cell, the `\unskip` (inserted by `array?`) will delete the skip we insert now and the label of the footnote will be composed in an overlapping position (by design).

```

392             \skip_horizontal:n { \box_wd:N \l_tmpa_box }
393         }
394     }
395 }
396 }
397 }

```

## 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.

```

398 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
399 {
400     \begin { pgfscope }
401     \pgfset
402     {
403         outer~sep = \c_zero_dim ,
404         inner~sep = \c_zero_dim ,
405         minimum~size = \c_zero_dim
406     }
407     \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
408     \pgfnode
409     { rectangle }
410     { center }
411     {
412         \vbox_to_ht:nn
413         { \dim_abs:n { #5 - #3 } }
414         {
415             \vfill
416             \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
417         }
418     }
419     { #1 }
420     { }
421     \end { pgfscope }
422 }

```

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

```

423 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
424 {
425     \begin { pgfscope }
426     \pgfset
427     {
428         outer~sep = \c_zero_dim ,
429         inner~sep = \c_zero_dim ,
430         minimum~size = \c_zero_dim
431     }
432     \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }

```

```

433 \pgfpointdiff { #3 } { #2 }
434 \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
435 \pgfnode
436 { rectangle }
437 { center }
438 {
439   \vbox_to_ht:nn
440   { \dim_abs:n \l_tmpb_dim }
441   { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
442 }
443 { #1 }
444 { }
445 \end { pgfscope }
446 }

```

## The options

By default, the commands `\cellcolor` and `\rowcolor` are available for the user in the cells of the tabular (the user may use the commands provided by `\colortbl`). However, if the key `colortbl-like` is used, these commands are available.

```

447 \bool_new:N \l_@@_colortbl_like_bool

```

By default, the behaviour of `\cline` is changed in the environments of `nicematrix`: a `\cline` spreads the array by an amount equal to `\arrayrulewidht`. It's possible to disable this feature with the key `\l_@@_standard_line_bool`.

```

448 \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`).

```

449 \dim_new:N \l_@@_cell_space_top_limit_dim
450 \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.

```

451 \dim_new:N \l_@@_inter_dots_dim
452 \AtBeginDocument { \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em } }

```

The `\AtBeginDocument` is only a security in case `revtex4-1` is used (even though it is obsolete).

The following dimension is the minimal distance between a node (in fact an anchor of that node) and a dotted line (we say “minimal” because, by definition, a dotted line is not a continuous line and, therefore, this distance may vary a little).

```

453 \dim_new:N \l_@@_xdots_shorten_dim
454 \AtBeginDocument { \dim_set:Nn \l_@@_xdots_shorten_dim { 0.3 em } }

```

The `\AtBeginDocument` is only a security in case `revtex4-1` is used (even though it is obsolete).

The following dimension is the radius of the dots for the dotted lines (when `line-style` is equal to `standard`, which is the initial value). The initial value is 0.53 pt but it will be changed if the option `small` is used.

```

455 \dim_new:N \l_@@_radius_dim
456 \AtBeginDocument { \dim_set:Nn \l_@@_radius_dim { 0.53 pt } }

```



The `\AtBeginDocument` is only a security in case `revtex4-1` is used (even if it is obsolete).

The token list `\l_@@_xdots_line_style_tl` corresponds to the option `tikz` of the commands `\Cdots`, `\Ldots`, etc. and of the options `line-style` for the environments and `\NiceMatrixOptions`. The constant `\c_@@_standard_tl` will be used in some tests.

```
457 \tl_new:N \l_@@_xdots_line_style_tl
458 \tl_const:Nn \c_@@_standard_tl { standard }
459 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

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

The string `\l_@@_baseline_tl` may contain one of the three values `t`, `c` or `b` as in the option of the environment `{array}`. However, it may also contain an integer (which represents the number of the row to which align the array).

```
461 \tl_new:N \l_@@_baseline_tl
462 \tl_set:Nn \l_@@_baseline_tl c
```

The flag `\l_@@_exterior_arraycolsep_bool` corresponds to the option `exterior-arraycolsep`. If this option is set, a space equal to `\arraycolsep` will be put on both sides of an environment `{NiceArray}` (as it is done in `{array}` of `array`).

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

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

```
464 \bool_new:N \l_@@_parallelize_diags_bool
465 \bool_set_true:N \l_@@_parallelize_diags_bool
```

The following parameter correspond to the key `corners`. The elements of that `clist` must be in `NW`, `SW`, `NE` and `SE`.

```
466 \clist_new:N \l_@@_corners_clist
```

```
467 \dim_new:N \l_@@_notes_above_space_dim
468 \AtBeginDocument { \dim_set:Nn \l_@@_notes_above_space_dim { 1 mm } }
```

The `\AtBeginDocument` is only a security in case `revtex4-1` is used (even if it is obsolete).

The flag `\l_@@_nullify_dots_bool` corresponds to the option `nullify-dots`. When the flag is down, the instructions like `\vdots` are inserted within a `\hphantom` (and so the constructed matrix has exactly the same size as a matrix constructed with the classical `{matrix}` and `\ldots`, `\vdots`, etc.).

```
469 \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).

```
470 \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.

```
471 \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”.

```
472 \bool_new:N \l_@@_medium_nodes_bool
473 \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`).

```
474 \dim_new:N \l_@@_left_margin_dim
475 \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`.

```
476 \dim_new:N \l_@@_extra_left_margin_dim
477 \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.

```
478 \tl_new:N \l_@@_end_of_row_tl
479 \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 “:”.

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

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

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

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

```
482 \bool_new:N \l_@@_delimiters_max_width_bool
```

```
483 \keys_define:nn { NiceMatrix / xdots }
484 {
485   line-style .code:n =
486   {
487     \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.

```
488   { \cs_if_exist_p:N \tikzpicture }
489   { \str_if_eq_p:nn { #1 } { standard } }
490   { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
491   { \@@_error:n { bad-option-for~line-style } }
492   } ,
493   line-style .value_required:n = true ,
494   color .tl_set:N = \l_@@_xdots_color_tl ,
495   color .value_required:n = true ,
496   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
497   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 `_`.

```
498   down .tl_set:N = \l_@@_xdots_down_tl ,
499   up .tl_set:N = \l_@@_xdots_up_tl ,
```

The key `draw-first`, which is meant to be used only with `\Ddots` and `\Iddots`, which be caught when `\Ddots` or `\Iddots` is used (during the construction of the array and not when we draw the dotted lines).

```
500   draw-first .code:n = \prg_do_nothing: ,
501   unknown .code:n = \@@_error:n { Unknown-key-for~xdots }
502 }
```

```

503 \keys_define:nn { NiceMatrix / rules }
504 {
505   color .tl_set:N = \l_@@_rules_color_tl ,
506   color .value_required:n = true ,
507   width .dim_set:N = \arrayrulewidth ,
508   width .value_required:n = true
509 }

```

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

```

510 \keys_define:nn { NiceMatrix / Global }
511 {
512   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
513   rules .value_required:n = true ,
514   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
515   standard-cline .default:n = true ,
516   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
517   cell-space-top-limit .value_required:n = true ,
518   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
519   cell-space-bottom-limit .value_required:n = true ,
520   cell-space-limits .meta:n =
521   {
522     cell-space-top-limit = #1 ,
523     cell-space-bottom-limit = #1 ,
524   } ,
525   cell-space-limits .value_required:n = true ,
526   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
527   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
528   light-syntax .default:n = true ,
529   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
530   end-of-row .value_required:n = true ,
531   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
532   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
533   last-row .int_set:N = \l_@@_last_row_int ,
534   last-row .default:n = -1 ,
535   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
536   code-for-first-col .value_required:n = true ,
537   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
538   code-for-last-col .value_required:n = true ,
539   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
540   code-for-first-row .value_required:n = true ,
541   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
542   code-for-last-row .value_required:n = true ,
543   hlines .clist_set:N = \l_@@_hlines_clist ,
544   vlines .clist_set:N = \l_@@_vlines_clist ,
545   hlines .default:n = all ,
546   vlines .default:n = all ,
547   vlines-in-sub-matrix .code:n =
548   {
549     \tl_if_single_token:nTF { #1 }
550     { \tl_set:Nn \l_@@_letter_vlism_tl { #1 } }
551     { \@@_error:n { One-letter-allowed } }
552   } ,
553   vlines-in-sub-matrix .value_required:n = true ,
554   hvlines .code:n =
555   {
556     \clist_set:Nn \l_@@_vlines_clist { all }
557     \clist_set:Nn \l_@@_hlines_clist { all }
558   } ,
559   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.

```

560   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
561   renew-dots .value_forbidden:n = true ,
562   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
563   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
564   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
565   create-extra-nodes .meta:n =
566     { create-medium-nodes , create-large-nodes } ,
567   left-margin .dim_set:N = \l_@@_left_margin_dim ,
568   left-margin .default:n = \arraycolsep ,
569   right-margin .dim_set:N = \l_@@_right_margin_dim ,
570   right-margin .default:n = \arraycolsep ,
571   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
572   margin .default:n = \arraycolsep ,
573   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
574   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
575   extra-margin .meta:n =
576     { extra-left-margin = #1 , extra-right-margin = #1 } ,
577   extra-margin .value_required:n = true ,
578 }

```

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

```

579 \keys_define:nn { NiceMatrix / Env }
580 {
581   delimiters/max-width .bool_set:N = \l_@@_delimiters_max_width_bool ,

```

The key `hvlines-except-corners` is now deprecated.

```

582   hvlines-except-corners .code:n =
583     {
584       \clist_set:Nn \l_@@_corners_clist { #1 }
585       \clist_set:Nn \l_@@_vlines_clist { all }
586       \clist_set:Nn \l_@@_hlines_clist { all }
587     } ,
588   hvlines-except-corners .default:n = { NW , SW , NE , SE } ,
589   corners .clist_set:N = \l_@@_corners_clist ,
590   corners .default:n = { NW , SW , NE , SE } ,
591   code-before .code:n =
592     {
593       \tl_if_empty:nF { #1 }
594       {
595         \tl_put_right:Nn \l_@@_code_before_tl { #1 }
596         \bool_set_true:N \l_@@_code_before_bool
597       }
598     } ,

```

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

```

599   c .code:n = \tl_set:Nn \l_@@_baseline_tl c ,
600   t .code:n = \tl_set:Nn \l_@@_baseline_tl t ,
601   b .code:n = \tl_set:Nn \l_@@_baseline_tl b ,
602   baseline .tl_set:N = \l_@@_baseline_tl ,
603   baseline .value_required:n = true ,
604   columns-width .code:n =
605     \tl_if_eq:nnTF { #1 } { auto }
606     { \bool_set_true:N \l_@@_auto_columns_width_bool }
607     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
608   columns-width .value_required:n = true ,
609   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.

```

610 \legacy_if:nF { measuring@ }
611 {
612     \str_set:Nn \l_tmpa_str { #1 }
613     \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
614     { \@@_error:nn { Duplicate~name } { #1 } }
615     { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
616     \str_set_eq:NN \l_@@_name_str \l_tmpa_str
617 } ,
618 name .value_required:n = true ,
619 code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
620 code-after .value_required:n = true ,
621 colortbl-like .code:n =
622     \bool_set_true:N \l_@@_colortbl_like_bool
623     \bool_set_true:N \l_@@_code_before_bool ,
624 colortbl-like .value_forbidden:n = true
625 }
626 \keys_define:nn { NiceMatrix / notes }
627 {
628     para .bool_set:N = \l_@@_notes_para_bool ,
629     para .default:n = true ,
630     code-before .tl_set:N = \l_@@_notes_code_before_tl ,
631     code-before .value_required:n = true ,
632     code-after .tl_set:N = \l_@@_notes_code_after_tl ,
633     code-after .value_required:n = true ,
634     bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
635     bottomrule .default:n = true ,
636     style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
637     style .value_required:n = true ,
638     label-in-tabular .code:n =
639         \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
640     label-in-tabular .value_required:n = true ,
641     label-in-list .code:n =
642         \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
643     label-in-list .value_required:n = true ,
644     enumitem-keys .code:n =
645         {
646             \bool_if:NTF \c_@@_in_preamble_bool
647             {
648                 \AtBeginDocument
649                 {
650                     \bool_if:NT \c_@@_enumitem_loaded_bool
651                     { \setlist* [ tabularnotes ] { #1 } }
652                 }
653             }
654             {
655                 \bool_if:NT \c_@@_enumitem_loaded_bool
656                 { \setlist* [ tabularnotes ] { #1 } }
657             }
658         } ,
659     enumitem-keys .value_required:n = true ,
660     enumitem-keys-para .code:n =
661         {
662             \bool_if:NTF \c_@@_in_preamble_bool
663             {
664                 \AtBeginDocument
665                 {
666                     \bool_if:NT \c_@@_enumitem_loaded_bool
667                     { \setlist* [ tabularnotes* ] { #1 } }
668                 }
669             }
670             {
671                 \bool_if:NT \c_@@_enumitem_loaded_bool
672                 { \setlist* [ tabularnotes* ] { #1 } }

```

```

673     }
674   } ,
675   enumitem-keys-para .value_required:n = true ,
676   unknown .code:n = \@@_error:n { Unknown~key~for~notes }
677 }

```

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

```

678 \keys_define:nn { NiceMatrix }
679 {
680   NiceMatrixOptions .inherit:n =
681     { NiceMatrix / Global } ,
682   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
683   NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
684   NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
685   NiceMatrixOptions / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
686   SubMatrix / rules .inherit:n = NiceMatrix / rules ,
687   CodeAfter / xdots .inherit:n = NiceMatrix / xdots ,
688   NiceMatrix .inherit:n =
689     {
690       NiceMatrix / Global ,
691       NiceMatrix / Env ,
692     } ,
693   NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
694   NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
695   NiceTabular .inherit:n =
696     {
697       NiceMatrix / Global ,
698       NiceMatrix / Env
699     } ,
700   NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
701   NiceTabular / rules .inherit:n = NiceMatrix / rules ,
702   NiceArray .inherit:n =
703     {
704       NiceMatrix / Global ,
705       NiceMatrix / Env ,
706     } ,
707   NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
708   NiceArray / rules .inherit:n = NiceMatrix / rules ,
709   pNiceArray .inherit:n =
710     {
711       NiceMatrix / Global ,
712       NiceMatrix / Env ,
713     } ,
714   pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
715   pNiceArray / rules .inherit:n = NiceMatrix / rules ,
716 }

```

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

```

717 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
718 {
719   delimiters / max-width .bool_set:N = \l_@@_delimiters_max_width_bool ,
720   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
721   delimiters / color .value_required:n = true ,
722   delimiters-color .code:n =
723     \tl_set:Nn \l_@@_delimiters_color_tl { #1 }
724     \@@_error:n { delimiters-color deleted } ,
725   delimiters-color .value_required:n = true ,
726   last-col .code:n = \tl_if_empty:nF { #1 }
727     { \@@_error:n { last-col~non~empty~for~NiceMatrixOptions } }
728     \int_zero:N \l_@@_last_col_int ,

```

```

729   small .bool_set:N = \l_@@_small_bool ,
730   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.

```

731   renew-matrix .code:n = \@@_renew_matrix: ,
732   renew-matrix .value_forbidden:n = true ,

```

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

```

733   transparent .code:n =
734   {
735     \@@_renew_matrix:
736     \bool_set_true:N \l_@@_renew_dots_bool
737     \@@_error:n { Key~transparent }
738   } ,
739   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}`.

```

740   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.

```

741   columns-width .code:n =
742   \tl_if_eq:nnTF { #1 } { auto }
743   { \@@_error:n { Option~auto~for~columns-width } }
744   { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

745   allow-duplicate-names .code:n =
746   \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
747   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`).

```

748   letter-for-dotted-lines .code:n =
749   {
750     \tl_if_single_token:nTF { #1 }
751     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
752     { \@@_error:n { One~letter~allowed } }
753   } ,
754   letter-for-dotted-lines .value_required:n = true ,
755   notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
756   notes .value_required:n = true ,
757   sub-matrix .code:n =
758   \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
759   sub-matrix .value_required:n = true ,
760   unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
761 }

762 \str_new:N \l_@@_letter_for_dotted_lines_str
763 \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.

```

764 \NewDocumentCommand \NiceMatrixOptions { m }
765 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

766 \keys_define:nn { NiceMatrix / NiceMatrix }
767 {
768   last-col .code:n = \tl_if_empty:nTF {#1}
769     {
770       \bool_set_true:N \l_@@_last_col_without_value_bool
771       \int_set:Nn \l_@@_last_col_int { -1 }
772     }
773     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
774   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
775   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
776   small .bool_set:N = \l_@@_small_bool ,
777   small .value_forbidden:n = true ,
778   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
779   delimiters / color .value_required:n = true ,
780   delimiters-color .code:n =
781     \tl_set:Nn \l_@@_delimiters_color_tl { #1 }
782     \@@_error:n { delimiters-color deleted } ,
783   delimiters-color .value_required:n = true ,
784   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
785 }

```

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

```

786 \keys_define:nn { NiceMatrix / NiceArray }
787 {

```

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.

```

788   small .bool_set:N = \l_@@_small_bool ,
789   small .value_forbidden:n = true ,
790   last-col .code:n = \tl_if_empty:nF { #1 }
791     { \@@_error:n { last-col-non-empty-for-NiceArray } }
792     \int_zero:N \l_@@_last_col_int ,
793   notes / para .bool_set:N = \l_@@_notes_para_bool ,
794   notes / para .default:n = true ,
795   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
796   notes / bottomrule .default:n = true ,
797   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
798   tabularnote .value_required:n = true ,
799   delimiters-color .code:n =
800     \tl_set:Nn \l_@@_delimiters_color_tl { #1 }
801     \@@_error:n { delimiters-color deleted } ,
802   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
803   delimiters / color .value_required:n = true ,
804   r .code:n = \@@_error:n { r~or~l~with~preamble } ,
805   l .code:n = \@@_error:n { r~or~l~with~preamble } ,
806   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
807 }

808 \keys_define:nn { NiceMatrix / pNiceArray }
809 {
810   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
811   last-col .code:n = \tl_if_empty:nF {#1}
812     { \@@_error:n { last-col-non-empty-for-NiceArray } }
813     \int_zero:N \l_@@_last_col_int ,
814   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
815   small .bool_set:N = \l_@@_small_bool ,
816   small .value_forbidden:n = true ,
817   r .code:n = \@@_error:n { r~or~l~with~preamble } ,
818   l .code:n = \@@_error:n { r~or~l~with~preamble } ,
819   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }

```



```
820 }
```

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

```
821 \keys_define:nn { NiceMatrix / NiceTabular }
822 {
823   notes / para .bool_set:N = \l_@@_notes_para_bool ,
824   notes / para .default:n = true ,
825   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
826   notes / bottomrule .default:n = true ,
827   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
828   tabularnote .value_required:n = true ,
829   last-col .code:n = \tl_if_empty:nF {#1}
830     { \@@_error:n { last-col~non~empty~for~NiceArray } }
831     \int_zero:N \l_@@_last_col_int ,
832   r .code:n = \@@_error:n { r~or~l~with~preamble } ,
833   l .code:n = \@@_error:n { r~or~l~with~preamble } ,
834   unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
835 }
```

## 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}).

```
836 \cs_new_protected:Npn \@@_Cell:
837 {
```

At the beginning of the cell, we link \CodeAfter to a command which do *not* begin with \omit (whereas the standard version of \CodeAfter begins with \omit).

```
838   \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n
```

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

```
839   \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 arydshn, create special rows in the \halign that we don’t want to take into account.

```
840   \int_compare:nNnT \c@jCol = 1
841     { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
```

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).

```
842   \hbox_set:Nw \l_@@_cell_box
843   \bool_if:NF \l_@@_NiceTabular_bool
844   {
845     \c_math_toggle_token
846     \bool_if:NT \l_@@_small_bool \scriptstyle
847   }
```

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.

```
848   \int_compare:nNnTF \c@iRow = 0
849   {
850     \int_compare:nNnT \c@jCol > 0
851     {
852       \l_@@_code_for_first_row_tl
853       \xglobal \colorlet { nicematrix-first-row } { . }
854     }
```

```

855     }
856     {
857         \int_compare:nNnT \c@iRow = \l_@@_last_row_int
858         {
859             \l_@@_code_for_last_row_tl
860             \xglobal \colorlet { nicematrix-last-row } { . }
861         }
862     }
863 }

```

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.

```

864 \cs_new_protected:Npn \@@_begin_of_row:
865 {
866     \int_gincr:N \c@iRow
867     \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
868     \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
869     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
870     \pgfpicture
871     \pgfrememberpicturepositiononpagetrue
872     \pgfcoordinate
873     { \@@_env: - row - \int_use:N \c@iRow - base }
874     { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
875     \str_if_empty:NF \l_@@_name_str
876     {
877         \pgfnodealias
878         { \l_@@_name_str - row - \int_use:N \c@iRow - base }
879         { \@@_env: - row - \int_use:N \c@iRow - base }
880     }
881     \endpgfpicture
882 }

```

The following code is used in each cell of the array. It actualises quantities that, at the end of the array, will give informations about the vertical dimension of the two first rows and the two last rows. If the user uses the `last-row`, some lines of code will be dynamically added to this command.

```

883 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
884 {
885     \int_compare:nNnTF \c@iRow = 0
886     {
887         \dim_gset:Nn \g_@@_dp_row_zero_dim
888         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
889         \dim_gset:Nn \g_@@_ht_row_zero_dim
890         { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
891     }
892     {
893         \int_compare:nNnT \c@iRow = 1
894         {
895             \dim_gset:Nn \g_@@_ht_row_one_dim
896             { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
897         }
898     }
899 }
900 \cs_new_protected:Npn \@@_rotate_cell_box:
901 {
902     \box_rotate:Nn \l_@@_cell_box { 90 }
903     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
904     {
905         \vbox_set_top:Nn \l_@@_cell_box
906         {
907             \vbox_to_zero:n { }

```

```

908         \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
909         \box_use:N \l_@@_cell_box
910     }
911 }
912 \bool_gset_false:N \g_@@_rotate_bool
913 }
914 \cs_new_protected:Npn \@@_adjust_size_box:
915 {
916     \dim_compare:nNnT \g_@@_blocks_wd_dim > \c_zero_dim
917     {
918         \box_set_wd:Nn \l_@@_cell_box
919         { \dim_max:nn { \box_wd:N \l_@@_cell_box } \g_@@_blocks_wd_dim }
920         \dim_gzero:N \g_@@_blocks_wd_dim
921     }
922     \dim_compare:nNnT \g_@@_blocks_dp_dim > \c_zero_dim
923     {
924         \box_set_dp:Nn \l_@@_cell_box
925         { \dim_max:nn { \box_dp:N \l_@@_cell_box } \g_@@_blocks_dp_dim }
926         \dim_gzero:N \g_@@_blocks_dp_dim
927     }
928     \dim_compare:nNnT \g_@@_blocks_ht_dim > \c_zero_dim
929     {
930         \box_set_ht:Nn \l_@@_cell_box
931         { \dim_max:nn { \box_ht:N \l_@@_cell_box } \g_@@_blocks_ht_dim }
932         \dim_gzero:N \g_@@_blocks_ht_dim
933     }
934 }
935 \cs_new_protected:Npn \@@_end_Cell:
936 {
937     \@@_math_toggle_token:
938     \hbox_set_end:
939     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
940     \@@_adjust_size_box:
941     \box_set_ht:Nn \l_@@_cell_box
942     { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
943     \box_set_dp:Nn \l_@@_cell_box
944     { \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”).

```

945     \dim_gset:Nn \g_@@_max_cell_width_dim
946     { \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”.

```

947     \@@_update_for_first_and_last_row:

```

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

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

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

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

```

948 \bool_if:NTF \g_@@_empty_cell_bool
949 { \box_use_drop:N \l_@@_cell_box }
950 {
951   \bool_lazy_or:nnTF
952     \g_@@_not_empty_cell_bool
953     { \dim_compare_p:nNn { \box_wd:N \l_@@_cell_box } > \c_zero_dim }
954     \@@_node_for_the_cell:
955     { \box_use_drop:N \l_@@_cell_box }
956 }
957 \int_gset:Nn \g_@@_col_total_int { \int_max:nn \g_@@_col_total_int \c_jCol }
958 \bool_gset_false:N \g_@@_empty_cell_bool
959 \bool_gset_false:N \g_@@_not_empty_cell_bool
960 }

```

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

```

961 \cs_new_protected:Npn \@@_node_for_the_cell:
962 {
963   \pgfpicture
964   \pgfsetbaseline \c_zero_dim
965   \pgfrememberpicturepositiononpagetrue
966   \pgfset
967   {
968     inner~sep = \c_zero_dim ,
969     minimum~width = \c_zero_dim
970   }
971   \pgfnode
972   { rectangle }
973   { base }
974   { \box_use_drop:N \l_@@_cell_box }
975   { \@@_env: - \int_use:N \c_iRow - \int_use:N \c_jCol }
976   { }
977   \str_if_empty:NF \l_@@_name_str
978   {
979     \pgfnodealias
980     { \l_@@_name_str - \int_use:N \c_iRow - \int_use:N \c_jCol }
981     { \@@_env: - \int_use:N \c_iRow - \int_use:N \c_jCol }
982   }
983   \endpgfpicture
984 }

```

The second argument of the following command `\@@_instruction_of_type:nnn` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). The third argument is the list of options. This command writes in the corresponding `\g_@@_type_lines_tl` the instruction which will actually draw the line after the construction of the matrix.

For example, for the following matrix,

```

\begin{pNiceMatrix}
1 & 2 & 3 & 4 \\
5 & \Cdots & & 6 \\
7 & \Cdots & [color=red]
\end{pNiceMatrix}

```

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

the content of `\g_@@_Cdots_lines_tl` will be:

```

\@@_draw_Cdots:nnn {2}{2}{}
\@@_draw_Cdots:nnn {3}{2}{color=red}

```

The first argument is a boolean which indicates whether you must put the instruction on the left or on the right on the list of instructions.

```

985 \cs_new_protected:Npn \@@_instruction_of_type:nnn #1 #2 #3
986 {
987   \bool_if:NTF { #1 } \tl_gput_left:cx \tl_gput_right:cx

```

```

988 { g_@@_ #2 _ lines _ t1 }
989 {
990   \use:c { @@ _ draw _ #2 : nnn }
991   { \int_use:N \c@iRow }
992   { \int_use:N \c@jCol }
993   { \exp_not:n { #3 } }
994 }
995 }

```

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`.

```

996 \cs_new_protected:Npn \@@_revtex_array:
997 {
998   \cs_set_eq:NN \@acoll \@arrayacol
999   \cs_set_eq:NN \@acolr \@arrayacol
1000   \cs_set_eq:NN \@acol \@arrayacol
1001   \cs_set_nopar:Npn \@halignto { }
1002   \@array@array
1003 }
1004 \cs_new_protected:Npn \@@_array:
1005 {
1006   \bool_if:NTF \c_@@_revtex_bool
1007     \@@_revtex_array:
1008     {
1009       \bool_if:NTF \l_@@_NiceTabular_bool
1010         { \dim_set_eq:NN \col@sep \tabcolsep }
1011         { \dim_set_eq:NN \col@sep \arraycolsep }
1012       \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim
1013         { \cs_set_nopar:Npn \@halignto { } }
1014         { \cs_set_nopar:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }

```

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

```

1015   \@tabarray
1016 }

```

`\l_@@_baseline_tl` may have the value `t`, `c` or `b`. However, if the value is `b`, we compose the `\array` (of `array`) with the option `t` and the right translation will be done further. Remark that `\str_if_eq:VnTF` is fully expandable and you need something fully expandable here.

```

1017 [ \str_if_eq:VnTF \l_@@_baseline_tl c c t ]
1018 }

```

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.

```

1019 \cs_set_eq:NN \@@_old_ialign: \ialign

```

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

```

1020 \cs_new_protected:Npn \@@_create_row_node:
1021 {

```

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

```

1022   \hbox
1023   {
1024     \bool_if:NT \l_@@_code_before_bool
1025     {
1026       \vtop
1027       {
1028         \skip_vertical:N 0.5\arrayrulewidth
1029         \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
1030         \skip_vertical:N -0.5\arrayrulewidth
1031       }
1032     }
1033     \pgfpicture

```

```

1034 \pgfrememberpicturepositiononpagetrue
1035 \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
1036 { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
1037 \str_if_empty:NF \l_@@_name_str
1038 {
1039 \pgfnodealias
1040 { \l_@@_name_str - row - \int_use:N \c@iRow }
1041 { \@@_env: - row - \int_use:N \c@iRow }
1042 }
1043 \endpgfpicture
1044 }
1045 }

```

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

```

1046 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
1047 \cs_new_protected:Npn \@@_everycr_i:
1048 {
1049 \int_gzero:N \c@jCol
1050 \bool_gset_false:N \g_@@_after_col_zero_bool
1051 \bool_if:NF \g_@@_row_of_col_done_bool
1052 {
1053 \@@_create_row_node:

```

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

```

1054 \tl_if_empty:NF \l_@@_hlines_clist
1055 {
1056 \tl_if_eq:NnF \l_@@_hlines_clist { all }
1057 {
1058 \exp_args:NNx
1059 \clist_if_in:NnT
1060 \l_@@_hlines_clist
1061 { \@@_succ:n \c@iRow }
1062 }
1063 {

```

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.

```

1064 \int_compare:nNnT \c@iRow > { -1 }
1065 {
1066 \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@`.

```

1067 { \hrule height \arrayrulewidth width \c_zero_dim }
1068 }
1069 }
1070 }
1071 }
1072 }

```

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

```

1073 \cs_set_protected:Npn \@@_newcolumnntype #1
1074 {
1075 \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
1076 \peek_meaning:NTF [
1077 { \newcol@ #1 }
1078 { \newcol@ #1 [ 0 ] }
1079 }

```

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

```

1080 \cs_set_protected:Npn \@@_renew_dots:
1081 {
1082   \cs_set_eq:NN \ldots \@@_Ldots
1083   \cs_set_eq:NN \cdots \@@_Cdots
1084   \cs_set_eq:NN \vdots \@@_Vdots
1085   \cs_set_eq:NN \ddots \@@_Ddots
1086   \cs_set_eq:NN \iddots \@@_Iddots
1087   \cs_set_eq:NN \dots \@@_Ldots
1088   \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
1089 }

```

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

```

1090 \cs_new_protected:Npn \@@_colortbl_like:
1091 {
1092   \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
1093   \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
1094   \cs_set_eq:NN \columncolor \@@_columncolor_preamble
1095 }

```

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

```

1096 \cs_new_protected:Npn \@@_pre_array_ii:
1097 {

```

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>53</sup>.

```

1098   \bool_if:NT \c_@@_booktabs_loaded_bool
1099     { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
1100   \box_clear_new:N \l_@@_cell_box
1101   \cs_if_exist:NT \theiRow
1102     { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
1103   \int_gzero_new:N \c@iRow
1104   \cs_if_exist:NT \thejCol
1105     { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
1106   \int_gzero_new:N \c@jCol
1107   \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}`).

```

1108   \bool_if:NT \l_@@_small_bool
1109     {
1110       \cs_set_nopar:Npn \arraystretch { 0.47 }
1111       \dim_set:Nn \arraycolsep { 1.45 pt }
1112     }

```

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`.

```

1113   \cs_set_nopar:Npn \ialign
1114     {

```

---

<sup>53</sup>cf. `\nicematrix@redefine@check@rerun`

```

1115 \bool_if:NTF \c_@@_colortbl_loaded_bool
1116 {
1117   \CT@everycr
1118   {
1119     \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
1120     \@@_everycr:
1121   }
1122 }
1123 { \everycr { \@@_everycr: } }
1124 \tabskip = \c_zero_skip

```

The box `\@arstrutbox` is a box constructed in the beginning of the environment `{array}`. The construction of that box takes into account the current value of `\arraystretch`<sup>54</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`.

```

1125 \dim_gzero_new:N \g_@@_dp_row_zero_dim
1126 \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
1127 \dim_gzero_new:N \g_@@_ht_row_zero_dim
1128 \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
1129 \dim_gzero_new:N \g_@@_ht_row_one_dim
1130 \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
1131 \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
1132 \dim_gzero_new:N \g_@@_ht_last_row_dim
1133 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
1134 \dim_gzero_new:N \g_@@_dp_last_row_dim
1135 \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`.

```

1136 \cs_set_eq:NN \ialign \@@_old_ialign:
1137 \halign
1138 }

```

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).

```

1139 \cs_set_eq:NN \@@_old_ldots \ldots
1140 \cs_set_eq:NN \@@_old_cdots \cdots
1141 \cs_set_eq:NN \@@_old_vdots \vdots
1142 \cs_set_eq:NN \@@_old_ddots \ddots
1143 \cs_set_eq:NN \@@_old_iddots \iddots
1144 \bool_if:NTF \l_@@_standard_cline_bool
1145 { \cs_set_eq:NN \cline \@@_standard_cline }
1146 { \cs_set_eq:NN \cline \@@_cline }
1147 \cs_set_eq:NN \Ldots \@@_Ldots
1148 \cs_set_eq:NN \Cdots \@@_Cdots
1149 \cs_set_eq:NN \Vdots \@@_Vdots
1150 \cs_set_eq:NN \Ddots \@@_Ddots
1151 \cs_set_eq:NN \Iddots \@@_Iddots
1152 \cs_set_eq:NN \hdottedline \@@_hdottedline:
1153 \cs_set_eq:NN \Hline \@@_Hline:
1154 \cs_set_eq:NN \Hspace \@@_Hspace:
1155 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
1156 \cs_set_eq:NN \Vdotsfor \@@_Vdotsfor:
1157 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
1158 \cs_set_eq:NN \Block \@@_Block:
1159 \cs_set_eq:NN \rotate \@@_rotate:
1160 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n

```

<sup>54</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.



```

1161 \cs_set_eq:NN \dotfill \@@_old_dotfill:
1162 \cs_set_eq:NN \CodeAfter \@@_CodeAfter:
1163 \cs_set_eq:NN \diagbox \@@_diagbox:nn
1164 \cs_set_eq:NN \NotEmpty \@@_NotEmpty:
1165 \bool_if:NT \l_@@_colortbl_like_bool \@@_colortbl_like:
1166 \bool_if:NT \l_@@_renew_dots_bool \@@_renew_dots:

```

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

```

1167 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
1168 \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).

```

1169 \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`).

```

1170 \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.

```

1171 \int_gzero_new:N \g_@@_col_total_int
1172 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1173 \@@_renew_NC@rewrite@S:
1174 \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.

```

1175 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1176 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1177 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1178 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1179 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1180 \tl_gclear_new:N \g_@@_HVDotsfor_lines_tl

1181 \tl_gclear_new:N \g_nicematrix_code_before_tl
1182 }

```

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

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.

```

1183 \cs_new_protected:Npn \@@_pre_array:
1184 {
1185   \seq_gclear:N \g_@@_submatrix_seq
1186   \bool_if:NT \l_@@_code_before_bool
1187   {

```

The list of the cells which are in the (empty) corners is stored in the `aux` file because we have to know it before the execution of the `\CodeBefore` (the commands which color the cells, row and columns won’t color the cells which are in the corners).

```

1188   \seq_if_exist:cT
1189   { c_@@_corners_cells_ \int_use:N \g_@@_env_int _ seq }
1190   {
1191     \seq_set_eq:Nc \l_@@_corners_cells_seq
1192     { c_@@_corners_cells_ \int_use:N \g_@@_env_int _ seq }
1193   }

```

Now, we have to reconstruct the `row` nodes and the `col` nodes.

```
1194 \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1195 {
```

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

```
1196 \int_zero_new:N \c@iRow
1197 \int_set:Nn \c@iRow
1198 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1199 \int_zero_new:N \c@jCol
1200 \int_set:Nn \c@jCol
1201 { \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.

```
1202 \int_compare:nNf \l_@@_last_row_int = { -2 }
1203 { \int_decr:N \c@iRow }
1204 \int_compare:nNf \l_@@_last_col_int = { -2 }
1205 { \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.

```
1206 \pgfsys@markposition { \@@_env: - position }
1207 \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1208 \pgfpicture
```

First, the creation of the `row` nodes.

```
1209 \int_step_inline:nnn
1210 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1211 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1212 {
1213 \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1214 \pgfcoordinate { \@@_env: - row - ##1 }
1215 { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1216 }
```

Now, the creation of the `col` nodes.

```
1217 \int_step_inline:nnn
1218 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1219 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1220 {
1221 \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1222 \pgfcoordinate { \@@_env: - col - ##1 }
1223 { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1224 }
1225 \endpgfpicture
```

Now, you create the diagonal nodes by using the `row` nodes and the `col` nodes. If the engine is `xetex` or `luatex` we also create the “½ nodes”.

```
1226 \@@_create_diag_nodes:
```

Now, yet other settings before the execution of the `code-before`.

```
1227 \group_begin:
1228 \bool_if:NT \c_@@_tikz_loaded_bool
1229 {
1230 \tikzset
1231 {
1232 every-picture / .style =
1233 { overlay , name-prefix = \@@_env: - }
1234 }
1235 }
1236 \cs_set_eq:NN \cellcolor \@@_cellcolor
1237 \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
```

```

1238      \cs_set_eq:NN \roundedrectanglecolor \@@_roundedrectanglecolor
1239      \cs_set_eq:NN \rowcolor \@@_rowcolor
1240      \cs_set_eq:NN \rowcolors \@@_rowcolors
1241      \cs_set_eq:NN \arraycolor \@@_arraycolor
1242      \cs_set_eq:NN \columncolor \@@_columncolor
1243      \cs_set_eq:NN \chessboardcolors \@@_chessboardcolors
1244      \cs_set_eq:NN \SubMatrix \@@_SubMatrix_in_code_before

```

We compose the `code-before` in math mode in order to nullify the spaces put by the user between instructions in the `code-before`.

```

1245      \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1246      \seq_gclear_new:N \g_@@_colors_seq

```

Here is the `\CodeBefore`. As of now, the keys that may be provided to the keyword `\CodeBefore` are the same as keys that may be provided to `\CodeAfter`, hence the `\@@_CodeAfter_keys:`.

```

1247      \exp_last_unbraced:NV \@@_CodeAfter_keys: \l_@@_code_before_tl

```

Now, all the cells which are specified to be colored by instructions in the `\CodeBefore` will actually be colored. It's a two-stages mechanism because we want to draw all the cells with the same color at the same time to absolutely avoid thin white lines in some PDF viewers.

```

1248      \@@_actually_color:
1249      \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1250      \group_end:
1251    }
1252  }

```

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).

```

1253      \int_compare:nNnT \l_@@_last_row_int > { -2 }
1254      {
1255        \tl_put_right:Nn \@@_update_for_first_and_last_row:
1256        {
1257          \dim_gset:Nn \g_@@_ht_last_row_dim
1258          { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1259          \dim_gset:Nn \g_@@_dp_last_row_dim
1260          { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1261        }
1262      }
1263      \int_compare:nNnT \l_@@_last_row_int = { -1 }
1264      {
1265        \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.

```

1266      \str_if_empty:NTF \l_@@_name_str
1267      {
1268        \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1269        {
1270          \int_set:Nn \l_@@_last_row_int
1271          { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1272        }
1273      }
1274      {
1275        \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1276        {
1277          \int_set:Nn \l_@@_last_row_int
1278          { \use:c { @@_last_row_ \l_@@_name_str } }
1279        }
1280      }
1281    }

```

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).

```

1282      \int_compare:nNnT \l_@@_last_col_int = { -1 }

```

```

1283 {
1284   \str_if_empty:NTF \l_@@_name_str
1285   {
1286     \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1287     {
1288       \int_set:Nn \l_@@_last_col_int
1289       { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1290     }
1291   }
1292   {
1293     \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1294     {
1295       \int_set:Nn \l_@@_last_col_int
1296       { \use:c { @@_last_col_ \l_@@_name_str } }
1297     }
1298   }
1299 }

```

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

```

1300   \@@_pre_array_ii:

```

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

```

1301   \box_clear_new:N \l_@@_the_array_box

```

If the user has loaded `nicematrix` with the option `define-L-C-R`, he will be able to use `L`, `C` and `R` instead of `l`, `c` and `r` in the preambles of the environments of `nicematrix` (it's a compatibility mode since `L`, `C` and `R` were mandatory before version 5.0).

```

1302   \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:

```

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

```

1303   \@@_construct_preamble:

```

Now, the preamble is constructed in `\g_@@_preamble_tl`

We compute the width of both delimiters. We remember that, when the environment `{NiceArray}` is used, it's possible to specify the delimiters in the preamble (eg `[ccc]`).

```

1304   \dim_zero_new:N \l_@@_left_delim_dim
1305   \dim_zero_new:N \l_@@_right_delim_dim
1306   \bool_if:NTF \g_@@_NiceArray_bool
1307   {
1308     \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1309     \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1310   }
1311   {

```

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

```

1312     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \g_@@_left_delim_tl $ }
1313     \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1314     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \g_@@_right_delim_tl $ }
1315     \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1316   }

```

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).

```

1317   \hbox_set:Nw \l_@@_the_array_box

```

```

1318 \skip_horizontal:N \l_@@_left_margin_dim
1319 \skip_horizontal:N \l_@@_extra_left_margin_dim
1320 \c_math_toggle_token
1321 \bool_if:NTF \l_@@_light_syntax_bool
1322 { \use:c { @@-light-syntax } }
1323 { \use:c { @@-normal-syntax } }
1324 }

```

The following command `\@@_pre_array_i:w` will be used when the keyword `\CodeBefore` is present at the beginning of the environment.

```

1325 \cs_new_protected:Npn \@@_pre_array_i:w #1 \Body
1326 {
1327   \tl_put_right:Nn \l_@@_code_before_tl { #1 }
1328   \bool_set_true:N \l_@@_code_before_bool

```

We go on with `\@@_pre_array:` which will (among other) execute the `\CodeBefore` (specified in the key `code-before` or after the keyword `\CodeBefore`). By definition, the `\CodeBefore` must be executed before the body of the array...

```

1329 \@@_pre_array:
1330 }

```

## The environment `{NiceArrayWithDelims}`

```

1331 \NewDocumentEnvironment { NiceArrayWithDelims }
1332 { m m O { } m ! O { } t \CodeBefore }
1333 {
1334   \@@_provide_pgfsyspdfmark:
1335   \bool_if:NT \c_@@_footnote_bool \savenotes

```

The aim of the following `\bgroup` (the corresponding `\egroup` is, of course, at the end of the environment) is to be able to put an exposant to a matrix in a mathematical formula.

```

1336 \bgroup

1337 \tl_gset:Nn \g_@@_left_delim_tl { #1 }
1338 \tl_gset:Nn \g_@@_right_delim_tl { #2 }
1339 \tl_gset:Nn \g_@@_preamble_tl { #4 }

1340 \int_gzero:N \g_@@_block_box_int
1341 \dim_zero:N \g_@@_width_last_col_dim
1342 \dim_zero:N \g_@@_width_first_col_dim
1343 \bool_gset_false:N \g_@@_row_of_col_done_bool
1344 \str_if_empty:NT \g_@@_name_env_str
1345 { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1346 \@@_adapt_S_column:
1347 \bool_if:NTF \l_@@_NiceTabular_bool
1348   \mode_leave_vertical:
1349   \@@_test_if_math_mode:
1350 \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
1351 \bool_set_true:N \l_@@_in_env_bool

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>55</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.

```

1352 \cs_gset_eq:NN \@@_old_CT@arc@ \CT@arc@

```

---

<sup>55</sup>e.g. `\color[rgb]{0.5,0.5,0}`

We deactivate Tikz externalization because we will use PGF pictures with the options `overlay` and `remember picture` (or equivalent forms). We deactivate with `\tikzexternaldisable` and not with `\tikzset{external/export=false}` which is *not* equivalent.

```

1353 \cs_if_exist:NT \tikz@library@external@loaded
1354 {
1355     \tikzexternaldisable
1356     \cs_if_exist:NT \ifstandalone
1357     { \tikzset { external / optimize = false } }
1358 }

```

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

```

1359 \int_gincr:N \g_@@_env_int
1360 \bool_if:NF \l_@@_block_auto_columns_width_bool
1361 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

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

```

1362 \seq_gclear:N \g_@@_blocks_seq
1363 \seq_gclear:N \g_@@_pos_of_blocks_seq

```

In fact, the sequence `\g_@@_pos_of_blocks_seq` will also contain the positions of the cells with a `\diagbox`.

```

1364 \seq_gclear:N \g_@@_pos_of_stroken_blocks_seq
1365 \seq_gclear:N \g_@@_pos_of_xdots_seq
1366 \tl_if_exist:cT { g_@@_code_before_ \int_use:N \g_@@_env_int _ t1 }
1367 {
1368     \bool_set_true:N \l_@@_code_before_bool
1369     \exp_args:NNv \tl_put_right:Nn \l_@@_code_before_t1
1370     { g_@@_code_before_ \int_use:N \g_@@_env_int _ t1 }
1371 }

```

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`.

```

1372 \bool_if:NTF \g_@@_NiceArray_bool
1373 { \keys_set:nn { NiceMatrix / NiceArray } }
1374 { \keys_set:nn { NiceMatrix / pNiceArray } }
1375 { #3 , #5 }

```

```

1376 \tl_if_empty:NF \l_@@_rules_color_t1
1377 { \exp_after:wN @@_set_CT@arc@: \l_@@_rules_color_t1 \q_stop }
1378 % \bigskip

```

The argument `#6` is the last argument of `{NiceArrayWithDelims}`. With that argument of type “`t \CodeBefore`”, we test whether there is the keyword `\CodeBefore` at the beginning of the environment. If that keyword is present, we have now to extract all the content between that keyword `\CodeBefore` and the (other) keyword `\Body`. It's the job that will do the command `\@@_pre_array_i:w`. After that job, the command `\@@_pre_array_i:w` will go on with `\@@_pre_array:`.

```

1379 \IfBooleanTF { #6 } \@@_pre_array_i:w \@@_pre_array:
1380 }
1381 {
1382     \bool_if:NTF \l_@@_light_syntax_bool
1383     { \use:c { end @@-light-syntax } }
1384     { \use:c { end @@-normal-syntax } }
1385     \c_math_toggle_token
1386     \skip_horizontal:N \l_@@_right_margin_dim
1387     \skip_horizontal:N \l_@@_extra_right_margin_dim
1388     \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).

```

1389 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1390 {
1391   \bool_if:NF \l_@@_last_row_without_value_bool
1392   {
1393     \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1394     {
1395       \@@_error:n { Wrong~last~row }
1396       \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1397     }
1398   }
1399 }

```

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>56</sup>

```

1400 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1401 \bool_if:nTF \g_@@_last_col_found_bool
1402 { \int_gdecr:N \c@jCol }
1403 {
1404   \int_compare:nNnT \l_@@_last_col_int > { -1 }
1405   { \@@_error:n { last~col~not~used } }
1406 }

```

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

```

1407 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1408 \int_compare:nNnT \l_@@_last_row_int > { -1 } { \int_gdecr:N \c@iRow }

```

**Now, we begin the real construction in the output flow of TeX.** First, we take into account a potential “first column” (we remind that this “first column” has been constructed in an overlapping position and that we have computed its width in `\g_@@_width_first_col_dim`: see p. 105).

```

1409 \int_compare:nNnT \l_@@_first_col_int = 0
1410 {
1411   \skip_horizontal:N \col@sep
1412   \skip_horizontal:N \g_@@_width_first_col_dim
1413 }

```

The construction of the real box is different when `\g_@@_NiceArray_bool` is true (`{NiceArray}` or `{NiceTabular}`) and in the other environments because, in `{NiceArray}` or `{NiceTabular}`, we have no delimiter to put (but we have tabular notes to put). We begin with this case.

Remark that, in all cases, `@@_use_arraybox_with_notes_c:` is used.

```

1414 \bool_if:NTF \g_@@_NiceArray_bool
1415 {
1416   \str_case:VnF \l_@@_baseline_tl
1417   {
1418     b \@@_use_arraybox_with_notes_b:
1419     c \@@_use_arraybox_with_notes_c:
1420   }
1421   \@@_use_arraybox_with_notes:
1422 }

```

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).

```

1423 {
1424   \int_compare:nNnTF \l_@@_first_row_int = 0
1425   {
1426     \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1427     \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1428   }
1429   { \dim_zero:N \l_tmpa_dim }

```

---

<sup>56</sup>We remind that the potential “first column” (exterior) has the number 0.

We compute `\l_tmpb_dim` which is the total height of the “last row” below the array (when the key `last-row` is used). A value of `-2` for `\l_@@_last_row_int` means that there is no “last row”.<sup>57</sup>

```

1430 \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1431 {
1432     \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1433     \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1434 }
1435 { \dim_zero:N \l_tmpb_dim }
1436 \hbox_set:Nn \l_tmpa_box
1437 {
1438     \c_math_toggle_token
1439     \tl_if_empty:NF \l_@@_delimiters_color_tl
1440     { \color { \l_@@_delimiters_color_tl } }
1441     \exp_after:wN \left \g_@@_left_delim_tl
1442     \vcenter
1443     {

```

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.

```

1444     \skip_vertical:N -\l_tmpa_dim
1445     \hbox
1446     {
1447         \bool_if:NTF \l_@@_NiceTabular_bool
1448         { \skip_horizontal:N -\tabcolsep }
1449         { \skip_horizontal:N -\arraycolsep }
1450         \@@_use_arraybox_with_notes_c:
1451         \bool_if:NTF \l_@@_NiceTabular_bool
1452         { \skip_horizontal:N -\tabcolsep }
1453         { \skip_horizontal:N -\arraycolsep }
1454     }

```

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

```

1455     \skip_vertical:N -\l_tmpb_dim
1456 }

```

Curiously, we have to put again the following specification of color. Otherwise, with XeLaTeX (and not with the other engines), the closing delimiter is not colored.

```

1457     \tl_if_empty:NF \l_@@_delimiters_color_tl
1458     { \color { \l_@@_delimiters_color_tl } }
1459     \exp_after:wN \right \g_@@_right_delim_tl
1460     \c_math_toggle_token
1461 }

```

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 `delimiters/max-width` is used.

```

1462     \bool_if:NTF \l_@@_delimiters_max_width_bool
1463     {
1464         \@@_put_box_in_flow_bis:nn
1465         \g_@@_left_delim_tl \g_@@_right_delim_tl
1466     }
1467     \@@_put_box_in_flow:
1468 }

```

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. 106).

```

1469     \bool_if:NT \g_@@_last_col_found_bool
1470     {
1471         \skip_horizontal:N \g_@@_width_last_col_dim
1472         \skip_horizontal:N \col@sep
1473     }
1474     \bool_if:NF \l_@@_Matrix_bool

```

---

<sup>57</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).



```

1475     {
1476       \int_compare:nNnT \c@jCol < \g_@@_static_num_of_col_int
1477       { \@@_error:n { columns-not-used } }
1478     }
1479     \group_begin:
1480     \globaldefs = 1
1481     \@@_msg_redirect_name:nn { columns-not-used } { error }
1482     \group_end:
1483     \@@_after_array:

```

The aim of the following `\egroup` (the corresponding `\bgroup` is, of course, at the beginning of the environment) is to be able to put an exposant to a matrix in a mathematical formula.

```

1484     \egroup
1485     \bool_if:NT \c_@@_footnote_bool \endsavenotes
1486   }

```

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

## We construct the preamble of the array

The transformation of the preamble is an operation in several steps.

The preamble given by the final user is in `\g_@@_preamble_tl` and the modified version will be stored in `\g_@@_preamble_tl` also.

```

1487 \cs_new_protected:Npn \@@_construct_preamble:
1488 {

```

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

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

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

```

1489   \group_begin:

```

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

```

1490   \bool_if:NF \l_@@_Matrix_bool
1491   {
1492     \@@_newcolumntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1493     \@@_newcolumntype W [ 2 ] { \@@_W: { ##1 } { ##2 } }

```

First, we have to store our preamble in the token register `\@temptokena` (those “token registers” are *not* supported by `expl3`).

```

1494     \exp_args:NV \@temptokena \g_@@_preamble_tl

```

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

```

1495     \@tempswatrue

```

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

```

1496     \@whilesw \if@tempswa \fi { \@tempswafalse \the \NC@list }

```

Now, we have to “patch” that preamble by transforming some columns. We will insert in the TeX flow the preamble in its actual form (that is to say after the “expansion”) following by a marker `\q_stop` and we will consume these tokens constructing the (new form of the) preamble in `\g_@@_preamble_tl`. This is done recursively with the command `\@@_patch_preamble:n`. In the same time, we will count the columns with the counter `\c@jCol`.

```

1497 \int_gzero_new:N \c@jCol
1498 \tl_gclear:N \g_@@_preamble_tl
1499 \tl_if_eq:NnTF \l_@@_vlines_clist { all }
1500 {
1501   \tl_gset:Nn \g_@@_preamble_tl
1502   { ! { \skip_horizontal:N \arrayrulewidth } }
1503 }
1504 {
1505   \clist_if_in:NnT \l_@@_vlines_clist 1
1506   {
1507     \tl_gset:Nn \g_@@_preamble_tl
1508     { ! { \skip_horizontal:N \arrayrulewidth } }
1509   }
1510 }

```

The sequence `\g_@@_cols_vlsim_seq` will contain the numbers of the columns where you will to have to draw vertical lines in the potential sub-matrices (hence the name `vlism`).

```

1511 \seq_clear:N \g_@@_cols_vlism_seq

```

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

```

1512 \int_zero:N \l_tmpa_int

```

We will raise `\l_tmpa_bool` if we are in `{NiceArray}` and that the first letter of the preamble is a opening delimiter (`[`, `(` of `\{`). In that case, the environment `{NiceArray}` will be transformed to an environment of the form `{xNiceArray}` after the construction of the preamble.

```

1513 \bool_set_false:N \l_tmpa_bool

```

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

```

1514 \exp_after:wN \@@_patch_preamble:n \the \@temptokena \q_stop
1515 \bool_if:NT \l_tmpa_bool { \bool_gset_false:N \g_@@_NiceArray_bool }
1516 \int_gset_eq:NN \g_@@_static_num_of_col_int \c@jCol
1517 }

```

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

```

1518 \bool_if:NT \l_@@_colortbl_like_bool
1519 {
1520   \regex_replace_all:NnN
1521   \c_@@_columncolor_regex
1522   { \c { @@_columncolor_preamble } }
1523   \g_@@_preamble_tl
1524 }

```

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

```

1525 \int_compare:nNnTF \l_@@_first_col_int = 0
1526 { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1527 {
1528   \bool_lazy_all:nT
1529   {
1530     \g_@@_NiceArray_bool
1531     { \bool_not_p:n \l_@@_NiceTabular_bool }
1532     { \tl_if_empty_p:N \l_@@_vlines_clist }
1533     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1534   }
1535   { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1536 }
1537 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1538 { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }
1539 {
1540   \bool_lazy_all:nT
1541   {
1542     \g_@@_NiceArray_bool
1543     { \bool_not_p:n \l_@@_NiceTabular_bool }
1544     { \tl_if_empty_p:N \l_@@_vlines_clist }
1545     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }

```

```

1546     }
1547     { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1548 }

```

We add a last column to raise a good error message when the user put more columns than allowed by its preamble. However, for technical reasons, it's not possible to do that in `{NiceTabular*}` (`\l_@@_tabular_width_dim=0pt`).

```

1549 \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1550 {
1551     \tl_gput_right:Nn \g_@@_preamble_tl
1552     { > { \@@_error_too_much_cols: } 1 }
1553 }

```

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

```

1554 \group_end:
1555 }

```

```

1556 \cs_new_protected:Npn \@@_patch_preamble:n #1
1557 {
1558     \str_case:nnF { #1 }
1559     {
1560         c { \@@_patch_preamble_i:n #1 }
1561         l { \@@_patch_preamble_i:n #1 }
1562         r { \@@_patch_preamble_i:n #1 }
1563         > { \@@_patch_preamble_ii:nn #1 }
1564         ! { \@@_patch_preamble_ii:nn #1 }
1565         @ { \@@_patch_preamble_ii:nn #1 }
1566         | { \@@_patch_preamble_iii:n #1 }
1567         p { \@@_patch_preamble_iv:nnn t #1 }
1568         m { \@@_patch_preamble_iv:nnn c #1 }
1569         b { \@@_patch_preamble_iv:nnn b #1 }
1570         \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1571         \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1572         \@@_true_c: { \@@_patch_preamble_vi:n #1 }
1573         ( { \@@_patch_preamble_vii:n #1 }
1574         [ { \@@_patch_preamble_vii:n #1 }
1575         \{ { \@@_patch_preamble_vii:n #1 }
1576         ) { \@@_patch_preamble_viii:nn #1 }
1577         ] { \@@_patch_preamble_viii:nn #1 }
1578         \} { \@@_patch_preamble_viii:nn #1 }
1579         C { \@@_error:nn { old-column-type } #1 }
1580         L { \@@_error:nn { old-column-type } #1 }
1581         R { \@@_error:nn { old-column-type } #1 }
1582         \q_stop { }
1583     }
1584     {
1585         \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1586         { \@@_patch_preamble_xi:n #1 }
1587         {
1588             \str_if_eq:VnTF \l_@@_letter_vlism_tl { #1 }
1589             {
1590                 \seq_gput_right:Nx \g_@@_cols_vlism_seq
1591                 { \int_eval:n { \c@jCol + 1 } }
1592                 \tl_gput_right:Nx \g_@@_preamble_tl
1593                 { \exp_not:N ! { \skip_horizontal:N \arrayrulewidth } }
1594                 \@@_patch_preamble:n
1595             }
1596             {
1597                 \bool_lazy_and:nnTF
1598                 { \str_if_eq_p:nn { : } { #1 } }
1599                 \c_@@_arydshln_loaded_bool
1600                 {

```

```

1601         \tl_gput_right:Nn \g_@@_preamble_tl { : }
1602         \@@_patch_preamble:n
1603     }
1604     { \@@_fatal:nn { unknown-column-type } { #1 } }
1605 }
1606 }
1607 }
1608 }

```

For c, l and r

```

1609 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1610 {
1611     \tl_gput_right:Nn \g_@@_preamble_tl
1612     {
1613         > { \@@_Cell: \tl_set:Nn \l_@@_cell_type_tl { #1 } }
1614         #1
1615         < \@@_end_Cell:
1616     }

```

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

```

1617     \int_gincr:N \c@jCol
1618     \@@_patch_preamble_x:n
1619 }

```

For >, ! and @

```

1620 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1621 {
1622     \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
1623     \@@_patch_preamble:n
1624 }

```

For |

```

1625 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1626 {

```

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

```

1627     \int_incr:N \l_tmpa_int
1628     \@@_patch_preamble_iii_i:n
1629 }

1630 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1631 {
1632     \str_if_eq:nnTF { #1 } |
1633     { \@@_patch_preamble_iii:n | }
1634     {
1635         \tl_gput_right:Nx \g_@@_preamble_tl
1636         {
1637             \exp_not:N !
1638             {
1639                 \skip_horizontal:n
1640                 {
1641                     \dim_eval:n
1642                     {
1643                         \arrayrulewidth * \l_tmpa_int
1644                         + \doublerulesep * ( \l_tmpa_int - 1 )
1645                     }
1646                 }
1647             }
1648         }
1649         \tl_gput_right:Nx \g_@@_internal_code_after_tl
1650         { \@@_vline:nn { \@@_succ:n \c@jCol } { \int_use:N \l_tmpa_int } }
1651         \int_zero:N \l_tmpa_int
1652         \@@_patch_preamble:n #1
1653     }
1654 }

```

For p, m and b

```

1655 \cs_new_protected:Npn \@@_patch_preamble_iv:nnn #1 #2 #3
1656 {
1657   \tl_gput_right:Nn \g_@@_preamble_tl
1658   {
1659     > {
1660       \@@_Cell:
1661       \begin { minipage } [ #1 ] { \dim_eval:n { #3 } }
1662       \mode_leave_vertical:
1663       \arraybackslash
1664       \vrule height \box_ht:N \@@arstrutbox depth 0 pt width 0 pt
1665     }
1666     c
1667     < {
1668       \vrule height 0 pt depth \box_dp:N \@@arstrutbox width 0 pt
1669       \end { minipage }
1670       \@@_end_Cell:
1671     }
1672   }

```

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

```

1673   \int_gincr:N \c@jCol
1674   \@@_patch_preamble_x:n
1675 }

```

For w and W

```

1676 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1677 {
1678   \tl_gput_right:Nn \g_@@_preamble_tl
1679   {
1680     > {
1681       \hbox_set:Nw \l_@@_cell_box
1682       \@@_Cell:
1683       \tl_set:Nn \l_@@_cell_type_tl { #3 }
1684     }
1685     c
1686     < {
1687       \@@_end_Cell:
1688       #1
1689       \hbox_set_end:
1690       \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1691       \@@_adjust_size_box:
1692       \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
1693     }
1694   }

```

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

```

1695   \int_gincr:N \c@jCol
1696   \@@_patch_preamble_x:n
1697 }

```

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

```

1698 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1699 {
1700   \tl_gput_right:Nn \g_@@_preamble_tl { c }

```

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

```

1701   \int_gincr:N \c@jCol
1702   \@@_patch_preamble_x:n
1703 }

```

For (, [ and \{.

```

1704 \cs_new_protected:Npn \@@_patch_preamble_vii:n #1
1705 {

```

```

1706 \bool_if:NT \l_@@_small_bool
1707 { \@@_fatal:n { Delimiter~with~small } }

```

If we are before the column 1 and not in {NiceArray}, we reserve space for the left delimiter.

```

1708 \int_compare:nNnTF \c@jCol = \c_zero_int
1709 {
1710   \bool_if:NTF \g_@@_NiceArray_bool
1711   {

```

In that case, in fact, the first letter of the preamble must be considered as the left delimiter of the array.

```

1712 \tl_gset:Nn \g_@@_left_delim_tl { #1 }
1713 \tl_gset:Nn \g_@@_right_delim_tl { . }

```

We raise the boolean \l\_tmpa\_bool, which means that the environment {NiceArray} will be transformed to {xNiceArray}.

```

1714 \bool_set_true:N \l_tmpa_bool
1715 \@@_patch_preamble:n
1716 }
1717 {
1718   \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } }
1719   \tl_gput_right:Nx \g_@@_internal_code_after_tl
1720   { \@@_delimiter:nnn #1 { \@@_succ:n \c@jCol } \c_true_bool }
1721   \@@_patch_preamble_vii_i:n
1722 }
1723 }
1724 {
1725   \tl_gput_right:Nx \g_@@_internal_code_after_tl
1726   { \@@_delimiter:nnn #1 { \@@_succ:n \c@jCol } \c_true_bool }
1727   \@@_patch_preamble_vii_i:n
1728 }
1729 }

```

```

1730 \cs_new_protected:Npn \@@_patch_preamble_vii_i:n #1
1731 {
1732   \tl_if_in:nnTF { ( [ \{ ) ] \} } { #1 }
1733   {
1734     \@@_error:nn { delimiter~after~opening } { #1 }
1735     \@@_patch_preamble:n
1736   }
1737   { \@@_patch_preamble:n #1 }
1738 }

```

For ), ] and \}. We have two arguments for the following command because we directly read the following letter in the preamble (we have to see whether we have a opening delimiter following and we also have to see whether we are at the end of the preamble because, in that case, our letter must be considered as the right delimiter of the environment).

```

1739 \cs_new_protected:Npn \@@_patch_preamble_viii:nn #1 #2
1740 {
1741   \bool_if:NT \l_@@_small_bool
1742   { \@@_fatal:n { Delimiter~with~small } }
1743   \tl_if_in:nnTF { ) ] \} } { #2 }
1744   {
1745     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1746     { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
1747     \@@_error:nn { double~closing~delimiter } { #2 }
1748     \@@_patch_preamble:n
1749   }
1750   {
1751     \tl_if_eq:nnTF { \q_stop } { #2 }
1752     {
1753       \bool_if:NTF \g_@@_NiceArray_bool
1754       {

```

```

1755         \tl_gset:Nn \g_@@_right_delim_tl { #1 }
1756         \bool_set_true:N \l_tmpa_bool
1757     }
1758     {
1759         \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } }
1760         \tl_gput_right:Nx \g_@@_internal_code_after_tl
1761             { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
1762         \@@_patch_preamble:n #2
1763     }
1764 }
1765 {
1766     \tl_if_in:nnT { ( [ \{ } { #2 }
1767         { \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } } }
1768     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1769         { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
1770     \@@_patch_preamble:n #2
1771 }
1772 }
1773 }

```

```

1774 \cs_new_protected:Npn \@@_patch_preamble_xi:n #1

```

```

1775 {
1776     \tl_gput_right:Nn \g_@@_preamble_tl
1777         { ! { \skip_horizontal:N 2\l_@@_radius_dim } }

```

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

```

1778     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1779         { \@@_vdottedline:n { \int_use:N \c@jCol } }
1780     \@@_patch_preamble:n
1781 }

```

After a specifier of column, we have to test whether there is one or several `<{..}` because, after those potential `<{...}`, we have to insert `!\skip_horizontal:N ...` when the key `vlines` is used.

```

1782 \cs_new_protected:Npn \@@_patch_preamble_x:n #1
1783 {
1784     \str_if_eq:nnTF { #1 } { < }
1785         \@@_patch_preamble_ix:n
1786         {
1787             \tl_if_eq:NnTF \l_@@_vlines_clist { all }
1788             {
1789                 \tl_gput_right:Nn \g_@@_preamble_tl
1790                     { ! { \skip_horizontal:N \arrayrulewidth } }
1791             }
1792             {
1793                 \exp_args:NNx
1794                 \clist_if_in:NnT \l_@@_vlines_clist { \@@_succ:n \c@jCol }
1795                 {
1796                     \tl_gput_right:Nn \g_@@_preamble_tl
1797                         { ! { \skip_horizontal:N \arrayrulewidth } }
1798                 }
1799             }
1800             \@@_patch_preamble:n { #1 }
1801         }
1802 }
1803 \cs_new_protected:Npn \@@_patch_preamble_ix:n #1
1804 {
1805     \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
1806     \@@_patch_preamble_x:n
1807 }

```

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`).

```

1808 \cs_new_protected:Npn \@@_put_box_in_flow:
1809 {
1810   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1811   \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1812   \tl_if_eq:NnTF \l_@@_baseline_tl { c }
1813     { \box_use_drop:N \l_tmpa_box }
1814     \@@_put_box_in_flow_i:
1815 }

```

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

```

1816 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1817 {
1818   \pgfpicture
1819     \@@_qpoint:n { row - 1 }
1820     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1821     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1822     \dim_gadd:Nn \g_tmpa_dim \pgf@y
1823     \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).

```

1824   \str_if_in:NnTF \l_@@_baseline_tl { line- }
1825   {
1826     \int_set:Nn \l_tmpa_int
1827     {
1828       \str_range:Nnn
1829         \l_@@_baseline_tl
1830         6
1831         { \tl_count:V \l_@@_baseline_tl }
1832     }
1833     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1834   }
1835   {
1836     \str_case:VnF \l_@@_baseline_tl
1837     {
1838       { t } { \int_set:Nn \l_tmpa_int 1 }
1839       { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1840     }
1841     { \int_set:Nn \l_tmpa_int \l_@@_baseline_tl }
1842     \bool_lazy_or:nnT
1843     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1844     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1845     {
1846       \@@_error:n { bad~value~for~baseline }
1847       \int_set:Nn \l_tmpa_int 1
1848     }
1849     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

We take into account the position of the mathematical axis.

```

1850     \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
1851   }
1852   \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

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

```

1853   \endpgfpicture
1854   \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1855   \box_use_drop:N \l_tmpa_box
1856 }

```



The following command is *always* used by `{NiceArrayWithDelims}` (even if, in fact, there is no tabular notes: in fact, it's not possible to know whether there is tabular notes or not before the composition of the blocks).

```
1857 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
1858 {
```

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

```
1859 \begin { minipage } [ t ] { \box_wd:N \l_@@_the_array_box }
1860 \box_use_drop:N \l_@@_the_array_box
```

We have to draw the blocks right now because there may be tabular notes in some blocks (which are not mono-column: the blocks which are mono-column have been composed in boxes yet)... and we have to create (potentially) the extra nodes before creating the blocks since there are `medium` nodes to create for the blocks.

```
1861 \@@_create_extra_nodes:
1862 \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
1863 \bool_lazy_or:nnT
1864 { \int_compare_p:nNn \c@tabularnote > 0 }
1865 { ! \tl_if_empty_p:V \l_@@_tabularnote_tl }
1866 \@@_insert_tabularnotes:
1867 \end { minipage }
1868 }
```

```
1869 \cs_new_protected:Npn \@@_insert_tabularnotes:
1870 {
1871 \skip_vertical:N 0.65ex
```

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

```
1872 \group_begin:
1873 \l_@@_notes_code_before_tl
1874 \tl_if_empty:NF \l_@@_tabularnote_tl { \l_@@_tabularnote_tl \par }
```

We compose the tabular notes with a list of `enumitem`. The `\strut` and the `\unskip` are designed to give the ability to put a `\bottomrule` at the end of the notes with a good vertical space.

```
1875 \int_compare:nNnT \c@tabularnote > 0
1876 {
1877 \bool_if:NTF \l_@@_notes_para_bool
1878 {
1879 \begin { tabularnotes* }
1880 \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1881 \end { tabularnotes* }
```

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

```
1882 \par
1883 }
1884 {
1885 \tabularnotes
1886 \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1887 \endtabularnotes
1888 }
1889 }
1890 \unskip
1891 \group_end:
1892 \bool_if:NT \l_@@_notes_bottomrule_bool
1893 {
1894 \bool_if:NTF \c_@@_booktabs_loaded_bool
1895 {
```

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

```
1896 \skip_vertical:N \aboverulesep
```

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

```

1897         { \CT@arc@ \hrule height \heavyrulewidth }
1898     }
1899     { \@_error:n { bottomrule~without~booktabs } }
1900 }
1901 \l_@@_notes_code_after_tl
1902 \seq_gclear:N \g_@@_tabularnotes_seq
1903 \int_gzero:N \c@tabularnote
1904 }

```

The case of baseline equal to b. Remember that, when the key b is used, the {array} (of array) is constructed with the option t (and not b). Now, we do the translation to take into account the option b.

```

1905 \cs_new_protected:Npn \@_use_arraybox_with_notes_b:
1906 {
1907     \pgfpicture
1908     \@_qpoint:n { row - 1 }
1909     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1910     \@_qpoint:n { row - \int_use:N \c@iRow - base }
1911     \dim_gsub:Nn \g_tmpa_dim \pgf@y
1912     \endpgfpicture
1913     \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1914     \int_compare:nNnT \l_@@_first_row_int = 0
1915     {
1916         \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1917         \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1918     }
1919     \box_move_up:nn \g_tmpa_dim { \hbox { \@_use_arraybox_with_notes_c: } }
1920 }

```

Now, the general case.

```

1921 \cs_new_protected:Npn \@_use_arraybox_with_notes:
1922 {

```

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

```

1923     \tl_if_eq:NnT \l_@@_baseline_tl { t }
1924     { \tl_set:Nn \l_@@_baseline_tl { 1 } }

```

Now, we convert the value of \l\_@@\_baseline\_tl (which should represent an integer) to an integer stored in \l\_tmpa\_int.

```

1925     \pgfpicture
1926     \@_qpoint:n { row - 1 }
1927     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1928     \str_if_in:NnTF \l_@@_baseline_tl { line- }
1929     {
1930         \int_set:Nn \l_tmpa_int
1931         {
1932             \str_range:Nnn
1933             \l_@@_baseline_tl
1934             6
1935             { \tl_count:V \l_@@_baseline_tl }
1936         }
1937         \@_qpoint:n { row - \int_use:N \l_tmpa_int }
1938     }
1939     {
1940         \int_set:Nn \l_tmpa_int \l_@@_baseline_tl
1941         \bool_lazy_or:nnT
1942         { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1943         { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1944         {
1945             \@_error:n { bad~value~for~baseline }
1946             \int_set:Nn \l_tmpa_int 1
1947         }
1948         \@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

```

1949     }
1950     \dim_gsub:Nn \g_tmpa_dim \pgf@y
1951     \endpgfpicture
1952     \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1953     \int_compare:nNnT \l_@@_first_row_int = 0
1954     {
1955         \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1956         \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1957     }
1958     \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
1959 }

```

The command `\@@_put_box_in_flow_bis:` is used when the option `delimiters/max-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.

```

1960 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1961 {

```

We will compute the real width of both delimiters used.

```

1962     \dim_zero_new:N \l_@@_real_left_delim_dim
1963     \dim_zero_new:N \l_@@_real_right_delim_dim
1964     \hbox_set:Nn \l_tmpb_box
1965     {
1966         \c_math_toggle_token
1967         \left #1
1968         \vcenter
1969         {
1970             \vbox_to_ht:nn
1971             { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1972             { }
1973         }
1974         \right .
1975         \c_math_toggle_token
1976     }
1977     \dim_set:Nn \l_@@_real_left_delim_dim
1978     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
1979     \hbox_set:Nn \l_tmpb_box
1980     {
1981         \c_math_toggle_token
1982         \left .
1983         \vbox_to_ht:nn
1984         { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1985         { }
1986         \right #2
1987         \c_math_toggle_token
1988     }
1989     \dim_set:Nn \l_@@_real_right_delim_dim
1990     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }

```

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

```

1991     \skip_horizontal:N \l_@@_left_delim_dim
1992     \skip_horizontal:N -\l_@@_real_left_delim_dim
1993     \@@_put_box_in_flow:
1994     \skip_horizontal:N \l_@@_right_delim_dim
1995     \skip_horizontal:N -\l_@@_real_right_delim_dim
1996 }

```

The construction of the array in the environment `{NiceArrayWithDelims}` is, in fact, done by the environment `{@@-light-syntax}` or by the environment `{@@-normal-syntax}` (whether the option `light-syntax` is in force or not). When the key `light-syntax` is not used, the construction is a standard environment (and, thus, it's possible to use verbatim in the array).

```

1997 \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).

```

1998 {
1999   \peek_meaning_ignore_spaces:NTF \end \@@_analyze_end:Nn

```

Here is the call to `\array` (we have a dedicated macro `\@@_array:` because of compatibility with the classes `revtex4-1` and `revtex4-2`).

```

2000   { \exp_args:NV \@@_array: \g_@@_preamble_tl }
2001 }
2002 {
2003   \@@_create_col_nodes:
2004   \endarray
2005 }

```

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

```

2006 \NewDocumentEnvironment { @@-light-syntax } { b }
2007 {

```

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`.

```

2008   \tl_if_empty:nT { #1 } { \@@_fatal:n { empty~environment } }
2009   \tl_map_inline:nn { #1 }
2010   {
2011     \str_if_eq:nnT { ##1 } { & }
2012     { \@@_fatal:n { ampersand~in~light-syntax } }
2013     \str_if_eq:nnT { ##1 } { \ }
2014     { \@@_fatal:n { double-backslash~in~light-syntax } }
2015   }

```

Now, you extract the `\CodeAfter` of the body of the environment. Maybe, there is no command `\CodeAfter` in the body. That's why you put a marker `\CodeAfter` after `#1`. If there is yet a `\CodeAfter` in `#1`, this second (or third...) `\CodeAfter` will be caught in the value of `\g_nicematrix_code_after_tl`. That doesn't matter because `\CodeAfter` will be set to *no-op* before the execution of `\g_nicematrix_code_after_tl`.

```

2016   \@@_light_syntax_i #1 \CodeAfter \q_stop
2017 }

```

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`.

```

2018 { }
2019 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
2020 {
2021   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #2 }

```

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

```

2022   \seq_gclear_new:N \g_@@_rows_seq
2023   \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
2024   \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.

```

2025   \int_compare:nNnT \l_@@_last_row_int = { -1 }
2026   { \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`).

```

2027   \exp_args:NV \@@_array: \g_@@_preamble_tl

```

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

```

2028   \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
2029   \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
2030   \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
2031   \@@_create_col_nodes:
2032   \endarray
2033 }

2034 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
2035 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }

2036 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
2037 {
2038   \seq_gclear_new:N \g_@@_cells_seq
2039   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
2040   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
2041   \l_tmpa_tl
2042   \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
2043 }

```

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).

```

2044 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
2045 {
2046   \str_if_eq:VnT \g_@@_name_env_str { #2 }
2047   { \@@_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.

```

2048   \end { #2 }
2049 }

```

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).

```

2050 \cs_new:Npn \@@_create_col_nodes:
2051 {
2052   \crrc
2053   \int_compare:nNnT \l_@@_first_col_int = 0
2054   {
2055     \omit
2056     \hbox_overlap_left:n
2057     {
2058       \bool_if:NT \l_@@_code_before_bool
2059       { \pgfsys@markposition { \@@_env: - col - 0 } }
2060       \pgfpicture
2061       \pgfrememberpicturerepositiononpagetrue
2062       \pgfcoordinate { \@@_env: - col - 0 } \pgfpointorigin
2063       \str_if_empty:NF \l_@@_name_str
2064       { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
2065       \endpgfpicture
2066       \skip_horizontal:N 2\col@sep
2067       \skip_horizontal:N \g_@@_width_first_col_dim
2068     }
2069     &
2070   }
2071   \omit

```

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

```

2072   \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`).

```

2073 \int_compare:nNnTF \l_@@_first_col_int = 0
2074 {
2075   \bool_if:NT \l_@@_code_before_bool
2076   {
2077     \hbox
2078     {
2079       \skip_horizontal:N -0.5\arrayrulewidth
2080       \pgfsys@markposition { \@@_env: - col - 1 }
2081       \skip_horizontal:N 0.5\arrayrulewidth
2082     }
2083   }
2084   \pgfpicture
2085   \pgfrememberpicturepositiononpagetrue
2086   \pgfcoordinate { \@@_env: - col - 1 }
2087   { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2088   \str_if_empty:NF \l_@@_name_str
2089   { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
2090   \endpgfpicture
2091 }
2092 {
2093   \bool_if:NT \l_@@_code_before_bool
2094   {
2095     \hbox
2096     {
2097       \skip_horizontal:N 0.5\arrayrulewidth
2098       \pgfsys@markposition { \@@_env: - col - 1 }
2099       \skip_horizontal:N -0.5\arrayrulewidth
2100     }
2101   }
2102   \pgfpicture
2103   \pgfrememberpicturepositiononpagetrue
2104   \pgfcoordinate { \@@_env: - col - 1 }
2105   { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
2106   \str_if_empty:NF \l_@@_name_str
2107   { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
2108   \endpgfpicture
2109 }

```

We compute in `\g_tmpa_skip` the common width of the columns (it's a skip and not a dimension). We use a global variable because we are in a cell of an `\halign` and because we have to use this variable in other cells (of the same row). The affectation of `\g_tmpa_skip`, like all the affectations, must be done after the `\omit` of the cell.

We give a default value for `\g_tmpa_skip` (0 pt plus 1 fill) but it will just after be erased by a fixed value in the concerned cases.

```

2110 \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
2111 \bool_if:NF \l_@@_auto_columns_width_bool
2112 { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
2113 {
2114   \bool_lazy_and:nnTF
2115   \l_@@_auto_columns_width_bool
2116   { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
2117   { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
2118   { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
2119   \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
2120 }
2121 \skip_horizontal:N \g_tmpa_skip
2122 \hbox
2123 {
2124   \bool_if:NT \l_@@_code_before_bool
2125   {
2126     \hbox

```

```

2127         {
2128             \skip_horizontal:N -0.5\arrayrulewidth
2129             \pgfsys@markposition { \@@_env: - col - 2 }
2130             \skip_horizontal:N 0.5\arrayrulewidth
2131         }
2132     }
2133     \pgfpicture
2134     \pgfrememberpicturepositiononpagetrue
2135     \pgfcoordinate { \@@_env: - col - 2 }
2136     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2137     \str_if_empty:NF \l_@@_name_str
2138     { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
2139     \endpgfpicture
2140 }

```

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.

```

2141 \int_gset:Nn \g_tmpa_int 1
2142 \bool_if:NTF \g_@@_last_col_found_bool
2143 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
2144 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
2145 {
2146     &
2147     \omit
2148     \int_gincr:N \g_tmpa_int

```

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

```

2149     \skip_horizontal:N \g_tmpa_skip
2150     \bool_if:NT \l_@@_code_before_bool
2151     {
2152         \hbox
2153         {
2154             \skip_horizontal:N -0.5\arrayrulewidth
2155             \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2156             \skip_horizontal:N 0.5\arrayrulewidth
2157         }
2158     }

```

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

```

2159     \pgfpicture
2160     \pgfrememberpicturepositiononpagetrue
2161     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2162     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2163     \str_if_empty:NF \l_@@_name_str
2164     {
2165         \pgfnodealias
2166         { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
2167         { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2168     }
2169     \endpgfpicture
2170 }
2171 \bool_if:NT \g_@@_last_col_found_bool
2172 {
2173     \hbox_overlap_right:n
2174     {
2175         % \skip_horizontal:N \col@sep
2176         \skip_horizontal:N \g_@@_width_last_col_dim
2177         \bool_if:NT \l_@@_code_before_bool
2178         {
2179             \pgfsys@markposition
2180             { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2181         }
2182         \pgfpicture
2183         \pgfrememberpicturepositiononpagetrue
2184         \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }

```

```

2185         \pgfpintorigin
2186         \str_if_empty:NF \l_@@_name_str
2187         {
2188             \pgfnodealias
2189             { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
2190             { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2191         }
2192         \endpgfpicture
2193     }
2194 }
2195 \cr
2196 }

```

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

```

2197 \tl_const:Nn \c_@@_preamble_first_col_tl
2198 {
2199     >
2200     {

```

At the beginning of the cell, we link `\CodeAfter` to a command which do *not* begin with `\omit` (whereas the standard version of `\CodeAfter` begins with `\omit`).

```

2201         \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n
2202         \bool_gset_true:N \g_@@_after_col_zero_bool
2203         \@@_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.

```

2204         \hbox_set:Nw \l_@@_cell_box
2205         \@@_math_toggle_token:
2206         \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”.

```

2207         \bool_lazy_and:nnT
2208         { \int_compare_p:nNn \c@iRow > 0 }
2209         {
2210             \bool_lazy_or_p:nn
2211             { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2212             { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2213         }
2214         {
2215             \l_@@_code_for_first_col_tl
2216             \xglobal \colorlet { nicematrix-first-col } { . }
2217         }
2218     }

```

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`.

```

2219     l
2220     <
2221     {
2222         \@@_math_toggle_token:
2223         \hbox_set_end:
2224         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2225         \@@_adjust_size_box:
2226         \@@_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.

```

2227         \dim_gset:Nn \g_@@_width_first_col_dim
2228         { \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.

```

2229 \hbox_overlap_left:n
2230 {
2231   \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2232     \@@_node_for_the_cell:
2233     { \box_use_drop:N \l_@@_cell_box }
2234     \skip_horizontal:N \l_@@_left_delim_dim
2235     \skip_horizontal:N \l_@@_left_margin_dim
2236     \skip_horizontal:N \l_@@_extra_left_margin_dim
2237   }
2238   \bool_gset_false:N \g_@@_empty_cell_bool
2239   \skip_horizontal:N -2\col@sep
2240 }
2241 }

```

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

```

2242 \tl_const:Nn \c_@@_preamble_last_col_tl
2243 {
2244   >
2245   {

```

At the beginning of the cell, we link `\CodeAfter` to a command which do *not* begin with `\omit` (whereas the standard version of `\CodeAfter` begins with `\omit`).

```

2246   \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n

```

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

```

2247   \bool_gset_true:N \g_@@_last_col_found_bool
2248   \int_gincr:N \c@jCol
2249   \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.

```

2250   \hbox_set:Nw \l_@@_cell_box
2251   \@@_math_toggle_token:
2252   \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”.

```

2253   \int_compare:nNnT \c@iRow > 0
2254   {
2255     \bool_lazy_or:nnT
2256     { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2257     { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2258     {
2259       \l_@@_code_for_last_col_tl
2260       \xglobal \colorlet { nicematrix-last-col } { . }
2261     }
2262   }
2263 }
2264 1
2265 <
2266 {
2267   \@@_math_toggle_token:
2268   \hbox_set_end:
2269   \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2270   \@@_adjust_size_box:
2271   \@@_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.

```

2272   \dim_gset:Nn \g_@@_width_last_col_dim
2273   { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
2274   \skip_horizontal:N -2\col@sep

```

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

```

2275     \hbox_overlap_right:n
2276     {
2277         \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2278         {
2279             \skip_horizontal:N \l_@@_right_delim_dim
2280             \skip_horizontal:N \l_@@_right_margin_dim
2281             \skip_horizontal:N \l_@@_extra_right_margin_dim
2282             \@@_node_for_the_cell:
2283         }
2284     }
2285     \bool_gset_false:N \g_@@_empty_cell_bool
2286 }
2287 }

```

The environment `{NiceArray}` is constructed upon the environment `{NiceArrayWithDelims}` but, in fact, there is a flag `\g_@@_NiceArray_bool`. In `{NiceArrayWithDelims}`, some special code will be executed if this flag is raised.

```

2288 \NewDocumentEnvironment { NiceArray } { }
2289 {
2290     \bool_gset_true:N \g_@@_NiceArray_bool
2291     \str_if_empty:NT \g_@@_name_env_str
2292     { \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 `\g_@@_NiceArray_bool` is raised).

```

2293     \NiceArrayWithDelims . .
2294 }
2295 { \endNiceArrayWithDelims }

```

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

```

2296 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
2297 {
2298     \NewDocumentEnvironment { #1 NiceArray } { }
2299     {
2300         \str_if_empty:NT \g_@@_name_env_str
2301         { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
2302         \@@_test_if_math_mode:
2303         \NiceArrayWithDelims #2 #3
2304     }
2305     { \endNiceArrayWithDelims }
2306 }
2307 \@@_def_env:nnn p ( )
2308 \@@_def_env:nnn b [ ]
2309 \@@_def_env:nnn B \{ \}
2310 \@@_def_env:nnn v | |
2311 \@@_def_env:nnn V \| \|

```

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

```

2312 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
2313 {
2314     \bool_set_true:N \l_@@_Matrix_bool
2315     \use:c { #1 NiceArray }
2316     {
2317         *
2318         {
2319             \int_compare:nNnTF \l_@@_last_col_int < 0

```

```

2320         \c@MaxMatrixCols
2321         { \@@_pred:n \l_@@_last_col_int }
2322     }
2323     { > \@@_Cell: #2 < \@@_end_Cell: }
2324 }
2325 }
2326 \clist_map_inline:nn { { } , p , b , B , v , V }
2327 {
2328     \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
2329     {
2330         \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
2331         \tl_set:Nn \l_@@_type_of_col_tl c
2332         \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
2333         \exp_args:Nne \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
2334     }
2335     { \use:c { end #1 NiceArray } }
2336 }

```

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

```

2337 \cs_new_protected:Npn \@@_NotEmpty:
2338 { \bool_gset_true:N \g_@@_not_empty_cell_bool }

```

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

```

2339 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
2340 {
2341     \str_gset:Nn \g_@@_name_env_str { NiceTabular }
2342     \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
2343     \bool_set_true:N \l_@@_NiceTabular_bool
2344     \NiceArray { #2 }
2345 }
2346 { \endNiceArray }

2347 \NewDocumentEnvironment { NiceTabular* } { m 0 { } m ! 0 { } }
2348 {
2349     \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
2350     \dim_set:Nn \l_@@_tabular_width_dim { #1 }
2351     \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2352     \bool_set_true:N \l_@@_NiceTabular_bool
2353     \NiceArray { #3 }
2354 }
2355 { \endNiceArray }

```

## After the construction of the array

```

2356 \cs_new_protected:Npn \@@_after_array:
2357 {
2358     \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.

```

2359     \bool_if:NT \g_@@_last_col_found_bool
2360     { \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`.

```

2361     \bool_if:NT \l_@@_last_col_without_value_bool

```

```

2362 {
2363   \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
2364   \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2365   \iow_shipout:Nx \@mainaux
2366   {
2367     \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
2368     { \int_use:N \g_@@_col_total_int }
2369   }
2370   \str_if_empty:NF \l_@@_name_str
2371   {
2372     \iow_shipout:Nx \@mainaux
2373     {
2374       \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
2375       { \int_use:N \g_@@_col_total_int }
2376     }
2377   }
2378   \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2379 }

```

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.

```

2380   \bool_if:NT \l_@@_last_row_without_value_bool
2381   {
2382     \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.

```

2383   \bool_if:NF \l_@@_light_syntax_bool
2384   {
2385     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2386     \iow_shipout:Nx \@mainaux
2387     {
2388       \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
2389       { \int_use:N \g_@@_row_total_int }
2390     }

```

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.

```

2391     \str_if_empty:NF \l_@@_name_str
2392     {
2393       \iow_shipout:Nx \@mainaux
2394       {
2395         \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
2396         { \int_use:N \g_@@_row_total_int }
2397       }
2398     }
2399     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2400   }
2401 }

```

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

```

2402   \bool_if:NT \l_@@_code_before_bool
2403   {
2404     \iow_now:Nn \@mainaux \ExplSyntaxOn
2405     \iow_now:Nx \@mainaux
2406     { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
2407     \iow_now:Nx \@mainaux
2408     {
2409       \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
2410       {
2411         \int_use:N \l_@@_first_row_int ,
2412         \int_use:N \g_@@_row_total_int ,
2413         \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`.

```

2414         \bool_lazy_and:nnTF
2415         { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
2416         { \bool_not_p:n \g_@@_last_col_found_bool }
2417         \@@_succ:n
2418         \int_use:N
2419         \g_@@_col_total_int
2420     }

```

We write also the potential content of `\g_@@_pos_of_blocks_seq` (it will be useful if the command `\rowcolors` is used with the key `respect-blocks`).

```

2421         \seq_gset_from_clist:cn
2422         { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
2423         { \seq_use:Nnnn \g_@@_pos_of_blocks_seq , , , }
2424     }
2425 }

```

Now, you create the diagonal nodes by using the `row` nodes and the `col` nodes. If the engine is `xetex` or `luatex` we also create the “½ nodes”.

```

2426 \@@_create_diag_nodes:

```

By default, the diagonal lines will be parallelized<sup>58</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.

```

2427     \bool_if:NT \l_@@_parallelize_diags_bool
2428     {
2429         \int_gzero_new:N \g_@@_ddots_int
2430         \int_gzero_new:N \g_@@_iddots_int

```

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

```

2431         \dim_gzero_new:N \g_@@_delta_x_one_dim
2432         \dim_gzero_new:N \g_@@_delta_y_one_dim
2433         \dim_gzero_new:N \g_@@_delta_x_two_dim
2434         \dim_gzero_new:N \g_@@_delta_y_two_dim
2435     }
2436     \int_zero_new:N \l_@@_initial_i_int
2437     \int_zero_new:N \l_@@_initial_j_int
2438     \int_zero_new:N \l_@@_final_i_int
2439     \int_zero_new:N \l_@@_final_j_int
2440     \bool_set_false:N \l_@@_initial_open_bool
2441     \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.

```

2442     \bool_if:NT \l_@@_small_bool
2443     {
2444         \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2445         \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.

```

2446         \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2447     }

```

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

```

2448 \@@_draw_dotted_lines:

```

---

<sup>58</sup>It’s possible to use the option `parallelize-diags` to disable this parallelization.

The following computes the “corners” (made up of empty cells) but if there is no corner to compute, it won’t do anything. The corners are computed in `\l_@@_corners_cells_seq` which will contain all the cells which are empty (and not in a block) considered in the corners of the array.

```
2449 \@@_compute_corners:
```

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

```
2450 \@@_adjust_pos_of_blocks_seq:
```

The following code is only for efficiency. We determine whether the potential horizontal and vertical rules are “complete”, that is to say drawn in the whole array. We are sure that all the rules will be complete when there is no block, no virtual block (determined by a command such as `\Cdots`, `\Vdots`, etc.) and no corners. In that case, we switch to a shortcut version of `\@@_vline_i:nn` and `\@@_hline:nn`.

```
2451 \bool_lazy_all:nT
2452 {
2453   { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
2454   { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
2455   { \seq_if_empty_p:N \l_@@_corners_cells_seq }
2456 }
2457 {
2458   \cs_set_eq:NN \@@_vline_i:nn \@@_vline_i_complete:nn
2459   \cs_set_eq:NN \@@_hline_i:nn \@@_hline_i_complete:nn
2460 }
2461 \tl_if_empty:NF \l_@@_hlines_clist \@@_draw_hlines:
2462 \tl_if_empty:NF \l_@@_vlines_clist \@@_draw_vlines:
2463 \cs_set_eq:NN \SubMatrix \@@_SubMatrix
```

Now, the internal code-after and then, the `\CodeAfter`.

```
2464 \bool_if:NT \c_@@_tikz_loaded_bool
2465 {
2466   \tikzset
2467   {
2468     every~picture / .style =
2469     {
2470       overlay ,
2471       remember~picture ,
2472       name-prefix = \@@_env: -
2473     }
2474   }
2475 }
2476 \cs_set_eq:NN \line \@@_line
2477 \g_@@_internal_code_after_tl
2478 \tl_gclear:N \g_@@_internal_code_after_tl
```

When `light-syntax` is used, we insert systematically a `\CodeAfter` in the flow. Thus, it’s possible to have two instructions `\CodeAfter` and the second may be in `\g_nicematrix_code_after_tl`. That’s why we set `\Code-after` to be *no-op* now.

```
2479 \cs_set_eq:NN \CodeAfter \prg_do_nothing:
```

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

```
2480 \seq_gclear:N \g_@@_submatrix_names_seq
```

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

```
2481 \exp_last_unbraced:NV \@@_CodeAfter_keys: \g_nicematrix_code_after_tl
2482 \scan_stop:
2483 \tl_gclear:N \g_nicematrix_code_after_tl
2484 \group_end:
```

`\g_nicematrix_code_before_tl` is for instructions in the cells of the array such as `\rowcolor` and `\cellcolor` (when the key `colortbl-like` is in force). These instructions will be written on the `aux` file to be added to the `code-before` in the next run.

```
2485 \tl_if_empty:NF \g_nicematrix_code_before_tl
2486 {
```

The command `\rowcolor` in `tabular` will in fact use `\rectanglecolor` in order to follow the behaviour of `\rowcolor` of `colortbl`. That's why there may be a command `\rectanglecolor` in `\g_nicematrix_code_before_tl`. In order to avoid an error during the expansion, we define a protected version of `\rectanglecolor`.

```
2487 \cs_set_protected:Npn \rectanglecolor { }
2488 \cs_set_protected:Npn \columncolor { }
2489 \iow_now:Nn \@mainaux \ExplSyntaxOn
2490 \iow_now:Nx \@mainaux
2491 {
2492 \tl_gset:cn
2493 { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
2494 { \exp_not:V \g_nicematrix_code_before_tl }
2495 }
2496 \iow_now:Nn \@mainaux \ExplSyntaxOff
2497 \bool_set_true:N \l_@@_code_before_bool
2498 }

2499 \bool_gset_false:N \g_@@_NiceArray_bool
2500 \str_gclear:N \g_@@_name_env_str
2501 \@@_restore_iRow_jCol:
```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>59</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`.

```
2502 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
2503 }
```

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

```
2504 \NewDocumentCommand \@@_CodeAfter_keys: { 0 { } }
2505 { \keys_set:nn { NiceMatrix / CodeAfter } { #1 } }
```

We remind that the first mandatory argument of the command `\Block` is the size of the block with the special format  $i-j$ . However, the user is allowed to omit  $i$  or  $j$  (or both). This will be interpreted as: the last row (resp. column) of the block will be the last row (resp. column) of the block (without the potential exterior row—resp. column—of the array). By convention, this is stored in `\g_@@_pos_of_blocks_seq` (and `\g_@@_blocks_seq`) as a number of rows (resp. columns) for the block equal to 100. It's possible, after the construction of the array, to replace these values by the correct ones (since we know the number of rows and columns of the array).

```
2506 \cs_new_protected:Npn \@@_adjust_pos_of_blocks_seq:
2507 {
2508 \seq_gset_map_x:NNn \g_@@_pos_of_blocks_seq \g_@@_pos_of_blocks_seq
2509 { \@@_adjust_pos_of_blocks_seq_i:nnnn ##1 }
2510 }
```

The following command must *not* be protected.

```
2511 \cs_new:Npn \@@_adjust_pos_of_blocks_seq_i:nnnn #1 #2 #3 #4
2512 {
2513 { #1 }
2514 { #2 }
2515 {
2516 \int_compare:nNnTF { #3 } > { 99 }
```

---

<sup>59</sup>e.g. `\color[rgb]{0.5,0.5,0}`

```

2517     { \int_use:N \c@iRow }
2518     { #3 }
2519   }
2520   {
2521     \int_compare:nNnTF { #4 } > { 99 }
2522     { \int_use:N \c@jCol }
2523     { #4 }
2524   }
2525 }

```

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).

```

2526 \AtBeginDocument
2527 {
2528   \cs_new_protected:Npx \@@_draw_dotted_lines:
2529   {
2530     \c_@@_pgfortikzpicture_tl
2531     \@@_draw_dotted_lines_i:
2532     \c_@@_endpgfortikzpicture_tl
2533   }
2534 }

```

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

```

2535 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2536 {
2537   \pgfrememberpicturepositiononpagetrue
2538   \pgf@relevantforpicturesizefalse
2539   \g_@@_HVdotsfor_lines_tl
2540   \g_@@_Vdots_lines_tl
2541   \g_@@_Ddots_lines_tl
2542   \g_@@_Iddots_lines_tl
2543   \g_@@_Cdots_lines_tl
2544   \g_@@_Ldots_lines_tl
2545 }

2546 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2547 {
2548   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2549   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2550 }

```

We define a new PGF shape for the diag nodes because we want to provide a anchor called `.5` for those nodes.

```

2551 \pgfdeclareshape { @@_diag_node }
2552 {
2553   \savedanchor { \five }
2554   {
2555     \dim_gset_eq:NN \pgf@x \l_tmpa_dim
2556     \dim_gset_eq:NN \pgf@y \l_tmpb_dim
2557   }
2558   \anchor { 5 } { \five }
2559   \anchor { center } { \pgfpictureorigin }
2560 }

```

The following command creates the diagonal nodes (in fact, if the matrix is not a square matrix, not all the nodes are on the diagonal).

```

2561 \cs_new_protected:Npn \@@_create_diag_nodes:
2562 {
2563   \pgfpicture
2564   \pgfrememberpicturepositiononpagetrue

```



```

2565 \int_step_inline:nn { \int_max:nn \c@iRow \c@jCol }
2566 {
2567   \@@_qpoint:n { col - \int_min:nn { ##1 } { \c@jCol + 1 } }
2568   \dim_set_eq:NN \l_tmpa_dim \pgf@x
2569   \@@_qpoint:n { row - \int_min:nn { ##1 } { \c@iRow + 1 } }
2570   \dim_set_eq:NN \l_tmpb_dim \pgf@y
2571   \@@_qpoint:n { col - \int_min:nn { ##1 + 1 } { \c@jCol + 1 } }
2572   \dim_set_eq:NN \l_tmpc_dim \pgf@x
2573   \@@_qpoint:n { row - \int_min:nn { ##1 + 1 } { \c@iRow + 1 } }
2574   \dim_set_eq:NN \l_tmpd_dim \pgf@y
2575   \pgftransformshift { \pgfpoint \l_tmpa_dim \l_tmpb_dim }

```

Now, `\l_tmpa_dim` and `\l_tmpb_dim` become the width and the height of the node (of shape `@_diag_node`) that we will construct.

```

2576   \dim_set:Nn \l_tmpa_dim { ( \l_tmpc_dim - \l_tmpa_dim ) / 2 }
2577   \dim_set:Nn \l_tmpb_dim { ( \l_tmpd_dim - \l_tmpb_dim ) / 2 }
2578   \pgfnode { @_diag_node } { center } { } { \@@_env: - ##1 } { }
2579 }

```

Now, the last node. Of course, that is only a coordinate because there is not `.5` anchor for that node.

```

2580 \int_set:Nn \l_tmpa_int { \int_max:nn \c@iRow \c@jCol + 1 }
2581 \@@_qpoint:n { row - \int_min:nn { \l_tmpa_int } { \c@iRow + 1 } }
2582 \dim_set_eq:NN \l_tmpa_dim \pgf@y
2583 \@@_qpoint:n { col - \int_min:nn { \l_tmpa_int } { \c@jCol + 1 } }
2584 \pgfcoordinate
2585 { \@@_env: - \int_use:N \l_tmpa_int } { \pgfpoint \pgf@x \l_tmpa_dim }
2586 \pgfnodelalias
2587 { \@@_env: - last }
2588 { \@@_env: - \int_eval:n { \int_max:nn \c@iRow \c@jCol + 1 } }
2589 \endpgfpicture
2590 }

```

## 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.

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

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

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

This command computes:

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

```

2591 \cs_new_protected:Npn \@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2592 {

```

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

```
2593 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
2594 \int_set:Nn \l_@@_initial_i_int { #1 }
2595 \int_set:Nn \l_@@_initial_j_int { #2 }
2596 \int_set:Nn \l_@@_final_i_int { #1 }
2597 \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.

```
2598 \bool_set_false:N \l_@@_stop_loop_bool
2599 \bool_do_until:Nn \l_@@_stop_loop_bool
2600 {
2601   \int_add:Nn \l_@@_final_i_int { #3 }
2602   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
2603   \bool_set_false:N \l_@@_final_open_bool
2604   \int_compare:nNnTF \l_@@_final_i_int > \l_@@_row_max_int
2605   {
2606     \int_compare:nNnTF { #3 } = 1
2607     { \bool_set_true:N \l_@@_final_open_bool }
2608     {
2609       \int_compare:nNnT \l_@@_final_j_int > \l_@@_col_max_int
2610       { \bool_set_true:N \l_@@_final_open_bool }
2611     }
2612   }
2613   {
2614     \int_compare:nNnTF \l_@@_final_j_int < \l_@@_col_min_int
2615     {
2616       \int_compare:nNnT { #4 } = { -1 }
2617       { \bool_set_true:N \l_@@_final_open_bool }
2618     }
2619     {
2620       \int_compare:nNnT \l_@@_final_j_int > \l_@@_col_max_int
2621       {
2622         \int_compare:nNnT { #4 } = 1
2623         { \bool_set_true:N \l_@@_final_open_bool }
2624       }
2625     }
2626   }
2627   \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.

```
2628   {
```

We do a step backwards.

```
2629   \int_sub:Nn \l_@@_final_i_int { #3 }
2630   \int_sub:Nn \l_@@_final_j_int { #4 }
2631   \bool_set_true:N \l_@@_stop_loop_bool
2632   }
```

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`.

```
2633   {
2634     \cs_if_exist:cTF
2635     {
2636       @@ _ dotted _
2637       \int_use:N \l_@@_final_i_int -
2638       \int_use:N \l_@@_final_j_int
2639     }
2640     {
2641       \int_sub:Nn \l_@@_final_i_int { #3 }
```

```

2642     \int_sub:Nn \l_@@_final_j_int { #4 }
2643     \bool_set_true:N \l_@@_final_open_bool
2644     \bool_set_true:N \l_@@_stop_loop_bool
2645   }
2646   {
2647     \cs_if_exist:cTF
2648     {
2649       pgf @ sh @ ns @ \@@_env:
2650       - \int_use:N \l_@@_final_i_int
2651       - \int_use:N \l_@@_final_j_int
2652     }
2653     { \bool_set_true:N \l_@@_stop_loop_bool }

```

If the case is empty, we declare that the cell as non-empty. Indeed, we will draw a dotted line and the cell will be on that dotted line. All the cells of a dotted line have to be marked as “dotted” because we don’t want intersections between dotted lines. We recall that the research of the extremities of the lines are all done in the same TeX group (the group of the environment), 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.

```

2654     {
2655       \cs_set:cpn
2656       {
2657         @@ _ dotted _
2658         \int_use:N \l_@@_final_i_int -
2659         \int_use:N \l_@@_final_j_int
2660       }
2661     { }
2662   }
2663 }
2664 }
2665 }

```

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

```

2666     \bool_set_false:N \l_@@_stop_loop_bool
2667     \bool_do_until:Nn \l_@@_stop_loop_bool
2668     {
2669       \int_sub:Nn \l_@@_initial_i_int { #3 }
2670       \int_sub:Nn \l_@@_initial_j_int { #4 }
2671       \bool_set_false:N \l_@@_initial_open_bool
2672       \int_compare:nNnTF \l_@@_initial_i_int < \l_@@_row_min_int
2673       {
2674         \int_compare:nNnTF { #3 } = 1
2675         { \bool_set_true:N \l_@@_initial_open_bool }
2676         {
2677           \int_compare:nNnT \l_@@_initial_j_int = { \l_@@_col_min_int -1 }
2678           { \bool_set_true:N \l_@@_initial_open_bool }
2679         }
2680       }
2681     {
2682       \int_compare:nNnTF \l_@@_initial_j_int < \l_@@_col_min_int
2683       {
2684         \int_compare:nNnT { #4 } = 1
2685         { \bool_set_true:N \l_@@_initial_open_bool }
2686       }
2687       {
2688         \int_compare:nNnT \l_@@_initial_j_int > \l_@@_col_max_int
2689         {
2690           \int_compare:nNnT { #4 } = { -1 }
2691           { \bool_set_true:N \l_@@_initial_open_bool }
2692         }
2693       }
2694     }

```

```

2695 \bool_if:NTF \l_@@_initial_open_bool
2696 {
2697   \int_add:Nn \l_@@_initial_i_int { #3 }
2698   \int_add:Nn \l_@@_initial_j_int { #4 }
2699   \bool_set_true:N \l_@@_stop_loop_bool
2700 }
2701 {
2702   \cs_if_exist:cTF
2703   {
2704     @@ _ dotted _
2705     \int_use:N \l_@@_initial_i_int -
2706     \int_use:N \l_@@_initial_j_int
2707   }
2708   {
2709     \int_add:Nn \l_@@_initial_i_int { #3 }
2710     \int_add:Nn \l_@@_initial_j_int { #4 }
2711     \bool_set_true:N \l_@@_initial_open_bool
2712     \bool_set_true:N \l_@@_stop_loop_bool
2713   }
2714   {
2715     \cs_if_exist:cTF
2716     {
2717       pgf @ sh @ ns @ \@@_env:
2718       - \int_use:N \l_@@_initial_i_int
2719       - \int_use:N \l_@@_initial_j_int
2720     }
2721     { \bool_set_true:N \l_@@_stop_loop_bool }
2722     {
2723       \cs_set:cpn
2724       {
2725         @@ _ dotted _
2726         \int_use:N \l_@@_initial_i_int -
2727         \int_use:N \l_@@_initial_j_int
2728       }
2729       { }
2730     }
2731   }
2732 }
2733 }

```

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.

```

2734 \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2735 {
2736   { \int_use:N \l_@@_initial_i_int }
2737   { \int_use:N \l_@@_initial_j_int }
2738   { \int_use:N \l_@@_final_i_int }
2739   { \int_use:N \l_@@_final_j_int }
2740 }
2741 }

```

The following command (*when it will be written*) will set the four counters `\l_@@_row_min_int`, `\l_@@_row_max_int`, `\l_@@_col_min_int` and `\l_@@_col_max_int` to the intersections of the submatrices which contains the cell of row #1 and column #2. As of now, it’s only the whole array (excepted exterior row and columns).

```

2742 \cs_new_protected:Npn \@@_adjust_to_submatrix:nn #1 #2
2743 {
2744   \int_set:Nn \l_@@_row_min_int 1
2745   \int_set:Nn \l_@@_col_min_int 1
2746   \int_set_eq:NN \l_@@_row_max_int \c@iRow
2747   \int_set_eq:NN \l_@@_col_max_int \c@jCol

```

We do a loop over all the submatrices specified in the code-before. We have stored the position of all those submatrices in `\g_@@_submatrix_seq`.

```

2748 \seq_map_inline:Nn \g_@@_submatrix_seq
2749 { \@@_adjust_to_submatrix:nnnnnn { #1 } { #2 } ##1 }
2750 }

```

#1 and #2 are the numbers of row and columns of the cell where the command of dotted line (ex.: \Vdots) has been issued. #3, #4, #5 and #6 are the specification (in  $i$  and  $j$ ) of the submatrix where are analysing.

```

2751 \cs_set_protected:Npn \@@_adjust_to_submatrix:nnnnnn #1 #2 #3 #4 #5 #6
2752 {
2753   \bool_if:nT
2754   {
2755     \int_compare_p:n { #3 <= #1 }
2756     && \int_compare_p:n { #1 <= #5 }
2757     && \int_compare_p:n { #4 <= #2 }
2758     && \int_compare_p:n { #2 <= #6 }
2759   }
2760   {
2761     \int_set:Nn \l_@@_row_min_int { \int_max:nn \l_@@_row_min_int { #3 } }
2762     \int_set:Nn \l_@@_col_min_int { \int_max:nn \l_@@_col_min_int { #4 } }
2763     \int_set:Nn \l_@@_row_max_int { \int_min:nn \l_@@_row_max_int { #5 } }
2764     \int_set:Nn \l_@@_col_max_int { \int_min:nn \l_@@_col_max_int { #6 } }
2765   }
2766 }

2767 \cs_new_protected:Npn \@@_set_initial_coords:
2768 {
2769   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2770   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2771 }
2772 \cs_new_protected:Npn \@@_set_final_coords:
2773 {
2774   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2775   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2776 }
2777 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2778 {
2779   \pgfpointanchor
2780   {
2781     \@@_env:
2782     - \int_use:N \l_@@_initial_i_int
2783     - \int_use:N \l_@@_initial_j_int
2784   }
2785   { #1 }
2786   \@@_set_initial_coords:
2787 }
2788 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2789 {
2790   \pgfpointanchor
2791   {
2792     \@@_env:
2793     - \int_use:N \l_@@_final_i_int
2794     - \int_use:N \l_@@_final_j_int
2795   }
2796   { #1 }
2797   \@@_set_final_coords:
2798 }

2799 \cs_new_protected:Npn \@@_open_x_initial_dim:
2800 {
2801   \dim_set_eq:NN \l_@@_x_initial_dim \c_max_dim
2802   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
2803   {
2804     \cs_if_exist:cT
2805     { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_initial_j_int }

```

```

2806     {
2807         \pgfpointanchor
2808         { \l_@@_env: - ##1 - \int_use:N \l_@@_initial_j_int }
2809         { west }
2810         \dim_set:Nn \l_@@_x_initial_dim
2811         { \dim_min:nn \l_@@_x_initial_dim \pgf@x }
2812     }
2813 }

```

If, in fact, all the cells of the columns are empty (no PGF/Tikz nodes in those cells).

```

2814     \dim_compare:nNnT \l_@@_x_initial_dim = \c_max_dim
2815     {
2816         \l_@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2817         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2818         \dim_add:Nn \l_@@_x_initial_dim \col@sep
2819     }
2820 }
2821 \cs_new_protected:Npn \l_@@_open_x_final_dim:
2822 {
2823     \dim_set:Nn \l_@@_x_final_dim { - \c_max_dim }
2824     \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
2825     {
2826         \cs_if_exist:cT
2827         { pgf @ sh @ ns @ \l_@@_env: - ##1 - \int_use:N \l_@@_final_j_int }
2828         {
2829             \pgfpointanchor
2830             { \l_@@_env: - ##1 - \int_use:N \l_@@_final_j_int }
2831             { east }
2832             \dim_set:Nn \l_@@_x_final_dim
2833             { \dim_max:nn \l_@@_x_final_dim \pgf@x }
2834         }
2835     }

```

If, in fact, all the cells of the columns are empty (no PGF/Tikz nodes in those cells).

```

2836     \dim_compare:nNnT \l_@@_x_final_dim = { - \c_max_dim }
2837     {
2838         \l_@@_qpoint:n { col - \l_@@_succ:n \l_@@_final_j_int }
2839         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2840         \dim_sub:Nn \l_@@_x_final_dim \col@sep
2841     }
2842 }

```

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.

```

2843 \cs_new_protected:Npn \l_@@_draw_Ldots:nnn #1 #2 #3
2844 {
2845     \l_@@_adjust_to_submatrix:nn { #1 } { #2 }
2846     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2847     {
2848         \l_@@_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.

```

2849     \group_begin:
2850     \int_compare:nNnTF { #1 } = 0
2851     { \color { nicematrix-first-row } }
2852     {

```

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.

```

2853         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2854         { \color { nicematrix-last-row } }
2855     }

```

```

2856         \keys_set:nn { NiceMatrix / xdots } { #3 }
2857         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2858         \@@_actually_draw_Ldots:
2859     \group_end:
2860 }
2861 }

```

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`.

```

2862 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2863 {
2864     \bool_if:NTF \l_@@_initial_open_bool
2865     {
2866         \@@_open_x_initial_dim:
2867         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2868         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2869     }
2870     { \@@_set_initial_coords_from_anchor:n { base-east } }
2871     \bool_if:NTF \l_@@_final_open_bool
2872     {
2873         \@@_open_x_final_dim:
2874         \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2875         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2876     }
2877     { \@@_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 (?).

```

2878     \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2879     \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2880     \@@_draw_line:
2881 }

```

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.

```

2882 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2883 {
2884     \@@_adjust_to_submatrix:nn { #1 } { #2 }
2885     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2886     {
2887         \@@_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.

```

2888     \group_begin:
2889     \int_compare:nNnTF { #1 } = 0
2890     { \color { nicematrix-first-row } }
2891     {

```

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.

```

2892         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2893         { \color { nicematrix-last-row } }
2894     }
2895     \keys_set:nn { NiceMatrix / xdots } { #3 }
2896     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2897     \@@_actually_draw_Cdots:
2898 \group_end:
2899 }
2900 }

```

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.`

```

2901 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2902 {
2903     \bool_if:NTF \l_@@_initial_open_bool
2904     { \@@_open_x_initial_dim: }
2905     { \@@_set_initial_coords_from_anchor:n { mid-east } }
2906     \bool_if:NTF \l_@@_final_open_bool
2907     { \@@_open_x_final_dim: }
2908     { \@@_set_final_coords_from_anchor:n { mid-west } }
2909     \bool_lazy_and:nnTF
2910     \l_@@_initial_open_bool
2911     \l_@@_final_open_bool
2912     {
2913         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2914         \dim_set_eq:NN \l_tmpa_dim \pgf@y
2915         \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2916         \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2917         \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2918     }
2919     {
2920         \bool_if:NT \l_@@_initial_open_bool
2921         { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2922         \bool_if:NT \l_@@_final_open_bool
2923         { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2924     }
2925     \@@_draw_line:
2926 }
2927 \cs_new_protected:Npn \@@_open_y_initial_dim:
2928 {
2929     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2930     \dim_set:Nn \l_@@_y_initial_dim
2931     { \pgf@y + ( \box_ht:N \strutbox + \extrarowheight ) * \arraystretch }
2932     \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
2933     {
2934         \cs_if_exist:cT
2935         { \pgf @ sh @ ns @ \@@_env: - \int_use:N \l_@@_initial_i_int - ##1 }
2936         {
2937             \pgfpointanchor
2938             { \@@_env: - \int_use:N \l_@@_initial_i_int - ##1 }

```



```

2939         { north }
2940         \dim_set:Nn \l_@@_y_initial_dim
2941         { \dim_max:nn \l_@@_y_initial_dim \pgf@y }
2942     }
2943 }
2944 }
2945 \cs_new_protected:Npn \@@_open_y_final_dim:
2946 {
2947     \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2948     \dim_set:Nn \l_@@_y_final_dim
2949     { \pgf@y - ( \box_dp:N \strutbox ) * \arraystretch }
2950     \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
2951     {
2952         \cs_if_exist:cT
2953         { \pgf @ sh @ ns @ \@@_env: - \int_use:N \l_@@_final_i_int - ##1 }
2954         {
2955             \pgfpointanchor
2956             { \@@_env: - \int_use:N \l_@@_final_i_int - ##1 }
2957             { south }
2958             \dim_set:Nn \l_@@_y_final_dim
2959             { \dim_min:nn \l_@@_y_final_dim \pgf@y }
2960         }
2961     }
2962 }

```

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.

```

2963 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2964 {
2965     \@@_adjust_to_submatrix:nn { #1 } { #2 }
2966     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2967     {
2968         \@@_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.

```

2969     \group_begin:
2970     \int_compare:nNnTF { #2 } = 0
2971     { \color { nicematrix-first-col } }
2972     {
2973         \int_compare:nNnT { #2 } = \l_@@_last_col_int
2974         { \color { nicematrix-last-col } }
2975     }
2976     \keys_set:nn { NiceMatrix / xdots } { #3 }
2977     \tl_if_empty:VF \l_@@_xdots_color_tl
2978     { \color { \l_@@_xdots_color_tl } }
2979     \@@_actually_draw_Vdots:
2980     \group_end:
2981 }
2982 }

```

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`.

```
2983 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2984 {
```

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

```
2985 \bool_set_false:N \l_tmpa_bool
```

First the case when the line is closed on both ends.

```
2986 \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2987 {
2988   \@@_set_initial_coords_from_anchor:n { south-west }
2989   \@@_set_final_coords_from_anchor:n { north-west }
2990   \bool_set:Nn \l_tmpa_bool
2991   { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2992 }
```

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

```
2993 \bool_if:NTF \l_@@_initial_open_bool
2994 \@@_open_y_initial_dim:
2995 { \@@_set_initial_coords_from_anchor:n { south } }
2996 \bool_if:NTF \l_@@_final_open_bool
2997 \@@_open_y_final_dim:
2998 { \@@_set_final_coords_from_anchor:n { north } }
2999 \bool_if:NTF \l_@@_initial_open_bool
3000 {
3001   \bool_if:NTF \l_@@_final_open_bool
3002   {
3003     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
3004     \dim_set_eq:NN \l_tmpa_dim \pgf@x
3005     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
3006     \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
3007     \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.

```
3008 \int_compare:nNnT \l_@@_last_col_int > { -2 }
3009 {
3010   \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
3011   {
3012     \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
3013     \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
3014     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
3015     \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
3016   }
3017 }
3018 }
3019 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
3020 }
3021 {
3022   \bool_if:NTF \l_@@_final_open_bool
3023   { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
3024 }
```

Now the case where both extremities are closed. The first conditional tests whether the column is of type `c` or may be considered as if.

```
3025 \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
3026 {
3027   \dim_set:Nn \l_@@_x_initial_dim
3028   {
3029     \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
3030     \l_@@_x_initial_dim \l_@@_x_final_dim
3031   }
3032   \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
3033 }
```

```

3034     }
3035   }
3036   \@@_draw_line:
3037 }

```

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.

```

3038 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
3039 {
3040   \@@_adjust_to_submatrix:nn { #1 } { #2 }
3041   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
3042   {
3043     \@@_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.

```

3044     \group_begin:
3045     \keys_set:nn { NiceMatrix / xdots } { #3 }
3046     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3047     \@@_actually_draw_Ddots:
3048   \group_end:
3049 }
3050 }

```

The command `\@@_actually_draw_Ddots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool`.

```

3051 \cs_new_protected:Npn \@@_actually_draw_Ddots:
3052 {
3053   \bool_if:NTF \l_@@_initial_open_bool
3054   {
3055     \@@_open_y_initial_dim:
3056     % \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
3057     % \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3058     \@@_open_x_initial_dim:
3059   }
3060   { \@@_set_initial_coords_from_anchor:n { south-east } }
3061   \bool_if:NTF \l_@@_final_open_bool
3062   {
3063     % \@@_open_y_final_dim:
3064     % \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
3065     \@@_open_x_final_dim:
3066     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3067   }
3068   { \@@_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.

```

3069   \bool_if:NT \l_@@_parallelize_diags_bool
3070   {
3071     \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).

```
3072 \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.

```
3073 {
3074   \dim_gset:Nn \g_@@_delta_x_one_dim
3075   { \l_@@_x_final_dim - \l_@@_x_initial_dim }
3076   \dim_gset:Nn \g_@@_delta_y_one_dim
3077   { \l_@@_y_final_dim - \l_@@_y_initial_dim }
3078 }
```

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`.

```
3079 {
3080   \dim_set:Nn \l_@@_y_final_dim
3081   {
3082     \l_@@_y_initial_dim +
3083     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3084     \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
3085   }
3086 }
3087 }
3088 \@@_draw_line:
3089 }
```

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.

```
3090 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
3091 {
3092   \@@_adjust_to_submatrix:nn { #1 } { #2 }
3093   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
3094   {
3095     \@@_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.

```
3096   \group_begin:
3097   \keys_set:nn { NiceMatrix / xdots } { #3 }
3098   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3099   \@@_actually_draw_Iddots:
3100   \group_end:
3101 }
3102 }
```

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`.

```

3103 \cs_new_protected:Npn \@@_actually_draw_Iddots:
3104 {
3105   \bool_if:NTF \l_@@_initial_open_bool
3106   {
3107     \@@_open_y_initial_dim:
3108     \@@_open_x_initial_dim:
3109   }
3110   { \@@_set_initial_coords_from_anchor:n { south-west } }
3111   \bool_if:NTF \l_@@_final_open_bool
3112   {
3113     \@@_open_y_final_dim:
3114     \@@_open_x_final_dim:
3115   }
3116   { \@@_set_final_coords_from_anchor:n { north-east } }
3117   \bool_if:NT \l_@@_parallelize_diags_bool
3118   {
3119     \int_gincr:N \g_@@_iddots_int
3120     \int_compare:nNnTF \g_@@_iddots_int = 1
3121     {
3122       \dim_gset:Nn \g_@@_delta_x_two_dim
3123       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
3124       \dim_gset:Nn \g_@@_delta_y_two_dim
3125       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
3126     }
3127     {
3128       \dim_set:Nn \l_@@_y_final_dim
3129       {
3130         \l_@@_y_initial_dim +
3131         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3132         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
3133       }
3134     }
3135   }
3136   \@@_draw_line:
3137 }

```

## 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`

```

3138 \cs_new_protected:Npn \@@_draw_line:
3139 {
3140   \pgfrememberpicturepositiononpagetrue
3141   \pgf@relevantforpicturesizefalse
3142   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
3143   \@@_draw_standard_dotted_line:
3144   \@@_draw_non_standard_dotted_line:
3145 }

```

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.

```

3146 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
3147 {
3148   \begin { scope }
3149   \exp_args:No \@@_draw_non_standard_dotted_line:n
3150     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
3151 }

```

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.

```

3152 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
3153 {
3154   \@@_draw_non_standard_dotted_line:nVV
3155     { #1 }
3156     \l_@@_xdots_up_tl
3157     \l_@@_xdots_down_tl
3158 }

3159 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:nnn #1 #2 #3
3160 {
3161   \draw
3162     [
3163       #1 ,
3164       shorten~> = \l_@@_xdots_shorten_dim ,
3165       shorten~< = \l_@@_xdots_shorten_dim ,
3166     ]
3167     ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )

3168   -- node [ sloped , above ] { $ \scriptstyle #2 $ }
3169     node [ sloped , below ] { $ \scriptstyle #3 $ }
3170     ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
3171   \end { scope }
3172 }
3173 \cs_generate_variant:Nn \@@_draw_non_standard_dotted_line:nnn { n V V }

```

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

```

3174 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
3175 {
3176   \bool_lazy_and:nnF
3177     { \tl_if_empty_p:N \l_@@_xdots_up_tl }
3178     { \tl_if_empty_p:N \l_@@_xdots_down_tl }
3179   {
3180     \pgfscope
3181     \pgftransformshift
3182       {
3183         \pgfpointlineattime { 0.5 }
3184         { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
3185         { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
3186       }
3187     \pgftransformrotate
3188       {
3189         \fp_eval:n
3190         {
3191           atand
3192             (
3193               \l_@@_y_final_dim - \l_@@_y_initial_dim ,
3194               \l_@@_x_final_dim - \l_@@_x_initial_dim
3195             )

```

```

3196         }
3197     }
3198     \pgfnode
3199     { rectangle }
3200     { south }
3201     {
3202         \c_math_toggle_token
3203         \scriptstyle \l_@@_xdots_up_tl
3204         \c_math_toggle_token
3205     }
3206     { }
3207     { \pgfusepath { } }
3208     \pgfnode
3209     { rectangle }
3210     { north }
3211     {
3212         \c_math_toggle_token
3213         \scriptstyle \l_@@_xdots_down_tl
3214         \c_math_toggle_token
3215     }
3216     { }
3217     { \pgfusepath { } }
3218     \endpgfscope
3219 }
3220 \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.

```

3221 \dim_zero_new:N \l_@@_l_dim
3222 \dim_set:Nn \l_@@_l_dim
3223 {
3224     \fp_to_dim:n
3225     {
3226         sqrt
3227         (
3228             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
3229             +
3230             ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
3231         )
3232     }
3233 }

```

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.

```

3234 \bool_lazy_or:nnF
3235 { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
3236 { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
3237 \@@_draw_standard_dotted_line_i:
3238 \group_end:
3239 }
3240 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
3241 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
3242 {

```

The number of dots will be `\l_tmpa_int + 1`.

```

3243 \bool_if:NTF \l_@@_initial_open_bool
3244 {
3245     \bool_if:NTF \l_@@_final_open_bool
3246     {
3247         \int_set:Nn \l_tmpa_int
3248         { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }

```

```

3249     }
3250     {
3251         \int_set:Nn \l_tmpa_int
3252         {
3253             \dim_ratio:nn
3254             { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
3255             \l_@@_inter_dots_dim
3256         }
3257     }
3258 }
3259 {
3260     \bool_if:NTF \l_@@_final_open_bool
3261     {
3262         \int_set:Nn \l_tmpa_int
3263         {
3264             \dim_ratio:nn
3265             { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
3266             \l_@@_inter_dots_dim
3267         }
3268     }
3269     {
3270         \int_set:Nn \l_tmpa_int
3271         {
3272             \dim_ratio:nn
3273             { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
3274             \l_@@_inter_dots_dim
3275         }
3276     }
3277 }

```

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

```

3278     \dim_set:Nn \l_tmpa_dim
3279     {
3280         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3281         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
3282     }
3283     \dim_set:Nn \l_tmpb_dim
3284     {
3285         ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
3286         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
3287     }

```

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`.

```

3288     \int_set:Nn \l_tmpb_int
3289     {
3290         \bool_if:NTF \l_@@_initial_open_bool
3291         { \bool_if:NTF \l_@@_final_open_bool 1 0 }
3292         { \bool_if:NTF \l_@@_final_open_bool 2 1 }
3293     }

```

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.

```

3294     \dim_gadd:Nn \l_@@_x_initial_dim
3295     {
3296         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3297         \dim_ratio:nn
3298         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3299         { 2 \l_@@_l_dim }
3300         * \l_tmpb_int
3301     }

```



```

3302 \dim_gadd:Nn \l_@@_y_initial_dim
3303 {
3304   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
3305   \dim_ratio:nn
3306   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3307   { 2 \l_@@_l_dim }
3308   * \l_tmpb_int
3309 }
3310 \pgf@relevantforpicturesizefalse
3311 \int_step_inline:nnn 0 \l_tmpa_int
3312 {
3313   \pgfpathcircle
3314   { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
3315   { \l_@@_radius_dim }
3316   \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
3317   \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
3318 }
3319 \pgfusepathqfill
3320 }

```

## User commands available in the new environments

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

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

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

```

3321 \AtBeginDocument
3322 {
3323   \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
3324   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3325   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
3326   {
3327     \int_compare:nNnTF \c@jCol = 0
3328     { \@@_error:nn { in~first~col } \Ldots }
3329     {
3330       \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3331       { \@@_error:nn { in~last~col } \Ldots }
3332       {
3333         \@@_instruction_of_type:nnn \c_false_bool { \Ldots }
3334         { #1 , down = #2 , up = #3 }
3335       }
3336     }
3337     \bool_if:NF \l_@@_nullify_dots_bool
3338     { \phantom { \ensuremath { \@@_old_ldots } } }
3339     \bool_gset_true:N \g_@@_empty_cell_bool
3340   }

3341   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
3342   {
3343     \int_compare:nNnTF \c@jCol = 0
3344     { \@@_error:nn { in~first~col } \Cdots }

```

```

3345     {
3346         \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3347         { \@@_error:nn { in~last~col } \Cdots }
3348         {
3349             \@@_instruction_of_type:nnn \c_false_bool { Cdots }
3350             { #1 , down = #2 , up = #3 }
3351         }
3352     }
3353     \bool_if:NF \l_@@_nullify_dots_bool
3354     { \phantom { \ensuremath { \@@_old_cdots } } }
3355     \bool_gset_true:N \g_@@_empty_cell_bool
3356 }

\exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
{
3359     \int_compare:nNnTF \c@iRow = 0
3360     { \@@_error:nn { in~first~row } \Vdots }
3361     {
3362         \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
3363         { \@@_error:nn { in~last~row } \Vdots }
3364         {
3365             \@@_instruction_of_type:nnn \c_false_bool { Vdots }
3366             { #1 , down = #2 , up = #3 }
3367         }
3368     }
3369     \bool_if:NF \l_@@_nullify_dots_bool
3370     { \phantom { \ensuremath { \@@_old_vdots } } }
3371     \bool_gset_true:N \g_@@_empty_cell_bool
3372 }

\exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
{
3375     \int_case:nnF \c@iRow
3376     {
3377         0 { \@@_error:nn { in~first~row } \Ddots }
3378         \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
3379     }
3380     {
3381         \int_case:nnF \c@jCol
3382         {
3383             0 { \@@_error:nn { in~first~col } \Ddots }
3384             \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
3385         }
3386         {
3387             \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3388             \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Ddots }
3389             { #1 , down = #2 , up = #3 }
3390         }
3391     }
3392 }
3393 \bool_if:NF \l_@@_nullify_dots_bool
3394 { \phantom { \ensuremath { \@@_old_ddots } } }
3395 \bool_gset_true:N \g_@@_empty_cell_bool
3396 }

\exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
{
3399     \int_case:nnF \c@iRow
3400     {
3401         0 { \@@_error:nn { in~first~row } \Iddots }
3402         \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }

```

```

3403     }
3404     {
3405         \int_case:nnF \c@jCol
3406         {
3407             0 { \@@_error:nn { in~first~col } \Iddots }
3408             \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
3409         }
3410         {
3411             \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3412             \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Iddots }
3413             { #1 , down = #2 , up = #3 }
3414         }
3415     }
3416     \bool_if:NF \l_@@_nullify_dots_bool
3417     { \phantom { \ensuremath { \@@_old_iddots } } }
3418     \bool_gset_true:N \g_@@_empty_cell_bool
3419 }
3420 }

```

End of the \AtBeginDocument.

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

```

3421 \keys_define:nn { NiceMatrix / Ddots }
3422 {
3423     draw-first .bool_set:N = \l_@@_draw_first_bool ,
3424     draw-first .default:n = true ,
3425     draw-first .value_forbidden:n = true
3426 }

```

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

```

3427 \cs_new_protected:Npn \@@_Hspace:
3428 {
3429     \bool_gset_true:N \g_@@_empty_cell_bool
3430     \hspace
3431 }

```

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

```

3432 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
3433 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
3434 {

```

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

```

3435     \exp_args:NNe
3436     \@@_old_multicolumn
3437     { #1 }
3438     {
3439         \exp_args:Ne \str_case:nn { \str_foldcase:n { #2 } }
3440         {
3441             l { > \@@_Cell: l < \@@_end_Cell: }
3442             r { > \@@_Cell: r < \@@_end_Cell: }
3443             c { > \@@_Cell: c < \@@_end_Cell: }
3444             { l | } { > \@@_Cell: l < \@@_end_Cell: | }
3445             { r | } { > \@@_Cell: r < \@@_end_Cell: | }
3446             { c | } { > \@@_Cell: c < \@@_end_Cell: | }
3447             { | l } { | > \@@_Cell: l < \@@_end_Cell: }
3448             { | r } { | > \@@_Cell: r < \@@_end_Cell: }
3449             { | c } { | > \@@_Cell: c < \@@_end_Cell: }
3450             { | l | } { | > \@@_Cell: l < \@@_end_Cell: | }
3451             { | r | } { | > \@@_Cell: r < \@@_end_Cell: | }
3452             { | c | } { | > \@@_Cell: c < \@@_end_Cell: | }
3453         }
3454     }
3455     { #3 }

```

The `\peek_remove_spaces:n` is mandatory.

```

3456 \peek_remove_spaces:n
3457 {
3458   \int_compare:nNnT #1 > 1
3459   {
3460     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
3461     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3462     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
3463     \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
3464     {
3465       { \int_use:N \c@iRow }
3466       { \int_use:N \c@jCol }
3467       { \int_use:N \c@iRow }
3468       { \int_eval:n { \c@jCol + #1 - 1 } }
3469     }
3470   }
3471   \int_gadd:Nn \c@jCol { #1 - 1 }
3472   \int_compare:nNnT \c@jCol > \g_@@_col_total_int
3473   { \int_gset_eq:NN \g_@@_col_total_int \c@jCol }
3474 }
3475 }

```

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`.

```

3476 \cs_new:Npn \@@_Hdotsfor:
3477 {
3478   \bool_lazy_and:nnTF
3479   { \int_compare_p:nNn \c@jCol = 0 }
3480   { \int_compare_p:nNn \l_@@_first_col_int = 0 }
3481   {
3482     \bool_if:NTF \g_@@_after_col_zero_bool
3483     {
3484       \multicolumn { 1 } { c } { }
3485       \@@_Hdotsfor_i
3486     }
3487     { \@@_fatal:n { Hdotsfor~in~col~0 } }
3488   }
3489   {
3490     \multicolumn { 1 } { c } { }
3491     \@@_Hdotsfor_i
3492   }
3493 }

```

The command `\@@_Hdotsfor_i` is defined with `\NewDocumentCommand` because it has an optional argument. Note that such a command defined by `\NewDocumentCommand` is protected and that's why we have put the `\multicolumn` before (in the definition of `\@@_Hdotsfor:`).

```

3494 \AtBeginDocument
3495 {
3496   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3497   \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.

```

3498 \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
3499 {
3500   \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
3501   {
3502     \@@_Hdotsfor:nnnn
3503     { \int_use:N \c@iRow }
3504     { \int_use:N \c@jCol }
3505     { #2 }
3506     {

```

```

3507         #1 , #3 ,
3508         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3509     }
3510 }
3511 \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }
3512 }
3513 }

```

Enf of \AtBeginDocument.

```

3514 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
3515 {
3516     \bool_set_false:N \l_@@_initial_open_bool
3517     \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

3518     \int_set:Nn \l_@@_initial_i_int { #1 }
3519     \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

3520     \int_compare:nNnTF { #2 } = 1
3521     {
3522         \int_set:Nn \l_@@_initial_j_int 1
3523         \bool_set_true:N \l_@@_initial_open_bool
3524     }
3525     {
3526         \cs_if_exist:cTF
3527         {
3528             pgf @ sh @ ns @ \@@_env:
3529             - \int_use:N \l_@@_initial_i_int
3530             - \int_eval:n { #2 - 1 }
3531         }
3532         { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
3533         {
3534             \int_set:Nn \l_@@_initial_j_int { #2 }
3535             \bool_set_true:N \l_@@_initial_open_bool
3536         }
3537     }
3538     \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
3539     {
3540         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3541         \bool_set_true:N \l_@@_final_open_bool
3542     }
3543     {
3544         \cs_if_exist:cTF
3545         {
3546             pgf @ sh @ ns @ \@@_env:
3547             - \int_use:N \l_@@_final_i_int
3548             - \int_eval:n { #2 + #3 }
3549         }
3550         { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
3551         {
3552             \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3553             \bool_set_true:N \l_@@_final_open_bool
3554         }
3555     }
3556     \group_begin:
3557     \int_compare:nNnTF { #1 } = 0
3558     { \color { nicematrix-first-row } }
3559     {
3560         \int_compare:nNnT { #1 } = \g_@@_row_total_int
3561         { \color { nicematrix-last-row } }
3562     }
3563     \keys_set:nn { NiceMatrix / xdots } { #4 }

```

```

3564 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3565 \@@_actually_draw_ldots:
3566 \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).

```

3567 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
3568 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
3569 }

3570 \AtBeginDocument
3571 {
3572   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3573   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3574   \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
3575   {
3576     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
3577     {
3578       \@@_Vdotsfor:nnnn
3579       { \int_use:N \c@iRow }
3580       { \int_use:N \c@jCol }
3581       { #2 }
3582       {
3583         #1 , #3 ,
3584         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3585       }
3586     }
3587   }
3588 }

```

Enf of `\AtBeginDocument`.

```

3589 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
3590 {
3591   \bool_set_false:N \l_@@_initial_open_bool
3592   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it’s easy.

```

3593 \int_set:Nn \l_@@_initial_j_int { #2 }
3594 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

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

```

3595 \int_compare:nNnTF #1 = 1
3596 {
3597   \int_set:Nn \l_@@_initial_i_int 1
3598   \bool_set_true:N \l_@@_initial_open_bool
3599 }
3600 {
3601   \cs_if_exist:cTF
3602   {
3603     pgf @ sh @ ns @ \@@_env:
3604     - \int_eval:n { #1 - 1 }
3605     - \int_use:N \l_@@_initial_j_int
3606   }
3607   { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
3608   {
3609     \int_set:Nn \l_@@_initial_i_int { #1 }
3610     \bool_set_true:N \l_@@_initial_open_bool
3611   }
3612 }
3613 \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
3614 {
3615   \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }

```

```

3616     \bool_set_true:N \l_@@_final_open_bool
3617   }
3618   {
3619     \cs_if_exist:cTF
3620     {
3621       pgf @ sh @ ns @ \@@_env:
3622       - \int_eval:n { #1 + #3 }
3623       - \int_use:N \l_@@_final_j_int
3624     }
3625     { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
3626     {
3627       \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3628       \bool_set_true:N \l_@@_final_open_bool
3629     }
3630   }
3631   \group_begin:
3632   \int_compare:nNnTF { #2 } = 0
3633   { \color { nicematrix-first-col } }
3634   {
3635     \int_compare:nNnT { #2 } = \g_@@_col_total_int
3636     { \color { nicematrix-last-col } }
3637   }
3638   \keys_set:nn { NiceMatrix / xdots } { #4 }
3639   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3640   \@@_actually_draw_Vdots:
3641   \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).

```

3642   \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
3643   { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
3644 }

```

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

```

3645 \cs_new_protected:Npn \@@_rotate: { \bool_gset_true:N \g_@@_rotate_bool }

```

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

In the `\CodeAfter`, the command `\@@_line:nn` will be linked to `\line`. This command takes two arguments which are the specifications of two cells in the array (in the format  $i$ - $j$ ) and draws a dotted line between these cells.

First, we write a command with an argument of the format  $i$ - $j$  and applies the command `\int_eval:n` to  $i$  and  $j$ ; this must *not* be protected (and is, of course fully expandable).<sup>60</sup>

```

3646 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
3647 { \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).

```

3648 \AtBeginDocument
3649 {
3650   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }

```

<sup>60</sup>Indeed, we want that the user may use the command `\line` in `\CodeAfter` with LaTeX counters in the arguments — with the command `\value`.

```

3651 \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3652 \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
3653 {
3654   \group_begin:
3655   \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
3656   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3657   \use:e
3658   {
3659     \@@_line_i:nn
3660     { \@@_double_int_eval:n #2 \q_stop }
3661     { \@@_double_int_eval:n #3 \q_stop }
3662   }
3663   \group_end:
3664 }
3665 }

3666 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3667 {
3668   \bool_set_false:N \l_@@_initial_open_bool
3669   \bool_set_false:N \l_@@_final_open_bool
3670   \bool_if:nTF
3671   {
3672     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3673     ||
3674     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3675   }
3676   {
3677     \@@_error:nnn { unknown~cell~for~line~in~CodeAfter } { #1 } { #2 }
3678   }
3679   { \@@_draw_line_ii:nn { #1 } { #2 } }
3680 }

3681 \AtBeginDocument
3682 {
3683   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3684   {

```

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:.`

```

3685   \c_@@_pgfortikzpicture_tl
3686   \@@_draw_line_iii:nn { #1 } { #2 }
3687   \c_@@_endpgfortikzpicture_tl
3688 }
3689 }

```

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

```

3690 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3691 {
3692   \pgfrememberpicturepositiononpagetrue
3693   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3694   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3695   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3696   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3697   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3698   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3699   \@@_draw_line:
3700 }

```

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots` don’t use this command because they have to do other settings (for example, the diagonal lines must be parallelized).



## Colors of cells, rows and columns

We want to avoid the thin white lines that are shown in some PDF viewers (eg: with the engine MuPDF used by SumatraPDF). That's why we try to draw rectangles of the same color in the same instruction `\pgfusepath { fill }` (and they will be in the same instruction `fill`—coded `f`— in the resulting PDF).

The commands `\@@_rowcolor`, `\@@_columncolor` and `\@@_rectanglecolor` (which are linked to `\rowcolor`, `\columncolor` and `\rectanglecolor` before the execution of the `code-before`) don't directly draw the corresponding rectangles. Instead, they store their instructions color by color:

- A sequence `\g_@@_colors_seq` will be built containing all the colors used by at least one of these instructions. Each *color* may be prefixed by its color model (eg: `[gray]{0.5}`).
- For the color whose index in `\g_@@_colors_seq` is equal to *i*, a list of instructions which use that color will be constructed in the token list `\g_@@_color_i_tl`. In that token list, the instructions will be written using `\@@_rowcolor:n`, `\@@_columncolor:n` and `\@@_rectanglecolor:nn` (corresponding of `\@@_rowcolor`, `\@@_columncolor` and `\@@_rectanglecolor`).

`bigskip #1` is the color and `#2` is an instruction using that color. Despite its name, the command `\@@_add_to_color_seq` doesn't only add a color to `\g_@@_colors_seq`: it also updates the corresponding token list `\g_@@_color_i_tl`. We add in a global way because the final user may use the instructions such as `\cellcolor` in a loop of `pgffor` in the `\CodeBefore` (and we recall that a loop of `pgffor` is encapsulated in a group).

```
3701 \cs_new_protected:Npn \@@_add_to_colors_seq:nn #1 #2
3702 {
```

First, we look for the number of the color and, if it's found, we store it in `\l_tmpa_int`. If the color is not present in `\l_@@_colors_seq`, `\l_tmpa_int` will remain equal to 0.

```
3703   \int_zero:N \l_tmpa_int
3704   \seq_map_indexed_inline:Nn \g_@@_colors_seq
3705     { \tl_if_eq:nnT { #1 } { ##2 } { \int_set:Nn \l_tmpa_int { ##2 } } }
3706   \int_compare:nNnTF \l_tmpa_int = \c_zero_int
```

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

```
3707     {
3708       \seq_gput_right:Nn \g_@@_colors_seq { #1 }
3709       \tl_gset:cx { g_@@_color _ \seq_count:N \g_@@_colors_seq _ tl } { #2 }
3710     }
```

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

```
3711     { \tl_gput_right:cx { g_@@_color _ \int_use:N \l_tmpa_int _ tl } { #2 } }
3712   }
```

```
3713 \cs_generate_variant:Nn \@@_add_to_colors_seq:nn { x n }
```

The macro `\@@_actually_color:` will actually fill all the rectangles, color by color (using the sequence `\l_@@_colors_seq` and all the token lists of the form `\l_@@_color_i_tl`).

```
3714 \cs_new_protected:Npn \@@_actually_color:
3715 {
3716   \pgfpicture
3717   \pgf@relevantforpicturesizefalse
3718   \seq_map_indexed_inline:Nn \g_@@_colors_seq
3719     {
3720       \color ##2
3721       \use:c { g_@@_color _ ##1 _ tl }
3722       \tl_gclear:c { g_@@_color _ ##1 _ tl }
3723       \pgfusepath { fill }
3724     }
3725   \endpgfpicture
3726 }
```

```

3727 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3728 {
3729   \tl_set:Nn \l_tmpa_tl { #1 }
3730   \tl_set:Nn \l_tmpb_tl { #2 }
3731 }

```

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

```

3732 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3733 {
3734   \tl_if_blank:nF { #2 }
3735   {
3736     \@@_add_to_colors_seq:xn
3737     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3738     { \@@_rowcolor:n { #3 } }
3739   }
3740 }
3741 \cs_new_protected:Npn \@@_rowcolor:n #1
3742 {
3743   \tl_set:Nn \l_@@_rows_tl { #1 }
3744   \tl_set:Nn \l_@@_cols_tl { - }

```

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

```

3745   \@@_cartesian_path:
3746 }

```

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

```

3747 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3748 {
3749   \tl_if_blank:nF { #2 }
3750   {
3751     \@@_add_to_colors_seq:xn
3752     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3753     { \@@_columncolor:n { #3 } }
3754   }
3755 }
3756 \cs_new_protected:Npn \@@_columncolor:n #1
3757 {
3758   \tl_set:Nn \l_@@_rows_tl { - }
3759   \tl_set:Nn \l_@@_cols_tl { #1 }

```

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

```

3760   \@@_cartesian_path:
3761 }

```

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

```

3762 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3763 {
3764   \tl_if_blank:nF { #2 }
3765   {
3766     \@@_add_to_colors_seq:xn
3767     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3768     { \@@_rectanglecolor:nnn { #3 } { #4 } { 0 pt } }
3769   }
3770 }

```

The last argument is the radius of the corners of the rectangle.

```

3771 \NewDocumentCommand \@@_roundedrectanglecolor { 0 { } m m m m }
3772 {
3773   \tl_if_blank:nF { #2 }
3774   {
3775     \@@_add_to_colors_seq:xn
3776     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3777     { \@@_rectanglecolor:nnn { #3 } { #4 } { #5 } }
3778   }
3779 }

```

The last argument is the radius of the corners of the rectangle.

```

3780 \cs_new_protected:Npn \@@_rectanglecolor:nnn #1 #2 #3
3781 {
3782   \@@_cut_on_hyphen:w #1 \q_stop
3783   \tl_clear_new:N \l_tmpc_tl
3784   \tl_clear_new:N \l_tmpd_tl
3785   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
3786   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
3787   \@@_cut_on_hyphen:w #2 \q_stop
3788   \tl_set:Nx \l_@@_rows_tl { \l_tmpc_tl - \l_tmpa_tl }
3789   \tl_set:Nx \l_@@_cols_tl { \l_tmpd_tl - \l_tmpb_tl }

```

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

```

3790   \@@_cartesian_path:n { #3 }
3791 }

```

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

```

3792 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3793 {
3794   \clist_map_inline:nn { #3 }
3795   { \@@_rectanglecolor [ #1 ] { #2 } { ##1 } { ##1 } }
3796 }

```

```

3797 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3798 {
3799   \int_step_inline:nn { \int_use:N \c@iRow }
3800   {
3801     \int_step_inline:nn { \int_use:N \c@jCol }
3802     {
3803       \int_if_even:nTF { ####1 + ##1 }
3804       { \@@_cellcolor [ #1 ] { #2 } }
3805       { \@@_cellcolor [ #1 ] { #3 } }
3806     } { ##1 - ####1 }
3807   }
3808 }
3809 }

```

```

3810 \keys_define:nn { NiceMatrix / arraycolor }
3811 { except-corners .code:n = \@@_error:n { key except-corners } }

```

The command `\@@_arraycolor` (linked to `\arraycolor` at the beginning of the `\CodeBefore`) will color the whole tabular (excepted the potential exterior rows and columns). The third argument is a optional argument which a list of pairs key-value. As for now, there is only one key: `except-corners`. When that key is used, the cells in the corners are not colored.

```

3812 \NewDocumentCommand \@@_arraycolor { 0 { } m 0 { } }
3813 {
3814   \keys_set:nn { NiceMatrix / arraycolor } { #3 }
3815   \@@_rectanglecolor [ #1 ] { #2 }

```

```

3816     { 1 - 1 }
3817     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3818 }

3819 \keys_define:nn { NiceMatrix / rowcolors }
3820 {
3821     respect-blocks .bool_set:N = \l_@@_respect_blocks_bool ,
3822     respect-blocks .default:n = true ,
3823     cols .tl_set:N = \l_@@_cols_tl ,
3824     restart .bool_set:N = \l_@@_rowcolors_restart_bool ,
3825     restart .default:n = true ,
3826     unknown .code:n = \@@_error:n { Unknown-key-for-rowcolors }
3827 }

```

The command `\rowcolors` (accessible in the `code-before`) is inspired by the command `\rowcolors` of the package `xcolor` (with the option `table`). However, the command `\rowcolors` of `nicematrix` has *not* the optional argument of the command `\rowcolors` of `xcolor`. Here is an example: `\rowcolors{1}{blue!10}{}[respect-blocks]`.

#1 (optional) is the color space ; #2 is a list of intervals of rows ; #3 is the first color ; #4 is the second color ; #5 is for the optional list of pairs key-value.

```

3828 \NewDocumentCommand \@@_rowcolors { 0 { } m m m 0 { } }
3829 {

```

The group is for the options.

```

3830     \group_begin:
3831     \tl_clear_new:N \l_@@_cols_tl
3832     \tl_set:Nn \l_@@_cols_tl { - }
3833     \keys_set:nn { NiceMatrix / rowcolors } { #5 }

```

The boolean `\l_tmpa_bool` will indicate whereas we are in a row of the first color or of the second color.

```

3834     \bool_set_true:N \l_tmpa_bool
3835     \bool_lazy_and:nnT
3836         \l_@@_respect_blocks_bool
3837         {
3838             \cs_if_exist_p:c
3839                 { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq } }
3840     {

```

We don't want to take into account a block which is completely in the “first column” of (number 0) or in the “last column” and that's why we filter the sequence of the blocks (in a the sequence `\l_tmpa_seq`).

```

3841         \seq_set_eq:Nc \l_tmpb_seq
3842         { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3843         \seq_set_filter:Nn \l_tmpa_seq \l_tmpb_seq
3844         { \@@_not_in_exterior_p:n #1 }
3845     }
3846     \pgfpicture
3847     \pgf@relevantforpicturesizefalse
3848     \clist_map_inline:nn { #2 }
3849     {
3850         \tl_set:Nn \l_tmpa_tl { #1 }
3851         \tl_if_in:NnTF \l_tmpa_tl { - }
3852             { \@@_cut_on_hyphen:w #1 \q_stop }
3853             { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }

```

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

```

3854         \int_set:Nn \l_tmpa_int \l_tmpa_tl
3855         \bool_if:NTF \l_@@_rowcolors_restart_bool
3856             { \bool_set_true:N \l_tmpa_bool }
3857             { \bool_set:Nn \l_tmpa_bool { \int_if_odd_p:n { \l_tmpa_tl } } }
3858         \int_zero_new:N \l_tmppc_int
3859         \int_set:Nn \l_tmppc_int \l_tmpb_tl

```

```

3860 \int_do_until:nNnn \l_tmpa_int > \l_tmpe_int
3861 {

```

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

```

3862 \int_set_eq:NN \l_tmpe_int \l_tmpa_int

```

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

```

3863 \bool_lazy_and:nnT
3864 \l_@@_respect_blocks_bool
3865 {
3866   \cs_if_exist_p:c
3867   { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3868 }
3869 {
3870   \seq_set_filter:NNn \l_tmpe_seq \l_tmpa_seq
3871   { \@@_intersect_our_row_p:nnnn ###1 }
3872   \seq_map_inline:Nn \l_tmpe_seq { \@@_rowcolors_i:nnnn ###1 }

```

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

```

3873 }
3874 \tl_set:Nx \l_@@_rows_tl
3875 { \int_use:N \l_tmpa_int - \int_use:N \l_tmpe_int }
3876 \bool_if:NTF \l_tmpe_bool
3877 {
3878   \tl_if_blank:nF { #3 }
3879   {
3880     \tl_if_empty:nTF { #1 }
3881     \color
3882     { \color [ #1 ] }
3883     { #3 }

```

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

```

3884 \@@_cartesian_path:
3885 \pgfusepath { fill }
3886 }
3887 \bool_set_false:N \l_tmpe_bool
3888 }
3889 {
3890   \tl_if_blank:nF { #4 }
3891   {
3892     \tl_if_empty:nTF { #1 }
3893     \color
3894     { \color [ #1 ] }
3895     { #4 }

```

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

```

3896 \@@_cartesian_path:
3897 \pgfusepath { fill }
3898 }
3899 \bool_set_true:N \l_tmpe_bool
3900 }
3901 \int_set:Nn \l_tmpa_int { \l_tmpe_int + 1 }
3902 }
3903 }
3904 \endpgfpicture
3905 \group_end:
3906 }

```

```

3907 \cs_new_protected:Npn \@@_rowcolors_i:nnnn #1 #2 #3 #4
3908 {
3909   \int_compare:nNnT { #3 } > \l_tmpe_int
3910   { \int_set:Nn \l_tmpe_int { #3 } }
3911 }

```

```

3912 \prg_new_conditional:Nnn \@@_not_in_exterior:nnnn p
3913 {
3914   \bool_lazy_or:nnTF
3915   { \int_compare_p:nNn { #4 } = \c_zero_int }
3916   { \int_compare_p:nNn { #2 } = { \@@_succ:n { \c@jCol } } }
3917   \prg_return_false:
3918   \prg_return_true:
3919 }

```

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

```

3920 \prg_new_conditional:Nnn \@@_intersect_our_row:nnnn p
3921 {
3922   \bool_if:nTF
3923   {
3924     \int_compare_p:n { #1 <= \l_tmpa_int }
3925     &&
3926     \int_compare_p:n { \l_tmpa_int <= #3 }
3927   }
3928   \prg_return_true:
3929   \prg_return_false:
3930 }

```

The following command uses two implicit arguments: \l\_@@\_rows\_tl and \l\_@@\_cols\_tl which are specifications for a set of rows and a set of columns. It creates a path but does *not* fill it. It must be filled by another command after. The argument is the radius of the corners. We define below a command \@@\_cartesian\_path: which corresponds to a value 0 pt for the radius of the corners. This command is in particular used in \@@\_rectanglecolor:nnn (used in \@@\_rectanglecolor, itself used in \@@\_cellcolor).

```

3931 \cs_new_protected:Npn \@@_cartesian_path:n #1
3932 {
3933   \bool_lazy_and:nnT
3934   { ! \seq_if_empty_p:N \l_@@_corners_cells_seq }
3935   { \dim_compare_p:nNn { #1 } = \c_zero_dim }
3936   {
3937     \@@_expand_clist:NN \l_@@_cols_tl \c@jCol
3938     \@@_expand_clist:NN \l_@@_rows_tl \c@iRow
3939   }

```

We begin the loop over the columns.

```

3940   \clist_map_inline:Nn \l_@@_cols_tl
3941   {
3942     \tl_set:Nn \l_tmpa_tl { ##1 }
3943     \tl_if_in:NnTF \l_tmpa_tl { - }
3944     { \@@_cut_on_hyphen:w ##1 \q_stop }
3945     { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3946     \bool_lazy_or:nnT
3947     { \tl_if_blank_p:V \l_tmpa_tl }
3948     { \str_if_eq_p:Vn \l_tmpa_tl { * } }
3949     { \tl_set:Nn \l_tmpa_tl { 1 } }
3950     \bool_lazy_or:nnT
3951     { \tl_if_blank_p:V \l_tmpb_tl }
3952     { \str_if_eq_p:Vn \l_tmpb_tl { * } }
3953     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3954     \int_compare:nNnT \l_tmpb_tl > \c@jCol
3955     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }

```

\l\_tmpc\_tl will contain the number of column.

```

3956   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl

```

If we decide to provide the commands \cellcolor, \rectanglecolor, \rowcolor, \columncolor, \rowcolors and \chessboardcolors in the code-before of a \SubMatrix, we will have to modify the following line, by adding a kind of offset. We will have also some other lines to modify.

```

3957   \@@_qpoint:n { col - \l_tmpa_tl }

```

```

3958 \int_compare:nNnTF \l_@@_first_col_int = \l_tmpa_tl
3959 { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3960 { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3961 \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3962 \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }

```

We begin the loop over the rows.

```

3963 \clist_map_inline:Nn \l_@@_rows_tl
3964 {
3965   \tl_set:Nn \l_tmpa_tl { #####1 }
3966   \tl_if_in:NnTF \l_tmpa_tl { - }
3967   { \@@_cut_on_hyphen:w #####1 \q_stop }
3968   { \@@_cut_on_hyphen:w #####1 - #####1 \q_stop }
3969   \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3970   \tl_if_empty:NT \l_tmpb_tl
3971   { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3972   \int_compare:nNnT \l_tmpb_tl > \c@iRow
3973   { \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.

```

3974 \seq_if_in:NxF \l_@@_corners_cells_seq
3975 { \l_tmpa_tl - \l_tmpc_tl }
3976 {
3977   \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3978   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3979   \@@_qpoint:n { row - \l_tmpa_tl }
3980   \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3981   \pgfsetcornersarced { \pgfpoint { #1 } { #1 } }
3982   \pgfpathrectanglecorners
3983   { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3984   { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3985 }
3986 }
3987 }
3988 }

```

The following command corresponds to a radius of the corners equal to 0 pt. This command is used by the commands \@@\_rowcolors, \@@\_columncolor and \@@\_rowcolor:n (used in \@@\_rowcolor).

```

3989 \cs_new_protected:Npn \@@_cartesian_path: { \@@_cartesian_path:n { 0 pt } }

```

The following command will be used only with \l\_@@\_cols\_tl and \c@jCol (first case) or with \l\_@@\_rows\_tl and \c@iRow (second case). For instance, with \l\_@@\_cols\_tl equal to 2,4-6,8-\* and \c@jCol equal to 10, the clist \l\_@@\_cols\_tl will be replaced by 2,4,5,6,8,9,10.

```

3990 \cs_new_protected:Npn \@@_expand_clist:NN #1 #2
3991 {
3992   \clist_set_eq:NN \l_tmpa_clist #1
3993   \clist_clear:N #1
3994   \clist_map_inline:Nn \l_tmpa_clist
3995   {
3996     \tl_set:Nn \l_tmpa_tl { ##1 }
3997     \tl_if_in:NnTF \l_tmpa_tl { - }
3998     { \@@_cut_on_hyphen:w ##1 \q_stop }
3999     { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
4000     \bool_lazy_or:nnT
4001     { \tl_if_blank_p:V \l_tmpa_tl }
4002     { \str_if_eq_p:Vn \l_tmpa_tl { * } }
4003     { \tl_set:Nn \l_tmpa_tl { 1 } }
4004     \bool_lazy_or:nnT
4005     { \tl_if_blank_p:V \l_tmpb_tl }
4006     { \str_if_eq_p:Vn \l_tmpb_tl { * } }
4007     { \tl_set:Nx \l_tmpb_tl { \int_use:N #2 } }
4008     \int_compare:nNnT \l_tmpb_tl > #2
4009     { \tl_set:Nx \l_tmpb_tl { \int_use:N #2 } }

```

```

4010     \int_step_inline:nnn \l_tmpa_tl \l_tmpb_tl
4011     { \clist_put_right:Nn #1 { ###1 } }
4012   }
4013 }

```

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

```

4014 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
4015 {
4016   \peek_remove_spaces:n
4017   {
4018     \tl_gput_right:Nx \g_nicematrix_code_before_tl
4019     {

```

We must not expand the color (`#2`) because the color may contain the token `!` which may be activated by some packages (ex.: `babel` with the option `french` on `latex` and `pdflatex`).

```

4020       \cellcolor [ #1 ] { \exp_not:n { #2 } }
4021       { \int_use:N \c@iRow - \int_use:N \c@jCol }
4022     }
4023   }
4024 }

```

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

```

4025 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
4026 {
4027   \peek_remove_spaces:n
4028   {
4029     \tl_gput_right:Nx \g_nicematrix_code_before_tl
4030     {
4031       \exp_not:N \rectanglecolor [ #1 ] { \exp_not:n { #2 } }
4032       { \int_use:N \c@iRow - \int_use:N \c@jCol }
4033       { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
4034     }
4035   }
4036 }

```

```

4037 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
4038 {

```

With the following line, we test whether the cell is the first one we encounter in its column (don't forget that some rows may be incomplete).

```

4039   \int_compare:nNnT \c@jCol > \g_@@_col_total_int
4040   {

```

You use `gput_left` because we want the specification of colors for the columns drawn before the specifications of color for the rows (and the cells). Be careful: maybe this is not effective since we have an analyze of the instructions in the `\CodeBefore` in order to fill color by color (to avoid the thin white lines).

```

4041     \tl_gput_left:Nx \g_nicematrix_code_before_tl
4042     {
4043       \exp_not:N \columncolor [ #1 ]
4044       { \exp_not:n { #2 } } { \int_use:N \c@jCol }
4045     }
4046   }
4047 }

```



## 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`.

```
4048 \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`.

```
4049 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
4050 {
4051   \int_compare:nNnTF \l_@@_first_col_int = 0
4052     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
4053     {
4054       \int_compare:nNnTF \c@jCol = 0
4055         {
4056           \int_compare:nNnF \c@iRow = { -1 }
4057             { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
4058         }
4059       { \@@_OnlyMainNiceMatrix_i:n { #1 } }
4060     }
4061 }
```

This definition may seem complicated by we must remind that the number of row `\c@iRow` is incremented in the first cell of the row, *after* a potential vertical rule on the left side of the first cell.

The command `\@@_OnlyMainNiceMatrix_i:n` is only a short-cut which is used twice in the above command. This command must *not* be protected.

```
4062 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
4063 {
4064   \int_compare:nNnF \c@iRow = 0
4065     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
4066 }
```

Remember that `\c@iRow` is not always inferior to `\l_@@_last_row_int` because `\l_@@_last_row_int` may be equal to  $-2$  or  $-1$  (we can't write `\int_compare:nNnT \c@iRow < \l_@@_last_row_int`).

The following command will be executed in the `internal-code-after`. The rule will be drawn *before* the column `#1` (that is to say on the left side). `#2` is the number of consecutive occurrences of `|`.

```
4067 \cs_new_protected:Npn \@@_vline:nn #1 #2
4068 {
```

The following test is for the case where the user don't use all the columns specified in the preamble of the environment (for instance, a preamble of `|c|c|c|` but only two columns used).

```
4069   \int_compare:nNnT { #1 } < { \c@jCol + 2 }
4070   {
4071     \pgfpicture
4072       \@@_vline_i:nn { #1 } { #2 }
4073     \endpgfpicture
4074   }
4075 }
4076 \cs_new_protected:Npn \@@_vline_i:nn #1 #2
4077 {
```

`\l_tmpa_tl` is the number of row and `\l_tmpb_tl` the number of column. When we have found a row corresponding to a rule to draw, we note its number in `\l_tmpc_tl`.

```
4078   \tl_set:Nx \l_tmpb_tl { #1 }
4079   \tl_clear_new:N \l_tmpc_tl
4080   \int_step_variable:nNn \c@iRow \l_tmpa_tl
4081   {
```

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.

```

4082      \bool_gset_true:N \g_tmpa_bool
4083      \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4084        { \@@_test_vline_in_block:nnnn ##1 }
4085      \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
4086        { \@@_test_vline_in_block:nnnn ##1 }
4087      \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
4088        { \@@_test_vline_in_stroken_block:nnnn ##1 }
4089      \clist_if_empty:NF \l_@@_corners_clist
4090      \@@_test_in_corner_v:
4091      \bool_if:NTF \g_tmpa_bool
4092      {
4093        \tl_if_empty:NT \l_tmpc_tl

```

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

```

4094        { \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl }
4095      }
4096      {
4097        \tl_if_empty:NF \l_tmpc_tl
4098        {
4099          \@@_vline_ii:nnnn
4100          { #1 }
4101          { #2 }
4102          \l_tmpc_tl
4103          { \int_eval:n { \l_tmpa_tl - 1 } }
4104          \tl_clear:N \l_tmpc_tl
4105        }
4106      }
4107    }
4108    \tl_if_empty:NF \l_tmpc_tl
4109    {
4110      \@@_vline_ii:nnnn
4111      { #1 }
4112      { #2 }
4113      \l_tmpc_tl
4114      { \int_use:N \c@iRow }
4115      \tl_clear:N \l_tmpc_tl
4116    }
4117  }

```

```

4118  \cs_new_protected:Npn \@@_test_in_corner_v:
4119  {
4120    \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
4121    {
4122      \seq_if_in:NxT
4123        \l_@@_corners_cells_seq
4124        { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
4125        { \bool_set_false:N \g_tmpa_bool }
4126    }
4127    {
4128      \seq_if_in:NxT
4129        \l_@@_corners_cells_seq
4130        { \l_tmpa_tl - \l_tmpb_tl }
4131      {
4132        \int_compare:nNnTF \l_tmpb_tl = 1
4133        { \bool_set_false:N \g_tmpa_bool }
4134      }
4135      \seq_if_in:NxT
4136        \l_@@_corners_cells_seq
4137        { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }

```

```

4138         { \bool_set_false:N \g_tmpa_bool }
4139     }
4140 }
4141 }
4142 }

```

#1 is the number of the column; #2 is the number of vertical rules to draw (with potentially a color between); #3 and #4 are the numbers of the rows between which the rule has to be drawn.

```

4143 \cs_new_protected:Npn \@@_vline_ii:nnnn #1 #2 #3 #4
4144 {
4145     \pgfrememberpicturepositiononpagetrue
4146     \pgf@relevantforpicturesizefalse
4147     \@@_qpoint:n { row - #3 }
4148     \dim_set_eq:NN \l_tmpa_dim \pgf@y
4149     \@@_qpoint:n { col - #1 }
4150     \dim_set_eq:NN \l_tmpb_dim \pgf@x
4151     \@@_qpoint:n { row - \@@_succ:n { #4 } }
4152     \dim_set_eq:NN \l_tmpc_dim \pgf@y
4153     \bool_lazy_and:nnT
4154         { \int_compare_p:nNn { #2 } > 1 }
4155         { ! \tl_if_blank_p:V \CT@drsc@ }
4156     {
4157         \group_begin:
4158         \CT@drsc@
4159         \dim_add:Nn \l_tmpa_dim { 0.5 \arrayrulewidth }
4160         \dim_sub:Nn \l_tmpc_dim { 0.5 \arrayrulewidth }
4161         \dim_set:Nn \l_tmpd_dim
4162             { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
4163         \pgfpathrectanglecorners
4164             { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4165             { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
4166         \pgfusepath { fill }
4167         \group_end:
4168     }
4169     \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4170     \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4171     \prg_replicate:nn { #2 - 1 }
4172     {
4173         \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
4174         \dim_sub:Nn \l_tmpb_dim \doublerulesep
4175         \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4176         \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4177     }
4178     \CT@arc@
4179     \pgfsetlinewidth { 1.1 \arrayrulewidth }
4180     \pgfsetrectcap
4181     \pgfusepathqstroke
4182 }

```

The following command draws a complete vertical rule in the column #1 (#2 is the number of consecutive rules specified by the number of | in the preamble). This command will be used if there is no block in the array (and the key `corners` is not used).

```

4183 \cs_new_protected:Npn \@@_vline_i_complete:nn #1 #2
4184 { \@@_vline_ii:nnnn { #1 } { #2 } 1 { \int_use:N \c@iRow } }

```

The command `\@@_draw_hlines:` draws all the vertical rules excepted in the blocks, in the virtual blocks (determined by a command such as `\Cdots`) and in the corners (if the key `corners` is used).

```

4185 \cs_new_protected:Npn \@@_draw_vlines:
4186 {
4187     \int_step_inline:nnn
4188         { \bool_if:NTF \g_@@_NiceArray_bool 1 2 }

```

```

4189     { \bool_if:NTF \g_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
4190     {
4191         \tl_if_eq:NnF \l_@@_vlines_clist { all }
4192         { \clist_if_in:NnT \l_@@_vlines_clist { ##1 } }
4193         { \@@_vline:nn { ##1 } 1 }
4194     }
4195 }

```

## The horizontal rules

The following command will be executed in the `internal-code-after`. The rule will be drawn *before* the row #1. #2 is the number of consecutive occurrences of `\Hline`.

```

4196 \cs_new_protected:Npn \@@_hline:nn #1 #2
4197 {
4198     \pgfpicture
4199     \@@_hline_i:nn { #1 } { #2 }
4200     \endpgfpicture
4201 }
4202 \cs_new_protected:Npn \@@_hline_i:nn #1 #2
4203 {

```

`\l_tmpa_tl` is the number of row and `\l_tmpb_tl` the number of column. When we have found a column corresponding to a rule to draw, we note its number in `\l_tmppc_tl`.

```

4204     \tl_set:Nn \l_tmpa_tl { #1 }
4205     \tl_clear_new:N \l_tmppc_tl
4206     \int_step_variable:nNn \c@jCol \l_tmpb_tl
4207     {

```

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.

```

4208         \bool_gset_true:N \g_tmpa_bool
4209         \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4210         { \@@_test_hline_in_block:nnnn ##1 }
4211         \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
4212         { \@@_test_hline_in_block:nnnn ##1 }
4213         \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
4214         { \@@_test_hline_in_stroken_block:nnnn ##1 }
4215         \clist_if_empty:NF \l_@@_corners_clist \@@_test_in_corner_h:
4216         \bool_if:NTF \g_tmpa_bool
4217         {
4218             \tl_if_empty:NT \l_tmppc_tl

```

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

```

4219             { \tl_set_eq:NN \l_tmppc_tl \l_tmpb_tl }
4220         }
4221         {
4222             \tl_if_empty:NF \l_tmppc_tl
4223             {
4224                 \@@_hline_ii:nnnn
4225                 { #1 }
4226                 { #2 }
4227                 \l_tmppc_tl
4228                 { \int_eval:n { \l_tmpb_tl - 1 } }
4229                 \tl_clear:N \l_tmppc_tl
4230             }
4231         }
4232     }
4233     \tl_if_empty:NF \l_tmppc_tl
4234     {

```

```

4235 \@@_hline_ii:nnnn
4236 { #1 }
4237 { #2 }
4238 \l_tmpc_tl
4239 { \int_use:N \c@jCol }
4240 \tl_clear:N \l_tmpc_tl
4241 }
4242 }

4243 \cs_new_protected:Npn \@@_test_in_corner_h:
4244 {
4245 \int_compare:nNnTF \l_tmpa_tl = { \@@_succ:n \c@iRow }
4246 {
4247 \seq_if_in:NxT
4248 \l_@@_corners_cells_seq
4249 { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
4250 { \bool_set_false:N \g_tmpa_bool }
4251 }
4252 {
4253 \seq_if_in:NxT
4254 \l_@@_corners_cells_seq
4255 { \l_tmpa_tl - \l_tmpb_tl }
4256 {
4257 \int_compare:nNnTF \l_tmpa_tl = 1
4258 { \bool_set_false:N \g_tmpa_bool }
4259 {
4260 \seq_if_in:NxT
4261 \l_@@_corners_cells_seq
4262 { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
4263 { \bool_set_false:N \g_tmpa_bool }
4264 }
4265 }
4266 }
4267 }

```

#1 is the number of the row; #2 is the number of horizontal rules to draw (with potentially a color between); #3 and #4 are the number of the columns between which the rule has to be drawn.

```

4268 \cs_new_protected:Npn \@@_hline_ii:nnnn #1 #2 #3 #4
4269 {
4270 \pgfrememberpicturepositiononpagetrue
4271 \pgf@relevantforpicturesizefalse
4272 \@@_qpoint:n { col - #3 }
4273 \dim_set_eq:NN \l_tmpa_dim \pgf@x
4274 \@@_qpoint:n { row - #1 }
4275 \dim_set_eq:NN \l_tmpb_dim \pgf@y
4276 \@@_qpoint:n { col - \@@_succ:n { #4 } }
4277 \dim_set_eq:NN \l_tmpc_dim \pgf@x
4278 \bool_lazy_and:nnT
4279 { \int_compare_p:nNn { #2 } > 1 }
4280 { ! \tl_if_blank_p:V \CT@drsc@ }
4281 {
4282 \group_begin:
4283 \CT@drsc@
4284 \dim_set:Nn \l_tmpd_dim
4285 { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
4286 \pgfpathrectanglecorners
4287 { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4288 { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
4289 \pgfusepathqfill
4290 \group_end:
4291 }
4292 \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }

```

```

4293 \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4294 \prg_replicate:nn { #2 - 1 }
4295 {
4296   \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
4297   \dim_sub:Nn \l_tmpb_dim \doublerulesep
4298   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4299   \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4300 }
4301 \CT@arc@
4302 \pgfsetlinewidth { 1.1 \arrayrulewidth }
4303 \pgfsetrectcap
4304 \pgfusepathqstroke
4305 }

4306 \cs_new_protected:Npn \@@_hline_i_complete:nn #1 #2
4307 { \@@_hline_ii:nnnn { #1 } { #2 } 1 { \int_use:N \c@jCol } }

```

The command `\@@_draw_hlines:` draws all the horizontal rules excepted in the blocks (even the virtual drawn determined by commands such as `\Cdots` and in the corners (if the key `corners` is used).

```

4308 \cs_new_protected:Npn \@@_draw_hlines:
4309 {
4310   \int_step_inline:nnn
4311   { \bool_if:NTF \g_@@_NiceArray_bool 1 2 }
4312   { \bool_if:NTF \g_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
4313   {
4314     \tl_if_eq:NnF \l_@@_hlines_clist { all }
4315     { \clist_if_in:NnT \l_@@_hlines_clist { ##1 } }
4316     { \@@_hline:nn { ##1 } 1 }
4317   }
4318 }

```

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

```

4319 \cs_set:Npn \@@_Hline: { \noalign { \ifnum 0 = ` } \fi \@@_Hline_i:n { 1 } }

```

The argument of the command `\@@_Hline_i:n` is the number of successive `\Hline` found.

```

4320 \cs_set:Npn \@@_Hline_i:n #1
4321 {
4322   \peek_meaning_ignore_spaces:NTF \Hline
4323   { \@@_Hline_ii:nn { #1 + 1 } }
4324   { \@@_Hline_iii:n { #1 } }
4325 }

4326 \cs_set:Npn \@@_Hline_ii:nn #1 #2 { \@@_Hline_i:n { #1 } }

4327 \cs_set:Npn \@@_Hline_iii:n #1
4328 {
4329   \skip_vertical:n
4330   {
4331     \arrayrulewidth * ( #1 )
4332     + \doublerulesep * ( \int_max:nn 0 { #1 - 1 } )
4333   }
4334   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4335   { \@@_hline:nn { \@@_succ:n { \c@iRow } } { #1 } }
4336   \ifnum 0 = `{ \fi }
4337 }

```

## The key hvlines

The following command tests whether the current position in the array (given by `\l_tmpa_tl` for the row and `\l_tmpb_tl` for the column) would provide an horizontal rule towards the right in the block delimited by the four arguments #1, #2, #3 and #4. If this rule would be in the block (it must not be drawn), the boolean `\l_tmpa_bool` is set to false.

```
4338 \cs_new_protected:Npn \@@_test_hline_in_block:nnnn #1 #2 #3 #4
4339 {
4340   \bool_lazy_all:nT
4341   {
4342     { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
4343     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4344     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4345     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4346   }
4347   { \bool_gset_false:N \g_tmpa_bool }
4348 }
```

The same for vertical rules.

```
4349 \cs_new_protected:Npn \@@_test_vline_in_block:nnnn #1 #2 #3 #4
4350 {
4351   \bool_lazy_all:nT
4352   {
4353     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4354     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4355     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
4356     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4357   }
4358   { \bool_gset_false:N \g_tmpa_bool }
4359 }

4360 \cs_new_protected:Npn \@@_test_hline_in_stroken_block:nnnn #1 #2 #3 #4
4361 {
4362   \bool_lazy_all:nT
4363   {
4364     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4365     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 2 } }
4366     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4367     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4368   }
4369   { \bool_gset_false:N \g_tmpa_bool }
4370 }

4371 \cs_new_protected:Npn \@@_test_vline_in_stroken_block:nnnn #1 #2 #3 #4
4372 {
4373   \bool_lazy_all:nT
4374   {
4375     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4376     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4377     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4378     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 2 } }
4379   }
4380   { \bool_gset_false:N \g_tmpa_bool }
4381 }
```

## The key corners

When the key `corners` is raised, the rules are not drawn in the corners. Of course, we have to compute the corners before we begin to draw the rules.

```
4382 \cs_new_protected:Npn \@@_compute_corners:
4383 {
```

The sequence `\l_@@_corners_cells_seq` will be the sequence of all the empty cells (and not in a block) considered in the corners of the array.

```

4384 \seq_clear_new:N \l_@@_corners_cells_seq
4385 \clist_map_inline:Nn \l_@@_corners_clist
4386 {
4387   \str_case:nnF { ##1 }
4388   {
4389     { NW }
4390     { \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol }
4391     { NE }
4392     { \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1 }
4393     { SW }
4394     { \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol }
4395     { SE }
4396     { \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1 }
4397   }
4398   { \@@_error:nn { bad~corner } { ##1 } }
4399 }

```

Even if the user has used the key `corners` (or the key `hvlines-except-corners`), the list of cells in the corners may be empty.

```

4400 \seq_if_empty:NF \l_@@_corners_cells_seq
4401 {

```

You write on the `aux` file the list of the cells which are in the (empty) corners because you need that information in the `\CodeBefore` since the commands which color the `rows`, `columns` and `cells` must not color the cells in the corners.

```

4402 \iow_now:Nn \@mainaux \ExplSyntaxOn
4403 \iow_now:Nx \@mainaux
4404 {
4405   \seq_gset_from_clist:cn
4406   { c_@@_corners_cells_ \int_use:N \g_@@_env_int _ seq }
4407   { \seq_use:Nnnn \l_@@_corners_cells_seq , , , }
4408 }
4409 \iow_now:Nn \@mainaux \ExplSyntaxOff
4410 }
4411 }

```

“Computing a corner” is determining all the empty cells (which are not in a block) that belong to that corner. These cells will be added to the sequence `\l_@@_corners_cells_seq`.

The six arguments of `\@@_compute_a_corner:nnnnnn` are as follow:

- **#1** and **#2** are the number of row and column of the cell which is actually in the corner;
- **#3** and **#4** are the steps in rows and the step in columns when moving from the corner;
- **#5** is the number of the final row when scanning the rows from the corner;
- **#6** is the number of the final column when scanning the columns from the corner.

```

4412 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
4413 {

```

For the explanations and the name of the variables, we consider that we are computing the left-upper corner.

First, we try to determine which is the last empty cell (and not in a block: we won’t add that precision any longer) in the column of number 1. The flag `\l_tmpa_bool` will be raised when a non-empty cell is found.

```

4414 \bool_set_false:N \l_tmpa_bool
4415 \int_zero_new:N \l_@@_last_empty_row_int
4416 \int_set:Nn \l_@@_last_empty_row_int { #1 }
4417 \int_step_inline:nnnn { #1 } { #3 } { #5 }
4418 {
4419   \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }

```



```

4420 \bool_lazy_or:nnTF
4421 {
4422   \cs_if_exist_p:c
4423   { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
4424 }
4425 \l_tmpb_bool
4426 { \bool_set_true:N \l_tmpa_bool }
4427 {
4428   \bool_if:NF \l_tmpa_bool
4429   { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
4430 }
4431 }

```

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

```

4432 \bool_set_false:N \l_tmpa_bool
4433 \int_zero_new:N \l_@@_last_empty_column_int
4434 \int_set:Nn \l_@@_last_empty_column_int { #2 }
4435 \int_step_inline:nnnn { #2 } { #4 } { #6 }
4436 {
4437   \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
4438   \bool_lazy_or:nnTF
4439   \l_tmpb_bool
4440   {
4441     \cs_if_exist_p:c
4442     { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
4443   }
4444   { \bool_set_true:N \l_tmpa_bool }
4445   {
4446     \bool_if:NF \l_tmpa_bool
4447     { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
4448   }
4449 }

```

Now, we loop over the rows.

```

4450 \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
4451 {

```

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

```

4452 \bool_set_false:N \l_tmpa_bool
4453 \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
4454 {
4455   \@@_test_if_cell_in_a_block:nn { ##1 } { ####1 }
4456   \bool_lazy_or:nnTF
4457   \l_tmpb_bool
4458   {
4459     \cs_if_exist_p:c
4460     { pgf @ sh @ ns @ \@@_env: - ##1 - ####1 }
4461   }
4462   { \bool_set_true:N \l_tmpa_bool }
4463   {
4464     \bool_if:NF \l_tmpa_bool
4465     {
4466       \int_set:Nn \l_@@_last_empty_column_int { ####1 }
4467       \seq_put_right:Nn
4468       \l_@@_corners_cells_seq
4469       { ##1 - ####1 }
4470     }
4471   }
4472 }
4473 }
4474 }

```

The following macro tests whether a cell is in (at least) one of the blocks of the array (or in a cell with a `\diagbox`).

The flag `\l_tmpb_bool` will be raised if the cell #1-#2 is in a block (or in a cell with a `\diagbox`).

```

4475 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
4476 {
4477   \int_set:Nn \l_tmpa_int { #1 }
4478   \int_set:Nn \l_tmpb_int { #2 }
4479   \bool_set_false:N \l_tmpb_bool
4480   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4481     { \@@_test_if_cell_in_block:nnnnnnn \l_tmpa_int \l_tmpb_int ##1 }
4482 }
4483 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnnn #1 #2 #3 #4 #5 #6
4484 {
4485   \int_compare:nNnT { #3 } < { \@@_succ:n { #1 } }
4486   {
4487     \int_compare:nNnT { #1 } < { \@@_succ:n { #5 } }
4488     {
4489       \int_compare:nNnT { #4 } < { \@@_succ:n { #2 } }
4490       {
4491         \int_compare:nNnT { #2 } < { \@@_succ:n { #6 } }
4492         { \bool_set_true:N \l_tmpb_bool }
4493       }
4494     }
4495   }
4496 }

```

## 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`.

```

4497 \cs_new:Npn \@@_hdottedline:
4498 {
4499   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
4500   \@@_hdottedline_i:
4501 }

```

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

```

4502 \cs_new_protected:Npn \@@_hdottedline_i:
4503 {

```

We write in the code-after the instruction that will actually draw the dotted line. It's not possible to draw this dotted line now because we don't know the length of the line (we don't even know the number of columns).

```

4504   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4505     { \@@_hdottedline:n { \int_use:N \c@iRow } }
4506 }

```

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

```

4507 \AtBeginDocument
4508 {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”. That's why we construct now a version of `\@@_hdottedline:n` with the right environment (`\begin{pgfpicture}\end{pgfpicture}` or `\begin{tikzpicture}...\end{tikzpicture}`).

```

4509   \cs_new_protected:Npx \@@_hdottedline:n #1
4510   {
4511     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool

```

```

4512     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
4513     \c_@@_pgfortikzpicture_tl
4514     \@@_hdottedline_i:n { #1 }
4515     \c_@@_endpgfortikzpicture_tl
4516   }
4517 }

```

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

```

4518 \cs_new_protected:Npn \@@_hdottedline_i:n #1
4519 {
4520   \pgfrememberpicturepositiononpagetrue
4521   \@@_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).

```

4522   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
4523   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
4524   \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

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

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

```
\begin{bNiceMatrix}
```

```
1 & 2 & 3 & 4 \\\
```

```
\hline
```

```
1 & 2 & 3 & 4 \\\
```

```
\hdottedline
```

```
1 & 2 & 3 & 4
```

```
\end{bNiceMatrix}
```

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

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

```
\begin{bNiceMatrix}[margin]
```

```
1 & 2 & 3 & 4 \\\
```

```
\hline
```

```
1 & 2 & 3 & 4 \\\
```

```
\hdottedline
```

```
1 & 2 & 3 & 4
```

```
\end{bNiceMatrix}
```

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

```

4525   \@@_qpoint:n { col - 1 }
4526   \dim_set:Nn \l_@@_x_initial_dim
4527   {
4528     \pgf@x +

```

We do a reduction by \arraycolsep for the environments with delimiters (and not for the other).

```

4529     \bool_if:NTF \g_@@_NiceArray_bool \c_zero_dim \arraycolsep
4530     - \l_@@_left_margin_dim
4531   }
4532   \@@_qpoint:n { col - \@@_succ:n \c@jCol }
4533   \dim_set:Nn \l_@@_x_final_dim
4534   {
4535     \pgf@x -
4536     \bool_if:NTF \g_@@_NiceArray_bool \c_zero_dim \arraycolsep
4537     + \l_@@_right_margin_dim
4538   }

```

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.

```

4539   \tl_if_eq:NnF \g_@@_left_delim_tl (
4540     { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
4541   \tl_if_eq:NnF \g_@@_right_delim_tl )
4542   { \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`.

```

4543 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4544 \@@_draw_line:
4545 }

```

## Vertical dotted lines

```

4546 \cs_new_protected:Npn \@@_vdottedline:n #1
4547 {
4548   \bool_set_true:N \l_@@_initial_open_bool
4549   \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”.

```

4550   \bool_if:NTF \c_@@_tikz_loaded_bool
4551   {
4552     \tikzpicture
4553     \@@_vdottedline_i:n { #1 }
4554     \endtikzpicture
4555   }
4556   {
4557     \pgfpicture
4558     \@@_vdottedline_i:n { #1 }
4559     \endpgfpicture
4560   }
4561 }

```

```

4562 \cs_new_protected:Npn \@@_vdottedline_i:n #1
4563 {

```

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.

```

4564   \CT@arc@
4565   \pgfrememberpicturepositiononpagetrue
4566   \@@_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).

```

4567   \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
4568   \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
4569   \@@_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.

```

4570   \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
4571   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
4572   \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`.

```

4573   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4574   \@@_draw_line:
4575 }

```

## 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.

```
4576 \bool_new:N \l_@@_block_auto_columns_width_bool
```

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

```
4577 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
4578 {
4579   auto-columns-width .code:n =
4580   {
4581     \bool_set_true:N \l_@@_block_auto_columns_width_bool
4582     \dim_gzero_new:N \g_@@_max_cell_width_dim
4583     \bool_set_true:N \l_@@_auto_columns_width_bool
4584   }
4585 }

4586 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
4587 {
4588   \int_gincr:N \g_@@_NiceMatrixBlock_int
4589   \dim_zero:N \l_@@_columns_width_dim
4590   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
4591   \bool_if:NT \l_@@_block_auto_columns_width_bool
4592   {
4593     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4594     {
4595       \exp_args:NNc \dim_set:Nn \l_@@_columns_width_dim
4596       { @@_max_cell_width _ \int_use:N \g_@@_NiceMatrixBlock_int }
4597     }
4598   }
4599 }
```

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).

```
4600 {
4601   \bool_if:NT \l_@@_block_auto_columns_width_bool
4602   {
4603     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
4604     \iow_shipout:Nx \@mainaux
4605     {
4606       \cs_gset:cpn
4607       { @@ _ max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }
```

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

```
4608       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
4609     }
4610     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
4611   }
4612 }
```

## The extra nodes

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

```
4613 \cs_generate_variant:Nn \dim_min:nn { v n }
4614 \cs_generate_variant:Nn \dim_max:nn { v n }
```

The following command is called in `\@@_use_arraybox_with_notes_c`: just before the construction of the blocks (if the creation of medium nodes is required, medium nodes are also created for the blocks dans that construction uses the standard medium nodes).

```

4615 \cs_new_protected:Npn \@@_create_extra_nodes:
4616 {
4617   \bool_if:nTF \l_@@_medium_nodes_bool
4618   {
4619     \bool_if:nTF \l_@@_large_nodes_bool
4620     \@@_create_medium_and_large_nodes:
4621     \@@_create_medium_nodes:
4622   }
4623   { \bool_if:nT \l_@@_large_nodes_bool \@@_create_large_nodes: }
4624 }

```

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`.

```

4625 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
4626 {
4627   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4628   {
4629     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
4630     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
4631     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
4632     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
4633   }
4634   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4635   {
4636     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
4637     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
4638     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
4639     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
4640   }

```

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

```

4641   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4642   {
4643     \int_step_variable:nnNn
4644     \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.

```

4645     {
4646       \cs_if_exist:cT
4647       { 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`.

```

4648         {
4649             \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
4650             \dim_set:cn { l_@@_row_\@@_i: _min_dim}
4651             { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
4652             \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4653             {
4654                 \dim_set:cn { l_@@_column _ \@@_j: _min_dim}
4655                 { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
4656             }

```

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`.

```

4657             \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
4658             \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
4659             { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
4660             \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4661             {
4662                 \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
4663                 { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
4664             }
4665         }
4666     }
4667 }

```

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

```

4668     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4669     {
4670         \dim_compare:nNnT
4671         { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
4672         {
4673             \@@_qpoint:n { row - \@@_i: - base }
4674             \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
4675             \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
4676         }
4677     }
4678     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4679     {
4680         \dim_compare:nNnT
4681         { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
4682         {
4683             \@@_qpoint:n { col - \@@_j: }
4684             \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
4685             \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
4686         }
4687     }
4688 }

```

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

```

4689 \cs_new_protected:Npn \@@_create_medium_nodes:
4690 {
4691     \pgfpicture
4692     \pgfrememberpicturepositiononpagetrue
4693     \pgf@relevantforpicturesizefalse
4694     \@@_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”.

```

4695     \tl_set:Nn \l_@@_suffix_tl { -medium }
4696     \@@_create_nodes:

```

```

4697 \endpgfpicture
4698 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones<sup>61</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:`.

```

4699 \cs_new_protected:Npn \@@_create_large_nodes:
4700 {
4701   \pgfpicture
4702   \pgfrememberpicturepositiononpagetrue
4703   \pgf@relevantforpicturesizefalse
4704   \@@_computations_for_medium_nodes:
4705   \@@_computations_for_large_nodes:
4706   \tl_set:Nn \l_@@_suffix_tl { - large }
4707   \@@_create_nodes:
4708   \endpgfpicture
4709 }
4710 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
4711 {
4712   \pgfpicture
4713   \pgfrememberpicturepositiononpagetrue
4714   \pgf@relevantforpicturesizefalse
4715   \@@_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”.

```

4716   \tl_set:Nn \l_@@_suffix_tl { - medium }
4717   \@@_create_nodes:
4718   \@@_computations_for_large_nodes:
4719   \tl_set:Nn \l_@@_suffix_tl { - large }
4720   \@@_create_nodes:
4721   \endpgfpicture
4722 }

```

For “large nodes”, the exterior rows and columns don’t interfer. 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.

```

4723 \cs_new_protected:Npn \@@_computations_for_large_nodes:
4724 {
4725   \int_set:Nn \l_@@_first_row_int 1
4726   \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`.

```

4727   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
4728   {
4729     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
4730     {
4731       (
4732         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
4733         \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4734       )
4735       / 2
4736     }
4737     \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4738     { l_@@_row_\@@_i: _ min_dim }
4739   }
4740   \int_step_variable:nNn { \c@jCol - 1 } \@@_j:

```

---

<sup>61</sup>If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`



```

4741 {
4742   \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
4743   {
4744     (
4745       \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
4746       \dim_use:c
4747         { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4748     )
4749     / 2
4750   }
4751   \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4752   { l_@@_column _ \@@_j: _ max _ dim }
4753 }

```

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

```

4754 \dim_sub:cn
4755   { l_@@_column _ 1 _ min _ dim }
4756   \l_@@_left_margin_dim
4757 \dim_add:cn
4758   { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
4759   \l_@@_right_margin_dim
4760 }

```

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`).

```

4761 \cs_new_protected:Npn \@@_create_nodes:
4762 {
4763   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4764   {
4765     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4766     {

```

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

```

4767       \@@_pgf_rect_node:nnnnn
4768       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4769       { \dim_use:c { l_@@_column _ \@@_j: _ min_dim } }
4770       { \dim_use:c { l_@@_row _ \@@_i: _ min_dim } }
4771       { \dim_use:c { l_@@_column _ \@@_j: _ max_dim } }
4772       { \dim_use:c { l_@@_row _ \@@_i: _ max_dim } }
4773       \str_if_empty:NF \l_@@_name_str
4774       {
4775         \pgfnodealias
4776         { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4777         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4778       }
4779     }
4780   }

```

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  $n$ .

```

4781 \seq_mapthread_function:NNN
4782   \g_@@_multicolumn_cells_seq
4783   \g_@@_multicolumn_sizes_seq
4784   \@@_node_for_multicolumn:nn
4785 }

```

```

4786 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
4787 {

```

```

4788 \cs_set_nopar:Npn \@@_i: { #1 }
4789 \cs_set_nopar:Npn \@@_j: { #2 }
4790 }

```

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”).

```

4791 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
4792 {
4793   \@@_extract_coords_values: #1 \q_stop
4794   \@@_pgf_rect_node:nnnnn
4795     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4796     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }
4797     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } }
4798     { \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
4799     { \dim_use:c { l_@@_row _ \@@_i: _ max _ dim } }
4800   \str_if_empty:NF \l_@@_name_str
4801   {
4802     \pgfnodealias
4803       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4804       { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl}
4805   }
4806 }

```

## The blocks

The code deals with the command `\Block`. This command has no direct link with the environment `{NiceMatrixBlock}`.

The options of the command `\Block` will be analyzed first in the cell of the array (and once again when the block will be put in the array). Here is the set of keys for the first pass.

```

4807 \keys_define:nn { NiceMatrix / Block / FirstPass }
4808 {
4809   l .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl l ,
4810   l .value_forbidden:n = true ,
4811   r .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl r ,
4812   r .value_forbidden:n = true ,
4813   c .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl c ,
4814   c .value_forbidden:n = true ,
4815   t .code:n = \tl_set:Nn \l_@@_vpos_of_block_tl t ,
4816   t .value_forbidden:n = true ,
4817   b .code:n = \tl_set:Nn \l_@@_vpos_of_block_tl b ,
4818   b .value_forbidden:n = true ,
4819   color .tl_set:N = \l_@@_color_tl ,
4820   color .value_required:n = true ,
4821 }

```

The following command `\@@_Block:` will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewExpandableDocumentCommand` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label and before the beginning of the small array of the block). It’s mandatory to use an expandable command.

```

4822 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
4823 {

```

If the first mandatory argument of the command (which is the size of the block with the syntax  $i$ - $j$ ) has not been provided by the user, you use 1-1 (that is to say a block of only one cell).

```

4824   \peek_remove_spaces:n
4825   {
4826     \tl_if_blank:nTF { #2 }
4827     { \@@_Block_i 1-1 \q_stop }
4828     { \@@_Block_i #2 \q_stop }

```

```

4829     { #1 } { #3 } { #4 }
4830   }
4831 }

```

With the following construction, we extract the values of  $i$  and  $j$  in the first mandatory argument of the command.

```

4832 \cs_new:Npn \@@_Block_i #1-#2 \q_stop { \@@_Block_ii:nnnnn { #1 } { #2 } }

```

Now, the arguments have been extracted:  $\#1$  is  $i$  (the number of rows of the block),  $\#2$  is  $j$  (the number of columns of the block),  $\#3$  is the list of key-values,  $\#4$  are the tokens to put before the math mode and the beginning of the small array of the block and  $\#5$  is the label of the block.

```

4833 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
4834 {

```

We recall that  $\#1$  and  $\#2$  have been extracted from the first mandatory argument of `\Block` (which is of the syntax  $i$ - $j$ ). However, the user is allowed to omit  $i$  or  $j$  (or both). We detect that situation by replacing a missing value by 100 (it's a convention: when the block will actually be drawn these values will be detected and interpreted as *maximal possible value* according to the actual size of the array).

```

4835   \bool_lazy_or:nnTF
4836     { \tl_if_blank_p:n { #1 } }
4837     { \str_if_eq_p:nn { #1 } { * } }
4838     { \int_set:Nn \l_tmpa_int { 100 } }
4839     { \int_set:Nn \l_tmpa_int { #1 } }
4840   \bool_lazy_or:nnTF
4841     { \tl_if_blank_p:n { #2 } }
4842     { \str_if_eq_p:nn { #2 } { * } }
4843     { \int_set:Nn \l_tmpb_int { 100 } }
4844     { \int_set:Nn \l_tmpb_int { #2 } }

```

If the block is mono-column.

```

4845   \int_compare:nNnTF \l_tmpb_int = 1
4846   {
4847     \tl_if_empty:NTF \l_@@_cell_type_tl
4848       { \tl_set:Nn \l_@@_hpos_of_block_tl c }
4849       { \tl_set_eq:NN \l_@@_hpos_of_block_tl \l_@@_cell_type_tl }
4850   }
4851   { \tl_set:Nn \l_@@_hpos_of_block_tl c }

```

The value of `\l_@@_hpos_of_block_tl` may be modified by the keys of the command `\Block` that we will analyze now.

```

4852   \keys_set:known:nn { NiceMatrix / Block / FirstPass } { #3 }
4853   \tl_set:Nx \l_tmpa_tl
4854   {
4855     { \int_use:N \c@iRow }
4856     { \int_use:N \c@jCol }
4857     { \int_eval:n { \c@iRow + \l_tmpa_int - 1 } }
4858     { \int_eval:n { \c@jCol + \l_tmpb_int - 1 } }
4859   }

```

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

$\{imin\}\{jmin\}\{imax\}\{jmax\}$ .

If the block is mono-column or mono-row, we have a special treatment. That's why we have two macros: `\@@_Block_iv:nnnnn` and `\@@_Block_v:nnnnn` (the five arguments of those macros are provided by currying).

```

4860   \bool_lazy_or:nnTF
4861     { \int_compare_p:nNn { \l_tmpa_int } = 1 }
4862     { \int_compare_p:nNn { \l_tmpb_int } = 1 }
4863     { \exp_args:Nxx \@@_Block_iv:nnnnn }
4864     { \exp_args:Nxx \@@_Block_v:nnnnn }

```

```

4865 { \l_tmpa_int } { \l_tmpb_int } { #3 } { #4 } { #5 }
4866 }

```

The following macro is for the case of a `\Block` which is mono-row or mono-column (or both). In that case, the content of the block is composed right now in a box (because we have to take into account the dimensions of that box for the width of the current column or the height and the depth of the current row). However, that box will be put in the array *after the construction of the array* (by using PGF).

```

4867 \cs_new_protected:Npn \@@_Block_iv:nnnnn #1 #2 #3 #4 #5
4868 {
4869   \int_gincr:N \g_@@_block_box_int
4870   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4871   {
4872     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4873     {
4874       \@@_actually_diagbox:nnnnnn
4875       { \int_use:N \c@iRow }
4876       { \int_use:N \c@jCol }
4877       { \int_eval:n { \c@iRow + #1 - 1 } }
4878       { \int_eval:n { \c@jCol + #2 - 1 } }
4879       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4880     }
4881   }
4882   \box_gclear_new:c
4883   { g_@@_block_box_int \int_use:N \g_@@_block_box_int _ box }
4884   \hbox_gset:cn
4885   { g_@@_block_box_int \int_use:N \g_@@_block_box_int _ box }
4886   {

```

For a mono-column block, if the user has specified a color for the column in the preamble of the array, we want to fix that color in the box we construct. We do that with `\set@color` and not `\color_ensure_current:` because that command seems to be bugged: it doesn't work in XeLaTeX when `fontspec` is loaded.

```

4887   \tl_if_empty:NTF \l_@@_color_tl
4888   { \int_compare:nNnT { #2 } = 1 \set@color }
4889   { \color { \l_@@_color_tl } }
4890   \group_begin:
4891   \cs_set:Npn \arraystretch { 1 }
4892   \dim_set_eq:NN \extrarowheight \c_zero_dim
4893   #4

```

If the box is rotated (the key `\rotate` may be in the previous #4), the tabular used for the content of the cell will be constructed with a format `c`. In the other cases, the tabular will be constructed with a format equal to the key of position of the box. In other words: the alignment internal to the tabular is the same as the external alignment of the tabular (that is to say the position of the block in its zone of merged cells).

```

4894   \bool_if:NT \g_@@_rotate_bool { \tl_set:Nn \l_@@_hpos_of_block_tl c }
4895   \bool_if:NTF \l_@@_NiceTabular_bool
4896   {
4897     \use:x
4898     {
4899       \exp_not:N \begin { tabular } [ \l_@@_vpos_of_block_tl ]
4900       { @ { } \l_@@_hpos_of_block_tl @ { } }
4901     }
4902     #5
4903     \end { tabular }
4904   }
4905   {
4906     \c_math_toggle_token
4907     \use:x
4908     {
4909       \exp_not:N \begin { array } [ \l_@@_vpos_of_block_tl ]
4910       { @ { } \l_@@_hpos_of_block_tl @ { } }

```

```

4911         }
4912         #5
4913         \end { array }
4914         \c_math_toggle_token
4915     }
4916     \group_end:
4917 }
4918 \bool_if:NT \g_@@_rotate_bool
4919 {
4920     \box_grotate:cn
4921     { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4922     { 90 }
4923     \bool_gset_false:N \g_@@_rotate_bool
4924 }

```

If we are in a mono-column block, we take into account the width of that block for the width of the column.

```

4925 \int_compare:nNnT { #2 } = 1
4926 {
4927     \dim_gset:Nn \g_@@_blocks_wd_dim
4928     {
4929         \dim_max:nn
4930         \g_@@_blocks_wd_dim
4931         {
4932             \box_wd:c
4933             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4934         }
4935     }
4936 }

```

If we are in a mono-row block, we take into account the height and the depth of that block for the height and the depth of the row.

```

4937 \int_compare:nNnT { #1 } = 1
4938 {
4939     \dim_gset:Nn \g_@@_blocks_ht_dim
4940     {
4941         \dim_max:nn
4942         \g_@@_blocks_ht_dim
4943         {
4944             \box_ht:c
4945             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4946         }
4947     }
4948     \dim_gset:Nn \g_@@_blocks_dp_dim
4949     {
4950         \dim_max:nn
4951         \g_@@_blocks_dp_dim
4952         {
4953             \box_dp:c
4954             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4955         }
4956     }
4957 }
4958 \seq_gput_right:Nx \g_@@_blocks_seq
4959 {
4960     \l_tmpa_tl

```

In the list of options #3, maybe there is a key for the horizontal alignment (l, r or c). In that case, that key has been read and stored in \l\_@@\_hpos\_of\_block\_tl. However, maybe there were no key of the horizontal alignment and that's why we put a key corresponding to the value of \l\_@@\_hpos\_of\_block\_tl, which is fixed by the type of current column.

```

4961     { \exp_not:n { #3 } , \l_@@_hpos_of_block_tl }
4962     {
4963         \box_use_drop:c

```

```

4964         { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4965     }
4966 }
4967 }

```

The following macro is for the standard case, where the block is not mono-row and not mono-column. In that case, the content of the block is *not* composed right now in a box. The composition in a box will be done further, just after the construction of the array.

```

4968 \cs_new_protected:Npn \@@_Block_v:nnnnn #1 #2 #3 #4 #5
4969 {
4970     \seq_gput_right:Nx \g_@@_blocks_seq
4971     {
4972         \l_tmpa_tl
4973         { \exp_not:n { #3 } }
4974         \exp_not:n
4975         {
4976             {
4977                 \bool_if:NTF \l_@@_NiceTabular_bool
4978                 {
4979                     \group_begin:
4980                     \cs_set:Npn \arraystretch { 1 }
4981                     \dim_set_eq:NN \extrarowheight \c_zero_dim
4982                     #4

```

If the box is rotated (the key `\rotate` may be in the previous #4), the tabular used for the content of the cell will be constructed with a format `c`. In the other cases, the tabular will be constructed with a format equal to the key of position of the box. In other words: the alignment internal to the tabular is the same as the external alignment of the tabular (that is to say the position of the block in its zone of merged cells).

```

4983         \bool_if:NT \g_@@_rotate_bool
4984         { \tl_set:Nn \l_@@_hpos_of_block_tl c }
4985         \use:x
4986         {
4987             \exp_not:N \begin { tabular } [ \l_@@_vpos_of_block_tl ]
4988             { @ { } \l_@@_hpos_of_block_tl @ { } }
4989         }
4990         #5
4991         \end { tabular }
4992         \group_end:
4993     }
4994     {
4995         \group_begin:
4996         \cs_set:Npn \arraystretch { 1 }
4997         \dim_set_eq:NN \extrarowheight \c_zero_dim
4998         #4
4999         \bool_if:NT \g_@@_rotate_bool
5000         { \tl_set:Nn \l_@@_hpos_of_block_tl c }
5001         \c_math_toggle_token
5002         \use:x
5003         {
5004             \exp_not:N \begin { array } [ \l_@@_vpos_of_block_tl ]
5005             { @ { } \l_@@_hpos_of_block_tl @ { } }
5006         }
5007         #5
5008         \end { array }
5009         \c_math_toggle_token
5010         \group_end:
5011     }
5012 }
5013 }
5014 }
5015 }

```

We recall that the options of the command `\Block` are analyzed twice: first in the cell of the array and once again when the block will be put in the array *after the construction of the array* (by using PGF).

```

5016 \keys_define:nn { NiceMatrix / Block / SecondPass }
5017 {
5018   fill .tl_set:N = \l_@@_fill_tl ,
5019   fill .value_required:n = true ,
5020   draw .tl_set:N = \l_@@_draw_tl ,
5021   draw .default:n = default ,
5022   rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
5023   rounded-corners .default:n = 4 pt ,
5024   color .code:n = \color { #1 } \tl_set:Nn \l_@@_draw_tl { #1 } ,
5025   color .value_required:n = true ,
5026   borders .clist_set:N = \l_@@_borders_clist ,
5027   borders .value_required:n = true ,
5028   hvlines .bool_set:N = \l_@@_hvlines_block_bool ,
5029   hvlines .default:n = true ,
5030   line-width .dim_set:N = \l_@@_line_width_dim ,
5031   line-width .value_required:n = true ,
5032   l .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl l ,
5033   l .value_forbidden:n = true ,
5034   r .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl r ,
5035   r .value_forbidden:n = true ,
5036   c .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl c ,
5037   c .value_forbidden:n = true ,
5038   t .code:n = \tl_set:Nn \l_@@_vpos_of_block_tl t ,
5039   t .value_forbidden:n = true ,
5040   b .code:n = \tl_set:Nn \l_@@_vpos_of_block_tl b ,
5041   b .value_forbidden:n = true ,
5042   unknown .code:n = @@_error:n { Unknown-key-for-Block }
5043 }

```

The command `\@@_draw_blocks:` will draw all the blocks. This command is used after the construction of the array. We have to revert to a clean version of `\ialign` because there may be tabulars in the `\Block` instructions that will be composed now.

```

5044 \cs_new_protected:Npn \@@_draw_blocks:
5045 {
5046   \cs_set_eq:NN \ialign \@@_old_ialign:
5047   \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iv:nnnnnn ##1 }
5048 }
5049 \cs_new_protected:Npn \@@_Block_iv:nnnnnn #1 #2 #3 #4 #5 #6
5050 {

```

The integer `\l_@@_last_row_int` will be the last row of the block and `\l_@@_last_col_int` its last column.

```

5051   \int_zero_new:N \l_@@_last_row_int
5052   \int_zero_new:N \l_@@_last_col_int

```

We remind that the first mandatory argument of the command `\Block` is the size of the block with the special format  $i$ - $j$ . However, the user is allowed to omit  $i$  or  $j$  (or both). This will be interpreted as: the last row (resp. column) of the block will be the last row (resp. column) of the block (without the potential exterior row—resp. column—of the array). By convention, this is stored in `\g_@@_blocks_seq` as a number of rows (resp. columns) for the block equal to 100. That's what we detect now.

```

5053   \int_compare:nNnTF { #3 } > { 99 }
5054     { \int_set_eq:NN \l_@@_last_row_int \c@iRow }
5055     { \int_set:Nn \l_@@_last_row_int { #3 } }
5056   \int_compare:nNnTF { #4 } > { 99 }
5057     { \int_set_eq:NN \l_@@_last_col_int \c@jCol }
5058     { \int_set:Nn \l_@@_last_col_int { #4 } }
5059   \int_compare:nNnTF \l_@@_last_col_int > \g_@@_col_total_int

```

```

5060 {
5061   \int_compare:nTF
5062     { \l_@@_last_col_int <= \g_@@_static_num_of_col_int }
5063     {
5064       \msg_error:nnnn { nicematrix } { Block-too~large~2 } { #1 } { #2 }
5065       \@@_msg_redirect_name:nn { Block-too~large~2 } { none }
5066       \group_begin:
5067       \globaldefs = 1
5068       \@@_msg_redirect_name:nn { columns~not~used } { none }
5069       \group_end:
5070     }
5071     { \msg_error:nnnn { nicematrix } { Block-too~large~1 } { #1 } { #2 } }
5072   }
5073   {
5074     \int_compare:nNnTF \l_@@_last_row_int > \g_@@_row_total_int
5075     { \msg_error:nnnn { nicematrix } { Block-too~large~1 } { #1 } { #2 } }
5076     { \@@_Block_v:nnnnnn { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } }
5077   }
5078 }
5079 \cs_new_protected:Npn \@@_Block_v:nnnnnn #1 #2 #3 #4 #5 #6
5080 {

```

The sequence of the positions of the blocks will be used when drawing the rules (in fact, there is also the `\multicolumn` and the `\diagbox` in that sequence).

```

5081   \seq_gput_left:Nn \g_@@_pos_of_blocks_seq { { #1 } { #2 } { #3 } { #4 } }

```

The group is for the keys.

```

5082   \group_begin:
5083   \keys_set:nn { NiceMatrix / Block / SecondPass } { #5 }
5084   \tl_if_empty:NF \l_@@_draw_tl
5085   {
5086     \tl_gput_right:Nx \g_nicematrix_code_after_tl
5087     {
5088       \@@_stroke_block:nnn
5089       { \exp_not:n { #5 } }
5090       { #1 - #2 }
5091       { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5092     }
5093     \seq_gput_right:Nn \g_@@_pos_of_stroken_blocks_seq
5094     { { #1 } { #2 } { #3 } { #4 } }
5095   }
5096   \bool_if:NT \l_@@_hvlines_block_bool
5097   {
5098     \tl_gput_right:Nx \g_nicematrix_code_after_tl
5099     {
5100       \@@_hvlines_block:nnn
5101       { \exp_not:n { #5 } }
5102       { #1 - #2 }
5103       { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5104     }
5105   }
5106   \clist_if_empty:NF \l_@@_borders_clist
5107   {
5108     \tl_gput_right:Nx \g_nicematrix_code_after_tl
5109     {
5110       \@@_stroke_borders_block:nnn
5111       { \exp_not:n { #5 } }
5112       { #1 - #2 }
5113       { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5114     }
5115   }

```



```

5116 \tl_if_empty:NF \l_@@_fill_tl
5117 {

```

The command `\@@_extract_brackets` will extract the potential specification of color space at the beginning of `\l_@@_fill_tl` and store it in `\l_tmpa_tl` and store the color itself in `\l_tmpb_tl`.

```

5118 \exp_last_unbraced:N \@@_extract_brackets \l_@@_fill_tl \q_stop
5119 \tl_gput_right:Nx \g_nicematrix_code_before_tl
5120 {
5121   \exp_not:N \roundedrectanglecolor
5122   [ \l_tmpa_tl ]
5123   { \exp_not:N \l_tmpb_tl }
5124   { #1 - #2 }
5125   { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5126   { \dim_use:N \l_@@_rounded_corners_dim }
5127 }
5128 }

```

```

5129 \cs_set_protected_nopar:Npn \diagbox ##1 ##2
5130 {
5131   \tl_gput_right:Nx \g_@@_internal_code_after_tl
5132   {
5133     \@@_actually_diagbox:nnnnnn
5134     { #1 }
5135     { #2 }
5136     { \int_use:N \l_@@_last_row_int }
5137     { \int_use:N \l_@@_last_col_int }
5138     { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
5139   }
5140 }

```

```

5141 \hbox_set:Nn \l_@@_cell_box { \set@color #6 }
5142 \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:

```

Let's consider the following `{NiceTabular}`. Because of the instruction `!\hspace{1cm}` in the preamble which increases the space between the columns (by adding, in fact, that space to the previous column, that is to say the second column of the tabular), we will create *two* nodes relative to the block: the node `1-1-block` and the node `1-1-block-short`. The latter will be used by `nicematrix` to put the label of the node. The first one won't be used explicitly.

```

\begin{NiceTabular}{cc!\hspace{1cm}}c}
\Block{2-2}{our block} & & one & \\\
& & two & \\\
three & & four & five & \\\
six & & seven & eight & \\\
\end{NiceTabular}

```

We highlight the node `1-1-block`

our block		one
		two
three	four	five
six	seven	eight

We highlight the node `1-1-block-short`

our block		one
		two
three	four	five
six	seven	eight

The construction of the node corresponding to the merged cells.

```

5143 \pgfpicture
5144 \pgfrememberpicturepositiononpagetrue
5145 \pgf@relevantforpicturesizefalse
5146 \@@_qpoint:n { row - #1 }
5147 \dim_set_eq:NN \l_tmpa_dim \pgf@y
5148 \@@_qpoint:n { col - #2 }
5149 \dim_set_eq:NN \l_tmpb_dim \pgf@x
5150 \@@_qpoint:n { row - \@@_succ:n { \l_@@_last_row_int } }

```

```

5151 \dim_set_eq:NN \l_tmpc_dim \pgf@y
5152 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
5153 \dim_set_eq:NN \l_tmpd_dim \pgf@x

```

We construct the node for the block with the name (#1-#2-block).

The function `\@@_pgf_rect_node:nnnnn` takes in as arguments the name of the node and the four coordinates of two opposite corner points of the rectangle.

```

5154 \begin { pgfscope }
5155 \@@_pgf_rect_node:nnnnn
5156 { \@@_env: - #1 - #2 - block }
5157 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
5158 \end { pgfscope }

```

We construct the short node.

```

5159 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
5160 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5161 {

```

We recall that, when a cell is empty, no (normal) node is created in that cell. That’s why we test the existence of the node before using it.

```

5162 \cs_if_exist:cT
5163 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
5164 {
5165 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
5166 {
5167 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
5168 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
5169 }
5170 }
5171 }

```

If all the cells of the column were empty, `\l_tmpb_dim` has still the same value `\c_max_dim`. In that case, you use for `\l_tmpb_dim` the value of the position of the vertical rule.

```

5172 \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
5173 {
5174 \@@_qpoint:n { col - #2 }
5175 \dim_set_eq:NN \l_tmpb_dim \pgf@x
5176 }
5177 \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
5178 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5179 {
5180 \cs_if_exist:cT
5181 { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
5182 {
5183 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
5184 {
5185 \pgfpointanchor
5186 { \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
5187 { east }
5188 \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
5189 }
5190 }
5191 }
5192 \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
5193 {
5194 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
5195 \dim_set_eq:NN \l_tmpd_dim \pgf@x
5196 }
5197 \@@_pgf_rect_node:nnnnn
5198 { \@@_env: - #1 - #2 - block - short }
5199 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim

```

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

```

5200 \bool_if:NT \l_@@_medium_nodes_bool
5201 {
5202   \@@_pgf_rect_node:nnn
5203   { \@@_env: - #1 - #2 - block - medium }
5204   { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
5205   {
5206     \pgfpointanchor
5207     { \@@_env:
5208       - \int_use:N \l_@@_last_row_int
5209       - \int_use:N \l_@@_last_col_int - medium
5210     }
5211     { south-east }
5212   }
5213 }

```

Now, we will put the label of the block beginning with the case of a \Block of one row.

```

5214 \int_compare:nNnTF { #1 } = { #3 }
5215 {

```

We take into account the case of a block of one row in the “first row” or the “last row”.

```

5216 \int_compare:nNnTF { #1 } = 0
5217 { \l_@@_code_for_first_row_tl }
5218 {
5219   \int_compare:nNnT { #1 } = \l_@@_last_row_int
5220   \l_@@_code_for_last_row_tl
5221 }

```

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.

```

5222 \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

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

```

5223 \pgfpointanchor
5224 { \@@_env: - #1 - #2 - block - short }
5225 {
5226   \str_case:Vn \l_@@_hpos_of_block_tl
5227   {
5228     c { center }
5229     l { west }
5230     r { east }
5231   }
5232 }

```

We put the label of the block which has been composed in \l\_@@\_cell\_box.

```

5233 \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
5234 \pgfset { inner~sep = \c_zero_dim }
5235 \pgfnode
5236 { rectangle }
5237 {
5238   \str_case:Vn \l_@@_hpos_of_block_tl
5239   {
5240     c { base }
5241     l { base~west }
5242     r { base~east }
5243   }
5244 }
5245 { \box_use_drop:N \l_@@_cell_box } { } { }
5246 }

```

If the number of rows is different of 1, we will put the label of the block by using the short node (the label of the block has been composed in \l\_@@\_cell\_box).

```

5247 {

```

If we are in the first column, we must put the block as if it was with the key `r`.

```

5248 \int_compare:nNt { #2 } = 0
5249 { \tl_set:Nn \l_@@_hpos_of_block_tl r }
5250 \bool_if:nT \g_@@_last_col_found_bool
5251 {
5252   \int_compare:nNt { #2 } = \g_@@_col_total_int
5253   { \tl_set:Nn \l_@@_hpos_of_block_tl l }
5254 }
5255 \pgftransformshift
5256 {
5257   \pgfpointanchor
5258   { \@@_env: - #1 - #2 - block - short }
5259   {
5260     \str_case:Nn \l_@@_hpos_of_block_tl
5261     {
5262       c { center }
5263       l { west }
5264       r { east }
5265     }
5266   }
5267 }
5268 \pgfset { inner-sep = \c_zero_dim }
5269 \pgfnode
5270 { rectangle }
5271 {
5272   \str_case:Nn \l_@@_hpos_of_block_tl
5273   {
5274     c { center }
5275     l { west }
5276     r { east }
5277   }
5278 }
5279 { \box_use_drop:N \l_@@_cell_box } { } { }
5280 }
5281 \endpgfpicture
5282 \group_end:
5283 }

```

```

5284 \NewDocumentCommand \@@_extract_brackets { 0 { } }
5285 {
5286   \tl_set:Nn \l_tmpa_tl { #1 }
5287   \@@_store_in_tmpl_tl
5288 }
5289 \cs_new_protected:Npn \@@_store_in_tmpl_tl #1 \q_stop
5290 { \tl_set:Nn \l_tmpl_tl { #1 } }

```

The first argument of `\@@_stroke_block:nnn` is a list of options for the rectangle that you will stroke. The second argument is the upper-left cell of the block (with, as usual, the syntax  $i-j$ ) and the third is the last cell of the block (with the same syntax).

```

5291 \cs_new_protected:Npn \@@_stroke_block:nnn #1 #2 #3
5292 {
5293   \group_begin:
5294   \tl_clear:N \l_@@_draw_tl
5295   \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
5296   \keys_set_known:nn { NiceMatrix / BlockStroke } { #1 }
5297   \pgfpicture
5298   \pgfrememberpicturepositiononpagetrue
5299   \pgf@relevantforpicturesizefalse
5300   \tl_if_empty:NF \l_@@_draw_tl
5301   {

```

If the user has used the key `color` of the command `\Block` without value, the color fixed by `\arrayrulecolor` is used.

```

5302     \str_if_eq:VnTF \l_@@_draw_tl { default }
5303     { \CT@arc@ }
5304     { \exp_args:NW \pgfsetstrokecolor \l_@@_draw_tl }
5305   }
5306   \pgfsetcornersarced
5307   {
5308     \pgfpoint
5309     { \dim_use:N \l_@@_rounded_corners_dim }
5310     { \dim_use:N \l_@@_rounded_corners_dim }
5311   }
5312   \@@_cut_on_hyphen:w #2 \q_stop
5313   \bool_lazy_and:nnT
5314   { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
5315   { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
5316   {
5317     \@@_qpoint:n { row - \l_tmpa_tl }
5318     \dim_set:Nn \l_tmpb_dim { \pgf@y }
5319     \@@_qpoint:n { col - \l_tmpb_tl }
5320     \dim_set:Nn \l_tmpc_dim { \pgf@x }
5321     \@@_cut_on_hyphen:w #3 \q_stop
5322     \int_compare:nNnT \l_tmpa_tl > \c@iRow
5323     { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
5324     \int_compare:nNnT \l_tmpb_tl > \c@jCol
5325     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
5326     \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
5327     \dim_set:Nn \l_tmpa_dim { \pgf@y }
5328     \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
5329     \dim_set:Nn \l_tmpd_dim { \pgf@x }
5330     \pgfpathrectanglecorners
5331     { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
5332     { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
5333     \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }

```

We can't use `\pgfusepathqstroke` because of the key `rounded-corners`.

```

5334     \pgfusepath { stroke }
5335   }
5336   \endpgfpicture
5337   \group_end:
5338 }

```

Here is the set of keys for the command `\@@_stroke_block:nnn`.

```

5339 \keys_define:nn { NiceMatrix / BlockStroke }
5340 {
5341   color .tl_set:N = \l_@@_draw_tl ,
5342   draw .tl_set:N = \l_@@_draw_tl ,
5343   draw .default:n = default ,
5344   line-width .dim_set:N = \l_@@_line_width_dim ,
5345   rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
5346   rounded-corners .default:n = 4 pt
5347 }

```

The first argument of `\@@_hvlines_block:nnn` is a list of options for the rules that we will draw. The second argument is the upper-left cell of the block (with, as usual, the syntax  $i-j$ ) and the third is the last cell of the block (with the same syntax).

```

5348 \cs_new_protected:Npn \@@_hvlines_block:nnn #1 #2 #3
5349 {
5350   \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
5351   \keys_set_known:nn { NiceMatrix / BlockBorders } { #1 }
5352   \@@_cut_on_hyphen:w #2 \q_stop
5353   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
5354   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl

```

```

5355 \@@_cut_on_hyphen:w #3 \q_stop
5356 \tl_set:Nx \l_tmpa_tl { \int_eval:n { \l_tmpa_tl + 1 } }
5357 \tl_set:Nx \l_tmpb_tl { \int_eval:n { \l_tmpb_tl + 1 } }
5358 \pgfpicture
5359 \pgfrememberpicturepositiononpagetrue
5360 \pgf@relevantforpicturesizefalse
5361 \CT@arc@
5362 \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }

```

First, the vertical rules.

```

5363 \@@_qpoint:n { row - \l_tmpa_tl }
5364 \dim_set_eq:NN \l_tmpa_dim \pgf@y
5365 \@@_qpoint:n { row - \l_tmpc_tl }
5366 \dim_set_eq:NN \l_tmpb_dim \pgf@y
5367 \int_step_inline:nnn \l_tmpd_tl \l_tmpb_tl
5368 {
5369   \@@_qpoint:n { col - ##1 }
5370   \pgfpathmoveto { \pgfpoint \pgf@x \l_tmpa_dim }
5371   \pgfpathlineto { \pgfpoint \pgf@x \l_tmpb_dim }
5372   \pgfusepathqstroke
5373 }

```

Now, the horizontal rules.

```

5374 \@@_qpoint:n { col - \l_tmpb_tl }
5375 \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }
5376 \@@_qpoint:n { col - \l_tmpd_tl }
5377 \dim_set:Nn \l_tmpb_dim { \pgf@x - 0.5 \arrayrulewidth }
5378 \int_step_inline:nnn \l_tmpc_tl \l_tmpa_tl
5379 {
5380   \@@_qpoint:n { row - ##1 }
5381   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \pgf@y }
5382   \pgfpathlineto { \pgfpoint \l_tmpb_dim \pgf@y }
5383   \pgfusepathqstroke
5384 }
5385 \endpgfpicture
5386 }

```

The first argument of `\@@_stroke_borders_block:nnn` is a list of options for the borders that you will stroke. The second argument is the upper-left cell of the block (with, as usual, the syntax *i-j*) and the third is the last cell of the block (with the same syntax).

```

5387 \cs_new_protected:Npn \@@_stroke_borders_block:nnn #1 #2 #3
5388 {
5389   \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
5390   \keys_set_known:nn { NiceMatrix / BlockBorders } { #1 }
5391   \dim_compare:nNnTF \l_@@_rounded_corners_dim > \c_zero_dim
5392   { \@@_error:n { borders~forbidden } }
5393   {
5394     \clist_map_inline:Nn \l_@@_borders_clist
5395     {
5396       \clist_if_in:nnF { top , bottom , left , right } { ##1 }
5397       { \@@_error:nn { bad-border } { ##1 } }
5398     }
5399     \@@_cut_on_hyphen:w #2 \q_stop
5400     \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
5401     \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
5402     \@@_cut_on_hyphen:w #3 \q_stop
5403     \tl_set:Nx \l_tmpa_tl { \int_eval:n { \l_tmpa_tl + 1 } }
5404     \tl_set:Nx \l_tmpb_tl { \int_eval:n { \l_tmpb_tl + 1 } }
5405     \pgfpicture
5406     \pgfrememberpicturepositiononpagetrue
5407     \pgf@relevantforpicturesizefalse
5408     \CT@arc@
5409     \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }
5410     \clist_if_in:NnT \l_@@_borders_clist { right }

```

```

5411     { \@@_stroke_vertical:n \l_tmpb_tl }
5412     \clist_if_in:NnT \l_@@_borders_clist { left }
5413     { \@@_stroke_vertical:n \l_tmpd_tl }
5414     \clist_if_in:NnT \l_@@_borders_clist { bottom }
5415     { \@@_stroke_horizontal:n \l_tmpa_tl }
5416     \clist_if_in:NnT \l_@@_borders_clist { top }
5417     { \@@_stroke_horizontal:n \l_tmpc_tl }
5418     \endpgfpicture
5419   }
5420 }

```

The following command is used to stroke the left border and the right border. The argument #1 is the number of column (in the sense of the col node).

```

5421 \cs_new_protected:Npn \@@_stroke_vertical:n #1
5422 {
5423   \@@_qpoint:n \l_tmpc_tl
5424   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \l_@@_line_width_dim }
5425   \@@_qpoint:n \l_tmpa_tl
5426   \dim_set:Nn \l_tmpc_dim { \pgf@y + 0.5 \l_@@_line_width_dim }
5427   \@@_qpoint:n { #1 }
5428   \pgfpathmoveto { \pgfpoint \pgf@x \l_tmpb_dim }
5429   \pgfpathlineto { \pgfpoint \pgf@x \l_tmpc_dim }
5430   \pgfusepathqstroke
5431 }

```

The following command is used to stroke the top border and the bottom border. The argument #1 is the number of row (in the sense of the row node).

```

5432 \cs_new_protected:Npn \@@_stroke_horizontal:n #1
5433 {
5434   \@@_qpoint:n \l_tmpd_tl
5435   \clist_if_in:NnTF \l_@@_borders_clist { left }
5436     { \dim_set:Nn \l_tmpa_dim { \pgf@x - 0.5 \l_@@_line_width_dim } }
5437     { \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \l_@@_line_width_dim } }
5438   \@@_qpoint:n \l_tmpb_tl
5439   \dim_set:Nn \l_tmpb_dim { \pgf@x + 0.5 \l_@@_line_width_dim }
5440   \@@_qpoint:n { #1 }
5441   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \pgf@y }
5442   \pgfpathlineto { \pgfpoint \l_tmpb_dim \pgf@y }
5443   \pgfusepathqstroke
5444 }

```

Here is the set of keys for the command \@@\_stroke\_borders\_block:nnn.

```

5445 \keys_define:nn { NiceMatrix / BlockBorders }
5446 {
5447   borders .clist_set:N = \l_@@_borders_clist ,
5448   rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
5449   rounded-corners .default:n = 4 pt ,
5450   line-width .dim_set:N = \l_@@_line_width_dim
5451 }

```

## How to draw the dotted lines transparently

```

5452 \cs_set_protected:Npn \@@_renew_matrix:
5453 {
5454   \RenewDocumentEnvironment { pmatrix } { } {
5455     { \pNiceMatrix }
5456     { \endpNiceMatrix }
5457   } \RenewDocumentEnvironment { vmatrix } { } {
5458     { \vNiceMatrix }
5459     { \endvNiceMatrix }
5460   } \RenewDocumentEnvironment { Vmatrix } { } {
5461     { \VNiceMatrix }

```

```

5462     { \endVNiceMatrix }
5463 \RenewDocumentEnvironment { bmatrix } { }
5464     { \bNiceMatrix }
5465     { \endbNiceMatrix }
5466 \RenewDocumentEnvironment { Bmatrix } { }
5467     { \BNiceMatrix }
5468     { \endBNiceMatrix }
5469 }

```

## Automatic arrays

```

5470 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
5471 {
5472   \int_set:Nn \l_@@_nb_rows_int { #1 }
5473   \int_set:Nn \l_@@_nb_cols_int { #2 }
5474 }
5475 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
5476 {
5477   \int_zero_new:N \l_@@_nb_rows_int
5478   \int_zero_new:N \l_@@_nb_cols_int
5479   \@@_set_size:n #4 \q_stop
5480   \begin { NiceArrayWithDelims } { #1 } { #2 }
5481     { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
5482   \int_compare:nNnT \l_@@_first_row_int = 0
5483     {
5484       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
5485       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
5486       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
5487     }
5488   \prg_replicate:nn \l_@@_nb_rows_int
5489     {
5490       \int_compare:nNnT \l_@@_first_col_int = 0 { & }

```

You put { } before #6 to avoid a hasty expansion of a potential \arabic{iRow} at the beginning of the row which would result in an incorrect value of that iRow (since iRow is incremented in the first cell of the row of the \halign).

```

5491       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
5492       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
5493     }
5494   \int_compare:nNnT \l_@@_last_row_int > { -2 }
5495     {
5496       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
5497       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
5498       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
5499     }
5500   \end { NiceArrayWithDelims }
5501 }
5502 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
5503 {
5504   \cs_set_protected:cpn { #1 AutoNiceMatrix }
5505   {
5506     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
5507     \AutoNiceMatrixWithDelims { #2 } { #3 }
5508   }
5509 }
5510 \@@_define_com:nnn p ( )
5511 \@@_define_com:nnn b [ ]
5512 \@@_define_com:nnn v | |
5513 \@@_define_com:nnn V \l \l
5514 \@@_define_com:nnn B \{ \}

```

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



```

5515 \NewDocumentCommand \AutoNiceMatrix { 0 { } m 0 { } m ! 0 { } }
5516 {
5517   \bool_gset_true:N \g_@@_NiceArray_bool
5518   \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
5519   \bool_gset_false: \g_@@_NiceArray_bool
5520 }

```

## The redefinition of the command `\dotfill`

```

5521 \cs_set_eq:NN \@@_old_dotfill \dotfill
5522 \cs_new_protected:Npn \@@_dotfill:
5523 {

```

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}`).

```

5524   \@@_old_dotfill
5525   \bool_if:NT \l_@@_NiceTabular_bool
5526     { \group_insert_after:N \@@_dotfill_ii: }
5527     { \group_insert_after:N \@@_dotfill_i: }
5528   }
5529 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
5530 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }

```

Now, if the box is not empty (unfortunately, we can’t actually test whether the box is empty and that’s why we only consider its width), we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in the cell of the array, and it will extend, since it is no longer in `\l_@@_cell_box`.

```

5531 \cs_new_protected:Npn \@@_dotfill_iii:
5532 { \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } = \c_zero_dim \@@_old_dotfill }

```

## The command `\diagbox`

The command `\diagbox` will be linked to `\diagbox:nn` in the environments of `nicematrix`.

```

5533 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
5534 {
5535   \tl_gput_right:Nx \g_@@_internal_code_after_tl
5536   {
5537     \@@_actually_diagbox:nnnnnn
5538     { \int_use:N \c@iRow }
5539     { \int_use:N \c@jCol }
5540     { \int_use:N \c@iRow }
5541     { \int_use:N \c@jCol }
5542     { \exp_not:n { #1 } }
5543     { \exp_not:n { #2 } }
5544   }

```

We put the cell with `\diagbox` in the sequence `\g_@@_pos_of_blocks_seq` because a cell with `\diagbox` must be considered as non empty by the key `corners`.

```

5545   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
5546   {
5547     { \int_use:N \c@iRow }
5548     { \int_use:N \c@jCol }
5549     { \int_use:N \c@iRow }
5550     { \int_use:N \c@jCol }
5551   }
5552 }

```

The command `\diagbox` is also redefined locally when we draw a block.

The first four arguments of `\@@_actually_diagbox:nnnnnn` correspond to the rectangle (=block) to slash (we recall that it’s possible to use `\diagbox` in a `\Block`). The two other are the elements to draw below and above the diagonal line.

```

5553 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
5554 {

```

```

5555 \pgfpicture
5556 \pgf@relevantforpicturesizefalse
5557 \pgfrememberpicturepositiononpagetrue
5558 \@@_qpoint:n { row - #1 }
5559 \dim_set_eq:NN \l_tmpa_dim \pgf@y
5560 \@@_qpoint:n { col - #2 }
5561 \dim_set_eq:NN \l_tmpb_dim \pgf@x
5562 \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
5563 \@@_qpoint:n { row - \@@_succ:n { #3 } }
5564 \dim_set_eq:NN \l_tmpc_dim \pgf@y
5565 \@@_qpoint:n { col - \@@_succ:n { #4 } }
5566 \dim_set_eq:NN \l_tmpd_dim \pgf@x
5567 \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
5568 {

```

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.

```

5569 \CT@arc@
5570 \pgfsetroundcap
5571 \pgfusepathqstroke
5572 }
5573 \pgfset { inner-sep = 1 pt }
5574 \pgfscope
5575 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
5576 \pgfnode { rectangle } { south-west }
5577 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
5578 \endpgfscope
5579 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
5580 \pgfnode { rectangle } { north-east }
5581 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
5582 \endpgfpicture
5583 }

```

## The keyword `\CodeAfter`

The `\CodeAfter` (inserted with the key `code-after` or after the keyword `\CodeAfter`) may always begin with a list of pairs *key-value* between square brackets. Here is the corresponding set of keys.

```

5584 \keys_define:nn { NiceMatrix }
5585 {
5586   CodeAfter / rules .inherit:n = NiceMatrix / rules ,
5587   CodeAfter / sub-matrix .inherit:n = NiceMatrix / sub-matrix
5588 }
5589 \keys_define:nn { NiceMatrix / CodeAfter }
5590 {
5591   sub-matrix .code:n = \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
5592   sub-matrix .value_required:n = true ,
5593   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
5594   delimiters / color .value_required:n = true ,
5595   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
5596   rules .value_required:n = true ,
5597   unknown .code:n = \@@_error:n { Unknown-key-for-CodeAfter }
5598 }

```

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. 100.

In the environments of `nicematrix`, `\CodeAfter` will be linked to `\@@_CodeAfter:`. That macro must *not* be protected since it begins with `\omit`.

```

5599 \cs_new:Npn \@@_CodeAfter: { \omit \@@_CodeAfter_i:n }

```

However, in each cell of the environment, the command `\CodeAfter` will be linked to the following command `\@@_CodeAfter_i:n` which do *not* begin with `\omit` (and thus, the user will be able to use `\CodeAfter` without error and without the need to prefix by `\omit`).

We have to catch everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```
5600 \cs_new_protected:Npn \@@_CodeAfter_i:n #1 \end
5601 {
5602   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
5603   \@@_CodeAfter_ii:n
5604 }
```

We catch the argument of the command `\end` (in `#1`).

```
5605 \cs_new_protected:Npn \@@_CodeAfter_ii:n #1
5606 {
```

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.

```
5607   \str_if_eq:eeTF \@currenvir { #1 } { \end { #1 } }
```

If this is not the `\end` we are looking for, we put those tokens in `\g_nicematrix_code_after_tl` and we go on searching for the next command `\end` with a recursive call to the command `\@@_CodeAfter:n`.

```
5608   {
5609     \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
5610     \@@_CodeAfter_i:n
5611   }
5612 }
```

## The delimiters in the preamble

The command `\@@_delimiter:nnn` will be used to draw delimiters inside the matrix when delimiters are specified in the preamble of the array. It does *not* concern the exterior delimiters added by `{NiceArrayWithDelims}` (and `{pNiceArray}`, `{pNiceMatrix}`, etc.).

A delimiter in the preamble of the array will write an instruction `\@@_delimiter:nnn` in the `\g_@@_internal_code_after_tl` (and also potentially add instructions in the preamble provided to `\array` in order to add space between columns).

The first argument is the type of delimiter (`(`, `[`, `\{`, `)`, `]` or `\}`). The second argument is the number of column. The third argument is a boolean equal to `\c_true_bool` (resp. `\c_false_true`) when the delimiter must be put on the left (resp. right) side.

```
5613 \cs_new_protected:Npn \@@_delimiter:nnn #1 #2 #3
5614 {
5615   \pgfpicture
5616   \pgfrememberpicturepositiononpagetrue
5617   \pgf@relevantforpicturesizefalse
```

`\l_@@_y_initial_dim` and `\l_@@_y_final_dim` will be the  $y$ -values of the extremities of the delimiter we will have to construct.

```
5618   \@@_qpoint:n { row - 1 }
5619   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
5620   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
5621   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
```

We will compute in `\l_tmpa_dim` the  $x$ -value where we will have to put our delimiter (on the left side or on the right side).

```
5622   \bool_if:nTF { #3 }
5623     { \dim_set_eq:NN \l_tmpa_dim \c_max_dim }
5624     { \dim_set:Nn \l_tmpa_dim { - \c_max_dim } }
5625   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5626     {
5627       \cs_if_exist:cT
5628         { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
```

```

5629     {
5630         \pgfpointanchor
5631         { \l_@@_env: - ##1 - #2 }
5632         { \bool_if:nTF { #3 } { west } { east } }
5633         \dim_set:Nn \l_tmpa_dim
5634         { \bool_if:nTF { #3 } \dim_min:nn \dim_max:nn \l_tmpa_dim \pgf@x }
5635     }
5636 }

```

Now we can put the delimiter with a node of PGF.

```

5637 \pgfset { inner~sep = \c_zero_dim }
5638 \dim_zero:N \nulldelimiterspace
5639 \pgftransformshift
5640 {
5641     \pgfpoint
5642     { \l_tmpa_dim }
5643     { ( \l_@@_y_initial_dim + \l_@@_y_final_dim + \arrayrulewidth ) / 2 }
5644 }
5645 \pgfnode
5646 { rectangle }
5647 { \bool_if:nTF { #3 } { east } { west } }
5648 {

```

Here is the content of the PGF node, that is to say the delimiter, constructed with its right size.

```

5649     \nullfont
5650     \c_math_toggle_token
5651     \tl_if_empty:NF \l_@@_delimiters_color_tl
5652     { \color { \l_@@_delimiters_color_tl } }
5653     \bool_if:nTF { #3 } { \left #1 } { \left . }
5654     \vcenter
5655     {
5656         \nullfont
5657         \hrule \@height
5658             \dim_eval:n { \l_@@_y_initial_dim - \l_@@_y_final_dim }
5659             \@depth \c_zero_dim
5660             \@width \c_zero_dim
5661     }
5662     \bool_if:nTF { #3 } { \right . } { \right #1 }
5663     \c_math_toggle_token
5664 }
5665 { }
5666 { }
5667 \endpgfpicture
5668 }

```

## The command `\SubMatrix`

```

5669 \keys_define:nn { NiceMatrix / sub-matrix }
5670 {
5671     extra-height .dim_set:N = \l_@@_submatrix_extra_height_dim ,
5672     extra-height .value_required:n = true ,
5673     left-xshift .dim_set:N = \l_@@_submatrix_left_xshift_dim ,
5674     left-xshift .value_required:n = true ,
5675     right-xshift .dim_set:N = \l_@@_submatrix_right_xshift_dim ,
5676     right-xshift .value_required:n = true ,
5677     xshift .meta:n = { left-xshift = #1, right-xshift = #1 } ,
5678     xshift .value_required:n = true ,
5679     delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
5680     delimiters / color .value_required:n = true ,
5681     slim .bool_set:N = \l_@@_submatrix_slim_bool ,
5682     slim .default:n = true ,
5683     hlines .clist_set:N = \l_@@_submatrix_hlines_clist ,
5684     hlines .default:n = all ,
5685     vlines .clist_set:N = \l_@@_submatrix_vlines_clist ,

```

```

5686   vlines .default:n = all ,
5687   hvlines .meta:n = { hlines, vlines } ,
5688   hvlines .value_forbidden:n = true ,
5689 }
5690 \keys_define:nn { NiceMatrix }
5691 {
5692   SubMatrix .inherit:n = NiceMatrix / sub-matrix ,
5693   CodeAfter / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5694   NiceMatrix / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5695   NiceArray / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5696   pNiceArray / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5697   NiceMatrixOptions / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5698 }

```

The following keys set is for the command `\SubMatrix` itself (not the tuning of `\SubMatrix` that can be done elsewhere).

```

5699 \keys_define:nn { NiceMatrix / SubMatrix }
5700 {
5701   hlines .clist_set:N = \l_@@_submatrix_hlines_clist ,
5702   hlines .default:n = all ,
5703   vlines .clist_set:N = \l_@@_submatrix_vlines_clist ,
5704   vlines .default:n = all ,
5705   hvlines .meta:n = { hlines, vlines } ,
5706   hvlines .value_forbidden:n = true ,
5707   name .code:n =
5708     \tl_if_empty:nTF { #1 }
5709     { \@@_error:n { Invalid~name~format } }
5710     {
5711       \regex_match:nnTF { \A[A-Za-z][A-Za-z0-9]*\Z } { #1 }
5712       {
5713         \seq_if_in:NnTF \g_@@_submatrix_names_seq { #1 }
5714         { \@@_error:nn { Duplicate~name~for~SubMatrix } { #1 } }
5715         {
5716           \str_set:Nn \l_@@_submatrix_name_str { #1 }
5717           \seq_gput_right:Nn \g_@@_submatrix_names_seq { #1 }
5718         }
5719       }
5720       { \@@_error:n { Invalid~name~format } }
5721     } ,
5722   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
5723   rules .value_required:n = true ,
5724   code .tl_set:N = \l_@@_code_tl ,
5725   code .value_required:n = true ,
5726   name .value_required:n = true ,
5727   unknown .code:n = \@@_error:n { Unknown~key~for~SubMatrix }
5728 }

5729 \NewDocumentCommand \@@_SubMatrix_in_code_before { m m m m ! 0 { } }
5730 {
5731   \@@_cut_on_hyphen:w #3 \q_stop
5732   \tl_clear_new:N \l_tmpc_tl
5733   \tl_clear_new:N \l_tmpd_tl
5734   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
5735   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
5736   \@@_cut_on_hyphen:w #2 \q_stop
5737   \seq_gput_right:Nx \g_@@_submatrix_seq
5738   { { \l_tmpa_tl } { \l_tmpb_tl } { \l_tmpc_tl } { \l_tmpd_tl } }
5739   \tl_gput_right:Nn \g_@@_internal_code_after_tl
5740   { \SubMatrix { #1 } { #2 } { #3 } { #4 } [ #5 ] }
5741 }

```

In the internal code-after and in the `\CodeAfter` the following command `\@@_SubMatrix` will be linked to `\SubMatrix`.

- #1 is the left delimiter;
- #2 is the upper-left cell of the matrix with the format  $i-j$ ;
- #3 is the lower-right cell of the matrix with the format  $i-j$ ;
- #4 is the right delimiter;
- #5 is the list of options of the command.

```

5742 \NewDocumentCommand \@@_SubMatrix { m m m m O { } }
5743 {
5744   \group_begin:

```

The four following token lists correspond to the position of the \SubMatrix.

```

5745   \tl_clear_new:N \l_@@_first_i_tl
5746   \tl_clear_new:N \l_@@_first_j_tl
5747   \tl_clear_new:N \l_@@_last_i_tl
5748   \tl_clear_new:N \l_@@_last_j_tl

```

The command \@@\_cut\_on\_hyphen:w cuts on the hyphen an argument of the form  $i-j$ . The value of  $i$  is stored in \l\_tmpa\_tl and the value of  $j$  is stored in \l\_tmpb\_tl.

```

5749   \@@_cut_on_hyphen:w #2 \q_stop
5750   \tl_set_eq:NN \l_@@_first_i_tl \l_tmpa_tl
5751   \tl_set_eq:NN \l_@@_first_j_tl \l_tmpb_tl
5752   \@@_cut_on_hyphen:w #3 \q_stop
5753   \tl_set_eq:NN \l_@@_last_i_tl \l_tmpa_tl
5754   \tl_set_eq:NN \l_@@_last_j_tl \l_tmpb_tl
5755   \bool_lazy_or:nnTF
5756     { \int_compare_p:nNn \l_@@_last_i_tl > \g_@@_row_total_int }
5757     { \int_compare_p:nNn \l_@@_last_j_tl > \g_@@_col_total_int }
5758     { \@@_error:n { SubMatrix~too~large } }
5759   {
5760     \str_clear_new:N \l_@@_submatrix_name_str
5761     \keys_set:nn { NiceMatrix / SubMatrix } { #5 }
5762     \pgfpicture
5763     \pgfrememberpicturepositiononpagetrue
5764     \pgf@relevantforpicturesizefalse
5765     \pgfset { inner-sep = \c_zero_dim }
5766     \dim_set_eq:NN \l_@@_x_initial_dim \c_max_dim
5767     \dim_set:Nn \l_@@_x_final_dim { - \c_max_dim }

```

The last value of \int\_step\_inline:nnn is provided by currification.

```

5768   \bool_if:NTF \l_@@_submatrix_slim_bool
5769   { \int_step_inline:nnn \l_@@_first_i_tl \l_@@_last_i_tl }
5770   { \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int }
5771   {
5772     \cs_if_exist:cT
5773       { pgf @ sh @ ns @ \@@_env: - ##1 - \l_@@_first_j_tl }
5774     {
5775       \pgfpointanchor { \@@_env: - ##1 - \l_@@_first_j_tl } { west }
5776       \dim_set:Nn \l_@@_x_initial_dim
5777         { \dim_min:nn \l_@@_x_initial_dim \pgf@x }
5778     }
5779     \cs_if_exist:cT
5780       { pgf @ sh @ ns @ \@@_env: - ##1 - \l_@@_last_j_tl }
5781     {
5782       \pgfpointanchor { \@@_env: - ##1 - \l_@@_last_j_tl } { east }
5783       \dim_set:Nn \l_@@_x_final_dim
5784         { \dim_max:nn \l_@@_x_final_dim \pgf@x }
5785     }
5786   }
5787   \dim_compare:nNnTF \l_@@_x_initial_dim = \c_max_dim
5788   { \@@_error:nn { impossible-delimiter } { left } }
5789   {
5790     \dim_compare:nNnTF \l_@@_x_final_dim = { - \c_max_dim }

```

```

5791         { \@@_error:nn { impossible~delimiter } { right } }
5792         { \@@_sub_matrix_i:nn { #1 } { #4 } }
5793     }
5794     \endpgfpicture
5795 }
5796 \group_end:
5797 }

```

#1 is the left delimiter dans #2 is the right one.

```

5798 \cs_new_protected:Npn \@@_sub_matrix_i:nn #1 #2
5799 {
5800     \@@_qpoint:n { row - \l_@@_first_i_tl - base }
5801     \dim_set:Nn \l_@@_y_initial_dim
5802     { \pgf@y + ( \box_ht:N \strutbox + \extrarowheight ) * \arraystretch }
5803     \@@_qpoint:n { row - \l_@@_last_i_tl - base }
5804     \dim_set:Nn \l_@@_y_final_dim
5805     { \pgf@y - ( \box_dp:N \strutbox ) * \arraystretch }
5806     \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
5807     {
5808         \cs_if_exist:cT
5809         { pgf @ sh @ ns @ \@@_env: - \l_@@_first_i_tl - ##1 }
5810         {
5811             \pgfpointanchor { \@@_env: - \l_@@_first_i_tl - ##1 } { north }
5812             \dim_set:Nn \l_@@_y_initial_dim
5813             { \dim_max:nn \l_@@_y_initial_dim \pgf@y }
5814         }
5815         \cs_if_exist:cT
5816         { pgf @ sh @ ns @ \@@_env: - \l_@@_last_i_tl - ##1 }
5817         {
5818             \pgfpointanchor { \@@_env: - \l_@@_last_i_tl - ##1 } { south }
5819             \dim_set:Nn \l_@@_y_final_dim
5820             { \dim_min:nn \l_@@_y_final_dim \pgf@y }
5821         }
5822     }
5823     \dim_set:Nn \l_tmpa_dim
5824     {
5825         \l_@@_y_initial_dim - \l_@@_y_final_dim +
5826         \l_@@_submatrix_extra_height_dim - \arrayrulewidth
5827     }
5828     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim

```

We will draw the rules in the \SubMatrix.

```

5829     \group_begin:
5830     \pgfsetlinewidth { 1.1 \arrayrulewidth }
5831     \tl_if_empty:NF \l_@@_rules_color_tl
5832     { \exp_after:wN \@@_set_CT@arc@: \l_@@_rules_color_tl \q_stop }
5833     \CT@arc@

```

Now, we draw the potential vertical rules specified in the preamble of the environments with the letter fixed with the key `vlines-in-sub-matrix`. The list of the columns where there is such rule to draw is in `\g_@@_cols_vlism_seq`.

```

5834     \seq_map_inline:Nn \g_@@_cols_vlism_seq
5835     {
5836         \int_compare:nNnT \l_@@_first_j_tl < { ##1 }
5837         {
5838             \int_compare:nNnT
5839             { ##1 } < { \int_eval:n { \l_@@_last_j_tl + 1 } }
5840             {

```

First, we extract the value of the abscissa of the rule we have to draw.

```

5841         \@@_qpoint:n { col - ##1 }
5842         \pgfpathmoveto { \pgfpoint \pgf@x \l_@@_y_initial_dim }
5843         \pgfpathlineto { \pgfpoint \pgf@x \l_@@_y_final_dim }

```

```

5844         \pgfusepathqstroke
5845     }
5846 }
5847 }

```

Now, we draw the vertical rules specified in the key `vlines` of `\SubMatrix`. The last argument of `\int_step_inline:nn` or `\clist_map_inline:Nn` is given by curryfication.

```

5848 \tl_if_eq:NnTF \l_@@_submatrix_vlines_clist { all }
5849 { \int_step_inline:nn { \l_@@_last_j_tl - \l_@@_first_j_tl } }
5850 { \clist_map_inline:Nn \l_@@_submatrix_vlines_clist }
5851 {
5852     \bool_lazy_and:nnTF
5853     { \int_compare_p:nNn { ##1 } > 0 }
5854     {
5855         \int_compare_p:nNn
5856         { ##1 } < { \l_@@_last_j_tl - \l_@@_first_j_tl + 1 } }
5857     {
5858         \@@_qpoint:n { col - \int_eval:n { ##1 + \l_@@_first_j_tl } }
5859         \pgfpathmoveto { \pgfpoint \pgf@x \l_@@_y_initial_dim }
5860         \pgfpathlineto { \pgfpoint \pgf@x \l_@@_y_final_dim }
5861         \pgfusepathqstroke
5862     }
5863     { \@@_error:nnn { Wrong~line~in~SubMatrix } { vertical } { ##1 } }
5864 }

```

Now, we draw the horizontal rules specified in the key `hlines` of `\SubMatrix`. The last argument of `\int_step_inline:nn` or `\clist_map_inline:Nn` is given by curryfication.

```

5865 \tl_if_eq:NnTF \l_@@_submatrix_hlines_clist { all }
5866 { \int_step_inline:nn { \l_@@_last_i_tl - \l_@@_first_i_tl } }
5867 { \clist_map_inline:Nn \l_@@_submatrix_hlines_clist }
5868 {
5869     \bool_lazy_and:nnTF
5870     { \int_compare_p:nNn { ##1 } > 0 }
5871     {
5872         \int_compare_p:nNn
5873         { ##1 } < { \l_@@_last_i_tl - \l_@@_first_i_tl + 1 } }
5874     {
5875         \@@_qpoint:n { row - \int_eval:n { ##1 + \l_@@_first_i_tl } }

```

We use a group to protect `\l_tmpa_dim` and `\l_tmpb_dim`.

```

5876 \group_begin:

```

We compute in `\l_tmpa_dim` the  $x$ -value of the left end of the rule.

```

5877 \dim_set:Nn \l_tmpa_dim
5878 { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
5879 \str_case:nn { #1 }
5880 {
5881     ( { \dim_sub:Nn \l_tmpa_dim { 0.9 mm } }
5882     [ { \dim_sub:Nn \l_tmpa_dim { 0.2 mm } }
5883     \{ { \dim_sub:Nn \l_tmpa_dim { 0.9 mm } }
5884     }
5885     \pgfpathmoveto { \pgfpoint \l_tmpa_dim \pgf@y }

```

We compute in `\l_tmpb_dim` the  $x$ -value of the right end of the rule.

```

5886 \dim_set:Nn \l_tmpb_dim
5887 { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
5888 \str_case:nn { #2 }
5889 {
5890     ) { \dim_add:Nn \l_tmpb_dim { 0.9 mm } }
5891     ] { \dim_add:Nn \l_tmpb_dim { 0.2 mm } }
5892     \} { \dim_add:Nn \l_tmpb_dim { 0.9 mm } }
5893 }
5894 \pgfpathlineto { \pgfpoint \l_tmpb_dim \pgf@y }
5895 \pgfusepathqstroke

```



```

5896         \group_end:
5897     }
5898     { \@@_error:nnn { Wrong~line~in~SubMatrix } { horizontal } { ##1 } }
5899 }

```

If the key name has been used for the command `\SubMatrix`, we create a PGF node with that name for the submatrix (this node does not encompass the delimiters that we will put after).

```

5900     \str_if_empty:NF \l_@@_submatrix_name_str
5901     {
5902         \@@_pgf_rect_node:nnnnn \l_@@_submatrix_name_str
5903         \l_@@_x_initial_dim \l_@@_y_initial_dim
5904         \l_@@_x_final_dim \l_@@_y_final_dim
5905     }
5906     \group_end:

```

The group was for `\CT@arc@` (the color of the rules).

Now, we deal with the left delimiter. Of course, the environment `{pgfscope}` is for the `\pgftransformshift`.

```

5907     \begin { pgfscope }
5908     \pgftransformshift
5909     {
5910         \pgfpoint
5911         { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
5912         { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
5913     }
5914     \str_if_empty:NTF \l_@@_submatrix_name_str
5915     { \@@_node_left:nn #1 { } }
5916     { \@@_node_left:nn #1 { \@@_env: - \l_@@_submatrix_name_str - left } }
5917     \end { pgfscope }

```

Now, we deal with the right delimiter.

```

5918     \pgftransformshift
5919     {
5920         \pgfpoint
5921         { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
5922         { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
5923     }
5924     \str_if_empty:NTF \l_@@_submatrix_name_str
5925     { \@@_node_right:nn #2 { } }
5926     {
5927         \@@_node_right:nn #2 { \@@_env: - \l_@@_submatrix_name_str - right }
5928     }
5929     \cs_set_eq:NN \pgfpointanchor \@@_pgfpointanchor:n
5930     \flag_clear_new:n { nicematrix }
5931     \l_@@_code_tl
5932 }

```

In the key code of the command `\SubMatrix` there may be Tikz instructions. We want that, in these instructions, the  $i$  and  $j$  in specifications of nodes of the forms  $i-j$ ,  $\text{row-}i$ ,  $\text{col-}j$  and  $i-|j$  refer to the number of row and column *relative* of the current `\SubMatrix`. That's why we will patch (locally in the `\SubMatrix`) the command `\pgfpointanchor`.

```

5933 \cs_set_eq:NN \@@_old_pgfpointanchor \pgfpointanchor

```

The following command will be linked to `\pgfpointanchor` just before the execution of the option code of the command `\SubMatrix`. In this command, we catch the argument #1 of `\pgfpointanchor` and we apply to it the command `\@@_pgfpointanchor_i:nn` before passing it to the original `\pgfpointanchor`. We have to act in an expandable way because the command `\pgfpointanchor` is used in names of Tikz nodes which are computed in an expandable way.

```

5934 \cs_new_protected:Npn \@@_pgfpointanchor:n #1
5935 {

```

```

5936 \use:e
5937 { \exp_not:N \@@_old_pgfpntanchor { \@@_pgfpntanchor_i:nn #1 } }
5938 }

```

In fact, the argument of `\pgfpntanchor` is always of the form `\a_command { name_of_node }` where “name\_of\_node” is the name of the Tikz node without the potential prefix and suffix. That’s why we catch two arguments and work only on the second by trying (first) to extract an hyphen -.

```

5939 \cs_new:Npn \@@_pgfpntanchor_i:nn #1 #2
5940 { #1 { \@@_pgfpntanchor_ii:w #2 - \q_stop } }

```

Since `\seq_if_in:NnTF` and `\clist_if_in:NnTF` are not expandable, we will use the following token list and `\str_case:nVTF` to test whether we have an integer or not.

```

5941 \tl_const:Nn \c_@@_integers_alist_tl
5942 {
5943   { 1 } { } { 2 } { } { 3 } { } { 4 } { } { 5 } { }
5944   { 6 } { } { 7 } { } { 8 } { } { 9 } { } { 10 } { }
5945   { 11 } { } { 12 } { } { 13 } { } { 14 } { } { 15 } { }
5946   { 16 } { } { 17 } { } { 18 } { } { 19 } { } { 20 } { }
5947 }

```

```

5948 \cs_new:Npn \@@_pgfpntanchor_ii:w #1-#2\q_stop
5949 {

```

If there is no hyphen, that means that the node is of the form of a single number (ex.: 5 or 11). In that case, we are in an analysis which result from a specification of node of the form  $i-j$ . In that case, the  $i$  of the number of row arrives first (and alone) in a `\pgfpntanchor` and, the, the  $j$  arrives (alone) in the following `\pgfpntanchor`. In order to know whether we have a number of row of a number of column, we keep track of the number of such treatments by the expandable flag called `nicematrix`.

```

5950 \tl_if_empty:nTF { #2 }
5951 {
5952   \str_case:nVTF { #1 } \c_@@_integers_alist_tl
5953   {
5954     \flag_raise:n { nicematrix }
5955     \int_if_even:nTF { \flag_height:n { nicematrix } }
5956     { \int_eval:n { #1 + \l_@@_first_i_tl - 1 } }
5957     { \int_eval:n { #1 + \l_@@_first_j_tl - 1 } }
5958   }
5959   { #1 }
5960 }

```

If there is an hyphen, we have to see whether we have a node of the form  $i-j$ , row- $i$  or col- $j$ .

```

5961 { \@@_pgfpntanchor_iii:w { #1 } #2 }
5962 }

```

There was an hyphen in the name of the node and that’s why we have to retrieve the extra hyphen we have put (cf. `\@@_pgfpntanchor_i:nn`).

```

5963 \cs_new:Npn \@@_pgfpntanchor_iii:w #1 #2 -
5964 {
5965   \str_case:nnF { #1 }
5966   {
5967     { row } { row - \int_eval:n { #2 + \l_@@_first_i_tl - 1 } }
5968     { col } { col - \int_eval:n { #2 + \l_@@_first_j_tl - 1 } }
5969   }

```

Now the case of a node of the form  $i-j$ .

```

5970 {
5971   \int_eval:n { #1 + \l_@@_first_i_tl - 1 }
5972   - \int_eval:n { #2 + \l_@@_first_j_tl - 1 }
5973 }
5974 }

```

The command `\@@_node_left:nn` puts the left delimiter with the correct size. The argument #1 is the delimiter to put. The argument #2 is the name we will give to this PGF node (if the key `name` has been used in `\SubMatrix`).

```

5975 \cs_new_protected:Npn \@@_node_left:nn #1 #2
5976 {
5977   \pgfnode
5978   { rectangle }
5979   { east }
5980   {
5981     \nullfont
5982     \c_math_toggle_token
5983     \tl_if_empty:NF \l_@@_delimiters_color_tl
5984     { \color { \l_@@_delimiters_color_tl } }
5985     \left #1
5986     \vcenter
5987     {
5988       \nullfont
5989       \hrule \@height \l_tmpa_dim
5990       \@depth \c_zero_dim
5991       \@width \c_zero_dim
5992     }
5993     \right .
5994     \c_math_toggle_token
5995   }
5996   { #2 }
5997   { }
5998 }

```

The command `\@@_node_right:nn` puts the right delimiter with the correct size. The argument #1 is the delimiter to put. The argument #2 is the name we will give to this PGF node (if the key `name` has been used in `\SubMatrix`).

```

5999 \cs_new_protected:Npn \@@_node_right:nn #1 #2
6000 {
6001   \pgfnode
6002   { rectangle }
6003   { west }
6004   {
6005     \nullfont
6006     \c_math_toggle_token
6007     \tl_if_empty:NF \l_@@_delimiters_color_tl
6008     { \color { \l_@@_delimiters_color_tl } }
6009     \left .
6010     \vcenter
6011     {
6012       \nullfont
6013       \hrule \@height \l_tmpa_dim
6014       \@depth \c_zero_dim
6015       \@width \c_zero_dim
6016     }
6017     \right #1
6018     \c_math_toggle_token
6019   }
6020   { #2 }
6021   { }
6022 }

```

## We process the options at package loading

We process the options when the package is loaded (with `\usepackage`) but we recommend to use `\NiceMatrixOptions` instead.

We must process these options after the definition of the environment {NiceMatrix} because the option `renew-matrix` executes the code `\cs_set_eq:NN \env@matrix \NiceMatrix`. Of course, the command `\NiceMatrix` must be defined before such an instruction is executed.

The boolean `\g_@@_footnotehyper_bool` will indicate if the option `footnotehyper` is used.

```
6023 \bool_new:N \c_@@_footnotehyper_bool
```

The boolean `\c_@@_footnote_bool` will indicate if the option `footnote` is used, but quickly, it will also be set to true if the option `footnotehyper` is used.

```
6024 \bool_new:N \c_@@_footnote_bool
6025 \@@_msg_new:nnn { Unknown~option~for~package }
6026 {
6027   The~key~'\l_keys_key_str'~is~unknown. \\
6028   If~you~go~on,~it~will~be~ignored. \\
6029   For~a~list~of~the~available~keys,~type~H~<return>.
6030 }
6031 {
6032   The~available~keys~are~(in~alphabetic~order):~
6033   define-L-C-R,~
6034   footnote,~
6035   footnotehyper,~
6036   renew-dots,~and
6037   renew-matrix.
6038 }
```

Maybe we will completely delete the key 'transparent' in a future version.

```
6039 \@@_msg_new:nn { Key~transparent }
6040 {
6041   The~key~'transparent'~is~now~obsolete~(because~it's~name~
6042   is~not~clear).~You~should~use~the~conjunction~of~'renew-dots'~
6043   and~'renew-matrix'.~However,~you~can~go~on.
6044 }
6045 \keys_define:nn { NiceMatrix / Package }
6046 {
6047   define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
6048   define-L-C-R .default:n = true ,
6049   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
6050   renew-dots .value_forbidden:n = true ,
6051   renew-matrix .code:n = \@@_renew_matrix: ,
6052   renew-matrix .value_forbidden:n = true ,
6053   transparent .code:n =
6054     {
6055       \@@_renew_matrix:
6056       \bool_set_true:N \l_@@_renew_dots_bool
6057       \@@_error:n { Key~transparent }
6058     } ,
6059   transparent .value_forbidden:n = true,
6060   footnote .bool_set:N = \c_@@_footnote_bool ,
6061   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
6062   unknown .code:n = \@@_error:n { Unknown~option~for~package }
6063 }
6064 \ProcessKeysOptions { NiceMatrix / Package }
```

```
6065 \@@_msg_new:nn { footnote~with~footnotehyper~package }
6066 {
6067   You~can't~use~the~option~'footnote'~because~the~package~
6068   footnotehyper~has~already~been~loaded.~
6069   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
6070   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
6071   of~the~package~footnotehyper.\\
6072   If~you~go~on,~the~package~footnote~won't~be~loaded.
6073 }
```

```

6074 \@@_msg_new:nn { footnotehyper~with~footnote~package }
6075 {
6076   You~can't~use~the~option~'footnotehyper'~because~the~package~
6077   footnote~has~already~been~loaded.~
6078   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
6079   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
6080   of~the~package~footnote.\\
6081   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
6082 }

```

```

6083 \bool_if:NT \c_@@_footnote_bool
6084 {

```

The class beamer has its own system to extract footnotes and that's why we have nothing to do if beamer is used.

```

6085   \@ifclassloaded { beamer }
6086   { \bool_set_false:N \c_@@_footnote_bool }
6087   {
6088     \@ifpackageloaded { footnotehyper }
6089     { \@@_error:n { footnote~with~footnotehyper~package } }
6090     { \usepackage { footnote } }
6091   }
6092 }

```

```

6093 \bool_if:NT \c_@@_footnotehyper_bool
6094 {

```

The class beamer has its own system to extract footnotes and that's why we have nothing to do if beamer is used.

```

6095   \@ifclassloaded { beamer }
6096   { \bool_set_false:N \c_@@_footnote_bool }
6097   {
6098     \@ifpackageloaded { footnote }
6099     { \@@_error:n { footnotehyper~with~footnote~package } }
6100     { \usepackage { footnotehyper } }
6101   }
6102   \bool_set_true:N \c_@@_footnote_bool
6103 }

```

The flag `\c_@@_footnote_bool` is raised and so, we will only have to test `\c_@@_footnote_bool` in order to know if we have to insert an environment `{savenotes}`.

## Error messages of the package

The following message will be deleted when we will delete the key `except-corners` for the command `\arraycolor`.

```

6104 \@@_msg_new:nn { key except-corners }
6105 {
6106   The~key~'except-corners'~has~been~deleted~for~the~command~\token_to_str:N
6107   \arraycolor\ in~the~\token_to_str:N \CodeBefore.~You~should~instead~use~
6108   the~key~'corners'~in~your~\@@_full_name_env:.\\
6109   If~you~go~on,~this~key~will~be~ignored.
6110 }

```

The following message will be deleted when we will delete the key `delimiters-color` (and keep only the key `delimiters/color`).

```

6111 \@@_msg_new:nn { delimiters-color deleted }
6112 {
6113   The~key~'delimiters-color'~has~been~renamed~'delimiters/color'.\\
6114   ~However,~you~can~go~on~for~this~time.
6115 }

```

```

6116 \seq_new:N \c_@@_types_of_matrix_seq
6117 \seq_set_from_clist:Nn \c_@@_types_of_matrix_seq
6118 {
6119     NiceMatrix ,
6120     pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
6121 }
6122 \seq_set_map_x:Nn \c_@@_types_of_matrix_seq \c_@@_types_of_matrix_seq
6123 { \tl_to_str:n { #1 } }

```

If the user uses too much columns, the command `\@@_error_too_much_cols:` is executed. This command raises an error but try to give the best information to the user in the error message. The command `\seq_if_in:NVTF` is not expandable and that's why we can't put it in the error message itself. We have to do the test before the `\@@_fatal:n`.

```

6124 \cs_new_protected:Npn \@@_error_too_much_cols:
6125 {
6126     \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
6127     {
6128         \int_compare:nNnTF \l_@@_last_col_int = { -2 }
6129         { \@@_fatal:n { too~much~cols~for~matrix } }
6130         {
6131             \bool_if:NF \l_@@_last_col_without_value_bool
6132             { \@@_fatal:n { too~much~cols~for~matrix~with~last~col } }
6133         }
6134     }
6135     { \@@_fatal:n { too~much~cols~for~array } }
6136 }

```

The following command must *not* be protected since it's used in an error message.

```

6137 \cs_new:Npn \@@_message_hdotsfor:
6138 {
6139     \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
6140     { ~Maybe~your~use~of~\token_to_str:N \Hdotsfor\ is~incorrect.}
6141 }
6142 \@@_msg_new:nn { too~much~cols~for~matrix~with~last~col }
6143 {
6144     You~try~to~use~more~columns~than~allowed~by~your~
6145     \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~
6146     columns~is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus~the~
6147     exterior~columns).~This~error~is~fatal.
6148 }
6149 \@@_msg_new:nn { too~much~cols~for~matrix }
6150 {
6151     You~try~to~use~more~columns~than~allowed~by~your~
6152     \@@_full_name_env:.\@@_message_hdotsfor:\ Recall~that~the~maximal~
6153     number~of~columns~for~a~matrix~is~fixed~by~the~LaTeX~counter~
6154     'MaxMatrixCols'.~Its~actual~value~is~\int_use:N \c@MaxMatrixCols.~
6155     This~error~is~fatal.
6156 }

```

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`.

```

6157 \@@_msg_new:nn { too~much~cols~for~array }
6158 {
6159     You~try~to~use~more~columns~than~allowed~by~your~
6160     \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~columns~is~
6161     \int_use:N \g_@@_static_num_of_col_int\
6162     ~(plus~the~potential~exterior~ones).~
6163     This~error~is~fatal.
6164 }

```

```

6165 \@@_msg_new:nn { last~col~not~used }
6166 {
6167     The~key~'last~col'~is~in~force~but~you~have~not~used~that~last~column~
6168     in~your~\@@_full_name_env:~.~However,~you~can~go~on.
6169 }
6170 \@@_msg_new:nn { columns~not~used }
6171 {
6172     The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
6173     \g_@@_static_num_of_col_int\ columns~but~you~use~only~\int_use:N \c@jCol.\
6174     You~can~go~on~but~the~columns~you~did~not~used~won't~be~created.
6175 }
6176 \@@_msg_new:nn { in~first~col }
6177 {
6178     You~can't~use~the~command~#1 in~the~first~column~(number~0)~of~the~array.\
6179     If~you~go~on,~this~command~will~be~ignored.
6180 }
6181 \@@_msg_new:nn { in~last~col }
6182 {
6183     You~can't~use~the~command~#1 in~the~last~column~(exterior)~of~the~array.\
6184     If~you~go~on,~this~command~will~be~ignored.
6185 }
6186 \@@_msg_new:nn { in~first~row }
6187 {
6188     You~can't~use~the~command~#1 in~the~first~row~(number~0)~of~the~array.\
6189     If~you~go~on,~this~command~will~be~ignored.
6190 }
6191 \@@_msg_new:nn { in~last~row }
6192 {
6193     You~can't~use~the~command~#1 in~the~last~row~(exterior)~of~the~array.\
6194     If~you~go~on,~this~command~will~be~ignored.
6195 }
6196 \@@_msg_new:nn { double~closing~delimiter }
6197 {
6198     You~can't~put~a~second~closing~delimiter~"#1"~just~after~a~first~closing~
6199     delimiter.~This~delimiter~will~be~ignored.
6200 }
6201 \@@_msg_new:nn { delimiter~after~opening }
6202 {
6203     You~can't~put~a~second~delimiter~"#1"~just~after~a~first~opening~
6204     delimiter.~This~delimiter~will~be~ignored.
6205 }
6206 \@@_msg_new:nn { bad~option~for~line~style }
6207 {
6208     Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line~style'~
6209     is~'standard'.~If~you~go~on,~this~key~will~be~ignored.
6210 }
6211 \@@_msg_new:nn { Unknown~key~for~xdots }
6212 {
6213     As~for~now,~there~is~only~three~key~available~here:~'color',~'line~style'~
6214     and~'shorten'~(and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
6215     this~key~will~be~ignored.
6216 }
6217 \@@_msg_new:nn { Unknown~key~for~rowcolors }
6218 {
6219     As~for~now,~there~is~only~two~keys~available~here:~'cols'~and~'respect~blocks'~
6220     (and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
6221     this~key~will~be~ignored.
6222 }

```

```

6223 \@@_msg_new:nn { ampersand~in~light-syntax }
6224 {
6225     You~can't~use~an~ampersand~(\token_to_str:N &)~to~separate~columns~because~
6226     ~you~have~used~the~key~'light-syntax'.~This~error~is~fatal.
6227 }
6228 \@@_msg_new:nn { SubMatrix~too~large }
6229 {
6230     Your~command~\token_to_str:N \SubMatrix\
6231     can't~be~drawn~because~your~matrix~is~too~small.\\
6232     If~you~go~on,~this~command~will~be~ignored.
6233 }
6234 \@@_msg_new:nn { double-backslash~in~light-syntax }
6235 {
6236     You~can't~use~\token_to_str:N \\~to~separate~rows~because~you~have~used~
6237     the~key~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
6238     (set~by~the~key~'end-of-row').~This~error~is~fatal.
6239 }
6240 \@@_msg_new:nn { standard~cline~in~document }
6241 {
6242     The~key~'standard~cline'~is~available~only~in~the~preamble.\\
6243     If~you~go~on~this~command~will~be~ignored.
6244 }
6245 \@@_msg_new:nn { old~column~type }
6246 {
6247     The~column~type~'#1'~is~no~longer~defined~in~'nicematrix'.~
6248     Since~version~5.0,~you~have~to~use~'l',~'c'~and~'r'~instead~of~'L',~
6249     'C'~and~'R'.~You~can~also~use~the~key~'define-L-C-R'.\\
6250     This~error~is~fatal.
6251 }
6252 \@@_msg_new:nn { bad~value~for~baseline }
6253 {
6254     The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
6255     valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
6256     \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\\
6257     If~you~go~on,~a~value~of~1~will~be~used.
6258 }
6259 \@@_msg_new:nn { Invalid~name~format }
6260 {
6261     You~can't~give~the~name~'\l_keys_value_tl'~to~a~\token_to_str:N
6262     \SubMatrix.\\
6263     A~name~must~be~accepted~by~the~regular~expression~[A-Za-z][A-Za-z0-9]*.\\
6264     If~you~go~on,~this~key~will~be~ignored.
6265 }
6266 \@@_msg_new:nn { Wrong~line~in~SubMatrix }
6267 {
6268     You~try~to~draw~a~#1~line~of~number~'#2'~in~a~
6269     \token_to_str:N \SubMatrix\ of~your~\@@_full_name_env:\ but~that~
6270     number~is~not~valid.~If~you~go~on,~it~will~be~ignored.
6271 }
6272 \@@_msg_new:nn { impossible~delimiter }
6273 {
6274     It's~impossible~to~draw~the~#1~delimiter~of~your~
6275     \token_to_str:N \SubMatrix\ because~all~the~cells~are~empty~
6276     in~that~column.
6277     \bool_if:NT \l_@@_submatrix_slim_bool
6278     { ~Maybe~you~should~try~without~the~key~'slim'. } \\
6279     If~you~go~on,~this~\token_to_str:N \SubMatrix\ will~be~ignored.
6280 }
6281 \@@_msg_new:nn { empty~environment }
6282 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }

```



```

6283 \@@_msg_new:nn { Delimiter~with~small }
6284 {
6285     You~can't~put~a~delimiter~in~the~preamble~of~your~\@@_full_name_env:\
6286     because~the~key~'small'~is~in~force.\\
6287     This~error~is~fatal.
6288 }
6289 \@@_msg_new:nn { unknown~cell~for~line~in~CodeAfter }
6290 {
6291     Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
6292     can't~be~executed~because~a~cell~doesn't~exist.\\
6293     If~you~go~on~this~command~will~be~ignored.
6294 }
6295 \@@_msg_new:nnn { Duplicate~name~for~SubMatrix }
6296 {
6297     The~name~'#1'~is~already~used~for~a~\token_to_str:N \SubMatrix\
6298     in~this~\@@_full_name_env:.\
6299     If~you~go~on,~this~key~will~be~ignored.\\
6300     For~a~list~of~the~names~already~used,~type~H~<return>.
6301 }
6302 {
6303     The~names~already~defined~in~this~\@@_full_name_env:\ are:~
6304     \seq_use:Nnnn \g_@@_submatrix_names_seq { ~and~ } { ,~ } { ~and~ }.
6305 }
6306 \@@_msg_new:nn { r~or~l~with~preamble }
6307 {
6308     You~can't~use~the~key~'\l_keys_key_str'~in~your~\@@_full_name_env:~
6309     You~must~specify~the~alignment~of~your~columns~with~the~preamble~of~
6310     your~\@@_full_name_env:.\
6311     If~you~go~on,~this~key~will~be~ignored.
6312 }
6313 \@@_msg_new:nn { Hdotsfor~in~col~0 }
6314 {
6315     You~can't~use~\token_to_str:N \Hdotsfor\ in~an~exterior~column~of~
6316     the~array.~This~error~is~fatal.
6317 }
6318 \@@_msg_new:nn { bad~corner }
6319 {
6320     #1~is~an~incorrect~specification~for~a~corner~(in~the~keys~
6321     'corners'~and~'except~corners').~The~available~
6322     values~are:~NW,~SW,~NE~and~SE.\\
6323     If~you~go~on,~this~specification~of~corner~will~be~ignored.
6324 }
6325 \@@_msg_new:nn { bad~border }
6326 {
6327     #1~is~an~incorrect~specification~for~a~border~(in~the~key~
6328     'borders'~of~the~command~\token_to_str:N \Block).~The~available~
6329     values~are:~left,~right,~top~and~bottom.\\
6330     If~you~go~on,~this~specification~of~border~will~be~ignored.
6331 }
6332 \@@_msg_new:nn { last~col~non~empty~for~NiceArray }
6333 {
6334     In~the~\@@_full_name_env:,~you~must~use~the~key~
6335     'last~col'~without~value.\\
6336     However,~you~can~go~on~for~this~time~
6337     (the~value~'\l_keys_value_tl'~will~be~ignored).
6338 }
6339 \@@_msg_new:nn { last~col~non~empty~for~NiceMatrixOptions }
6340 {
6341     In~\NiceMatrixoptions,~you~must~use~the~key~
6342     'last~col'~without~value.\\

```

```

6343     However,~you~can~go~on~for~this~time~
6344     (the~value~'\l_keys_value_tl'~will~be~ignored).
6345 }

6346 \@@_msg_new:nn { Block-too-large-1 }
6347 {
6348     You~try~to~draw~a~block~in~the~cell~#1-#2~of~your~matrix~but~the~matrix~is~
6349     too~small~for~that~block. \\
6350 }

6351 \@@_msg_new:nn { Block-too-large-2 }
6352 {
6353     The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
6354     \g_@@_static_num_of_col_int\
6355     columns~but~you~use~only~\int_use:N \c@jCol\ and~that's~why~a~block~
6356     specified~in~the~cell~#1-#2~can't~be~drawn.~You~should~add~some~ampersands~
6357     (&)~at~the~end~of~the~first~row~of~your~
6358     \@@_full_name_env:.\
6359     If~you~go~on,~this~block~and~maybe~others~will~be~ignored.
6360 }

6361 \@@_msg_new:nn { unknown-column-type }
6362 {
6363     The~column~type~'#1'~in~your~\@@_full_name_env:\
6364     is~unknown. \\
6365     This~error~is~fatal.
6366 }

6367 \@@_msg_new:nn { tabularnote-forbidden }
6368 {
6369     You~can't~use~the~command~\token_to_str:N\tabularnote\
6370     ~in~a~\@@_full_name_env:~This~command~is~available~only~in~
6371     \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
6372     If~you~go~on,~this~command~will~be~ignored.
6373 }

6374 \@@_msg_new:nn { borders-forbidden }
6375 {
6376     You~can't~use~the~key~'borders'~of~the~command~\token_to_str:N \Block\
6377     because~the~option~'rounded-corners'~
6378     is~in~force~with~a~non-zero~value.\\
6379     If~you~go~on,~this~key~will~be~ignored.
6380 }

6381 \@@_msg_new:nn { bottomrule-without-booktabs }
6382 {
6383     You~can't~use~the~key~'tabular/bottomrule'~because~you~haven't~
6384     loaded~'booktabs'.\\
6385     If~you~go~on,~this~key~will~be~ignored.
6386 }

6387 \@@_msg_new:nn { enumitem-not-loaded }
6388 {
6389     You~can't~use~the~command~\token_to_str:N\tabularnote\
6390     ~because~you~haven't~loaded~'enumitem'.\\
6391     If~you~go~on,~this~command~will~be~ignored.
6392 }

6393 \@@_msg_new:nn { Wrong-last-row }
6394 {
6395     You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
6396     \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
6397     If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
6398     last~row.~You~can~avoid~this~problem~by~using~'last-row'~
6399     without~value~(more~compilations~might~be~necessary).
6400 }

6401 \@@_msg_new:nn { Yet-in-env }
6402 { Environments~of~nicematrix~can't~be~nested.\\ This~error~is~fatal. }

```

```

6403 \@@_msg_new:nn { Outside-math-mode }
6404 {
6405   The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
6406   (and~not~in~\token_to_str:N \vcenter).\
6407   This~error~is~fatal.
6408 }
6409 \@@_msg_new:nn { One-letter-allowed }
6410 {
6411   The~value~of~key~'\l_keys_key_str'~must~be~of~length~1.\
6412   If~you~go~on,~it~will~be~ignored.
6413 }
6414 \@@_msg_new:nnn { Unknown~key~for~Block }
6415 {
6416   The~key~'\l_keys_key_str'~is~unknown~for~the~command~\token_to_str:N
6417   \Block.\ If~you~go~on,~it~will~be~ignored. \
6418   For~a~list~of~the~available~keys,~type~H~<return>.
6419 }
6420 {
6421   The~available~keys~are~(in~alphabetic~order):~b,~borders,~c,~draw,~fill,~
6422   hvlines,~l,~line-width,~rounded-corners,~r~and~t.
6423 }
6424 \@@_msg_new:nnn { Unknown~key~for~CodeAfter }
6425 {
6426   The~key~'\l_keys_key_str'~is~unknown.\
6427   If~you~go~on,~it~will~be~ignored. \
6428   For~a~list~of~the~available~keys~in~\token_to_str:N
6429   \CodeAfter,~type~H~<return>.
6430 }
6431 {
6432   The~available~keys~are~(in~alphabetic~order):~
6433   delimiters/color,~
6434   rules~(with~the~subkeys~'color'~and~'width'),~
6435   sub-matrix~(several~subkeys)~
6436   and~xdots~(several~subkeys).~
6437   The~latter~is~for~the~command~\token_to_str:N \line.
6438 }
6439 \@@_msg_new:nnn { Unknown~key~for~SubMatrix }
6440 {
6441   The~key~'\l_keys_key_str'~is~unknown.\
6442   If~you~go~on,~this~key~will~be~ignored. \
6443   For~a~list~of~the~available~keys~in~\token_to_str:N
6444   \SubMatrix,~type~H~<return>.
6445 }
6446 {
6447   The~available~keys~are~(in~alphabetic~order):~
6448   'delimiters/color',~
6449   'extra-height',~
6450   'hlines',~
6451   'hvlines',~
6452   'left-xshift',~
6453   'name',~
6454   'right-xshift',~
6455   'rules'~(with~the~subkeys~'color'~and~'width'),~
6456   'slim',~
6457   'vlines'~and~'xshift'~(which~sets~both~'left-xshift'~
6458   and~'right-xshift').\
6459 }
6460 \@@_msg_new:nnn { Unknown~key~for~notes }
6461 {
6462   The~key~'\l_keys_key_str'~is~unknown.\
6463   If~you~go~on,~it~will~be~ignored. \
6464   For~a~list~of~the~available~keys~about~notes,~type~H~<return>.

```

```

6465 }
6466 {
6467   The~available~keys~are~(in~alphabetic~order):~
6468   bottomrule,~
6469   code-after,~
6470   code-before,~
6471   enumitem-keys,~
6472   enumitem-keys-para,~
6473   para,~
6474   label-in-list,~
6475   label-in-tabular~and~
6476   style.
6477 }

6478 \@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
6479 {
6480   The~key~'\l_keys_key_str'~is~unknown~for~the~command~
6481   \token_to_str:N \NiceMatrixOptions. \\
6482   If~you~go~on,~it~will~be~ignored. \\
6483   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
6484 }
6485 {
6486   The~available~keys~are~(in~alphabetic~order):~
6487   allow-duplicate-names,~
6488   cell-space-bottom-limit,~
6489   cell-space-limits,~
6490   cell-space-top-limit,~
6491   code-for-first-col,~
6492   code-for-first-row,~
6493   code-for-last-col,~
6494   code-for-last-row,~
6495   corners,~
6496   create-extra-nodes,~
6497   create-medium-nodes,~
6498   create-large-nodes,~
6499   delimiters/color,~
6500   end-of-row,~
6501   first-col,~
6502   first-row,~
6503   hlines,~
6504   hvlines,~
6505   last-col,~
6506   last-row,~
6507   left-margin,~
6508   letter-for-dotted-lines,~
6509   light-syntax,~
6510   notes~(several~subkeys),~
6511   nullify-dots,~
6512   renew-dots,~
6513   renew-matrix,~
6514   right-margin,~
6515   rules~(with~the~subkeys~'color'~and~'width'),~
6516   small,~
6517   sub-matrix~(several~subkeys),~
6518   vlines,~
6519   xdots~(several~subkeys).
6520 }

6521 \@_msg_new:nnn { Unknown~option~for~NiceArray }
6522 {
6523   The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
6524   \{NiceArray\}. \\
6525   If~you~go~on,~it~will~be~ignored. \\
6526   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
6527 }

```

```

6528 {
6529     The~available~keys~are~(in~alphabetic~order):~
6530     b,~
6531     baseline,~
6532     c,~
6533     cell-space-bottom-limit,~
6534     cell-space-limits,~
6535     cell-space-top-limit,~
6536     code-after,~
6537     code-for-first-col,~
6538     code-for-first-row,~
6539     code-for-last-col,~
6540     code-for-last-row,~
6541     colortbl-like,~
6542     columns-width,~
6543     corners,~
6544     create-extra-nodes,~
6545     create-medium-nodes,~
6546     create-large-nodes,~
6547     delimiters/color,~
6548     extra-left-margin,~
6549     extra-right-margin,~
6550     first-col,~
6551     first-row,~
6552     hlines,~
6553     hvlines,~
6554     last-col,~
6555     last-row,~
6556     left-margin,~
6557     light-syntax,~
6558     name,~
6559     notes/bottomrule,~
6560     notes/para,~
6561     nullify-dots,~
6562     renew-dots,~
6563     right-margin,~
6564     rules~(with~the~subkeys~'color'~and~'width'),~
6565     small,~
6566     t,~
6567     tabularnote,~
6568     vlines,~
6569     xdots/color,~
6570     xdots/shorten~and~
6571     xdots/line-style.
6572 }

```

This error message is used for the set of keys NiceMatrix/NiceMatrix and NiceMatrix/pNiceArray (but not by NiceMatrix/NiceArray because, for this set of keys, there is also the keys t, c and b).

```

6573 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
6574 {
6575     The~key~'\l_keys_key_str'~is~unknown~for~the~
6576     \@@_full_name_env:. \
6577     If~you~go~on,~it~will~be~ignored. \
6578     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
6579 }
6580 {
6581     The~available~keys~are~(in~alphabetic~order):~
6582     b,~
6583     baseline,~
6584     c,~
6585     cell-space-bottom-limit,~
6586     cell-space-limits,~
6587     cell-space-top-limit,~

```

```

6588 code-after,~
6589 code-for-first-col,~
6590 code-for-first-row,~
6591 code-for-last-col,~
6592 code-for-last-row,~
6593 colortbl-like,~
6594 columns-width,~
6595 corners,~
6596 create-extra-nodes,~
6597 create-medium-nodes,~
6598 create-large-nodes,~
6599 delimiters/color,~
6600 extra-left-margin,~
6601 extra-right-margin,~
6602 first-col,~
6603 first-row,~
6604 hlines,~
6605 hvlines,~
6606 l,~
6607 last-col,~
6608 last-row,~
6609 left-margin,~
6610 light-syntax,~
6611 name,~
6612 nullify-dots,~
6613 r,~
6614 renew-dots,~
6615 right-margin,~
6616 rules~(with~the~subkeys~'color'~and~'width'),~
6617 small,~
6618 t,~
6619 vlines,~
6620 xdots/color,~
6621 xdots/shorten~and~
6622 xdots/line-style.
6623 }

6624 \@_msg_new:nnn { Unknown~option~for~NiceTabular }
6625 {
6626   The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
6627   \{NiceTabular\}. \\
6628   If~you~go~on,~it~will~be~ignored. \\
6629   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
6630 }
6631 {
6632   The~available~keys~are~(in~alphabetic~order):~
6633   b,~
6634   baseline,~
6635   c,~
6636   cell-space-bottom-limit,~
6637   cell-space-limits,~
6638   cell-space-top-limit,~
6639   code-after,~
6640   code-for-first-col,~
6641   code-for-first-row,~
6642   code-for-last-col,~
6643   code-for-last-row,~
6644   colortbl-like,~
6645   columns-width,~
6646   corners,~
6647   create-extra-nodes,~
6648   create-medium-nodes,~
6649   create-large-nodes,~
6650   extra-left-margin,~

```

```

6651 extra-right-margin,~
6652 first-col,~
6653 first-row,~
6654 hlines,~
6655 hvlines,~
6656 last-col,~
6657 last-row,~
6658 left-margin,~
6659 light-syntax,~
6660 name,~
6661 notes/bottomrule,~
6662 notes/para,~
6663 nullify-dots,~
6664 renew-dots,~
6665 right-margin,~
6666 rules~(with~the~subkeys~'color'~and~'width'),~
6667 t,~
6668 tabularnote,~
6669 vlines,~
6670 xdots/color,~
6671 xdots/shorten~and~
6672 xdots/line-style.
6673 }

6674 \@@_msg_new:nnn { Duplicate-name }
6675 {
6676   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
6677   the~same~environment~name~twice.~You~can~go~on,~but,~
6678   maybe,~you~will~have~incorrect~results~especially~
6679   if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
6680   message~again,~use~the~key~'allow-duplicate-names'~in~
6681   '\token_to_str:N \NiceMatrixOptions'.\\
6682   For~a~list~of~the~names~already~used,~type~H~<return>. \\
6683 }
6684 {
6685   The~names~already~defined~in~this~document~are:~
6686   \seq_use:Nnnn \g_@@_names_seq { ~and~ } { ,~ } { ~and~ }.
6687 }

6688 \@@_msg_new:nn { Option-auto-for-columns-width }
6689 {
6690   You~can't~give~the~value~'auto'~to~the~key~'columns-width'~here.~
6691   If~you~go~on,~the~key~will~be~ignored.
6692 }

```

## 18 History

The successive versions of the file `nicematrix.sty` provided by TeXLive are available on the SVN server of TeXLive:

<https://www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/nicematrix/nicematrix.sty>

### Changes between versions 1.0 and 1.1

The dotted lines are no longer drawn with Tikz nodes but with Tikz circles (for efficiency).  
Modification of the code which is now twice faster.

## Changes between versions 1.1 and 1.2

New environment `\NiceArray` with column types L, C and R.

## Changes between version 1.2 and 1.3

New environment `\pNiceArrayC` and its variants.

Correction of a bug in the definition of `\BNiceMatrix`, `\vNiceMatrix` and `\VNiceMatrix` (in fact, it was a typo).

Options are now available locally in `\pNiceMatrix` and its variants.

The names of the options are changed. The old names were names in “camel style”.

## Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `\NiceArray`, `\pNiceArrayC` and its variants with the same meaning as in the package `array`.

New option `columns-width` to fix the same width for all the columns of the array.

## Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of `nicematrix` were focused on the continuous dotted lines whereas the version 2.0 of `nicematrix` provides different features to improve the typesetting of mathematical matrices.

## Changes between version 2.0 and 2.1

New implementation of the environment `\pNiceArrayRC`. With this new implementation, there is no restriction on the width of the columns.

The package `nicematrix` no longer loads `mathtools` but only `amsmath`.

Creation of “medium nodes” and “large nodes”.

## Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.

Following a discussion on TeX StackExchange<sup>62</sup>, Tikz externalization is now deactivated in the environments of the package `nicematrix`.<sup>63</sup>

## Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\begin{pmatrix} & C_j & \\ 0 & \vdots & 0 \\ & a \cdots & \\ 0 & & 0 \end{pmatrix} L_i$$

---

<sup>62</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>63</sup>Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.



## Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See [www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)

## Changes between version 2.1.4 and 2.1.5

Compatibility with the classes `revtex4-1` and `revtex4-2`.

Option `allow-duplicate-names`.

## Changes between version 2.1.5 and 2.2

Possibility to draw horizontal dotted lines to separate rows with the command `\hdottedline` (similar to the classical command `\hline` and the command `\hdashline` of `arydshln`).

Possibility to draw vertical dotted lines to separate columns with the specifier `“:”` in the preamble (similar to the classical specifier `“|”` and the specifier `“:”` of `arydshln`).

## Changes between version 2.2 and 2.2.1

Improvement of the vertical dotted lines drawn by the specifier `“:”` in the preamble.

Modification of the position of the dotted lines drawn by `\hdottedline`.

## Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.

Option `hlines`.

## Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of `“|”`) as `\hdotsfor` does.

Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

## Changes between version 3.0 and 3.1

Command `\Block` to draw block matrices.

Error message when the user gives an incorrect value for `last-row`.

A dotted line can no longer cross another dotted line (excepted the dotted lines drawn by `\cdottedline`, the symbol `“:”` (in the preamble of the array) and `\line` in `code-after`).

The starred versions of `\Cdots`, `\Ldots`, etc. are now deprecated because, with the new implementation, they become pointless. These starred versions are no longer documented.

The vertical rules in the matrices (drawn by `“|”`) are now compatible with the color fixed by `colortbl`.

Correction of a bug: it was not possible to use the colon `“:”` in the preamble of an array when `pdflatex` was used with `french-babel` (because `french-babel` activates the colon in the beginning of the document).

## Changes between version 3.1 and 3.2 (and 3.2a)

Option `small`.

## Changes between version 3.2 and 3.3

The options `first-row`, `last-row`, `first-col` and `last-col` are now available in the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.

The option `columns-width=auto` doesn't need any more a second compilation.

The options `renew-dots`, `renew-matrix` and `transparent` are now available as package options (as said in the documentation).

The previous version of `nicematrix` was incompatible with a recent version of `expl3` (released 2019/09/30). This version is compatible.

## Changes between version 3.3 and 3.4

Following a discussion on TeX StackExchange<sup>64</sup>, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

## Changes between version 3.4 and 3.5

Correction on a bug on the two previous versions where the `code-after` was not executed.

## Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate-name` in the environments of `amsmath`.

## Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

## Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`. An error is raised when an obsolete environment is used.

## Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.

New options `create-medium-nodes` and `create-large-nodes`.

## Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).

New option `dotted-lines-margin` for fine tuning of the dotted lines.

---

<sup>64</sup>cf. [tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize](https://tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize)

## Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

## Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.

Options `vlines`, `hlines` and `hvlines`.

Option `baseline` pour `{NiceArray}` (not for the other environments).

The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell  $i-j$ , the name is  $i-j$ -block and, if the creation of the “medium nodes” is required, a node  $i-j$ -block-medium is created.

If the user tries to use more columns than allowed by its environment, an error is raised by `nicematrix` (instead of a low-level error).

The package must be loaded with the option `obsolete-environments` if we want to use the deprecated environments.

## Changes between versions 3.12 and 3.13

The behaviour of the command `\rotate` is improved when used in the “last row”.

The option `dotted-lines-margin` has been renamed in `xdots/shorten` and the options `xdots/color` and `xdots/line-style` have been added for a complete customisation of the dotted lines.

In the environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, etc.), it’s possible to use the options `l` (=L) or `r` (=R) to specify the type of the columns.

The starred versions of the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots` and `\Iddots` are deprecated since the version 3.1 of `nicematrix`. Now, one should load `nicematrix` with the option `starred-commands` to avoid an error at the compilation.

The code of `nicematrix` no longer uses Tikz but only PGF. By default, Tikz is *not* loaded by `nicematrix`.

## Changes between versions 3.13 and 3.14

Correction of a bug (question 60761504 on [stackoverflow](#)).

Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.

## Changes between versions 3.14 and 3.15

It’s possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the `code-after` with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New keyword `\CodeAfter` (in the environments of `nicematrix`).

## Changes between versions 3.15 and 4.0

New environment `{NiceTabular}`

Commands to color cells, rows and columns with a perfect result in the PDF.

## Changes between versions 4.0 and 4.1

New keys `cell-space-top-limit` and `cell-space-bottom-limit`

New command `\diagbox`

The key `hvline` don’t draw rules in the blocks (commands `\Block`) and in the virtual blocks corresponding to the dotted lines.

## Changes between versions 4.1 and 4.2

It's now possible to write `\begin{pNiceMatrix}a&b\\c&d\end{pNiceMatrix}`<sup>2</sup> with the expected result.

## Changes between versions 4.2 and 4.3

The horizontal centering of the content of a `\Block` is correct even when an instruction such as `!\qqquad` is used in the preamble of the array.

It's now possible to use the command `\Block` in the “last row”.

## Changes between versions 4.3 and 4.4

New key `hvlines-except-corners`.

## Changes between versions 4.4 and 5.0

Use of the standard column types `l`, `c` and `r` instead of `L`, `C` and `R`.

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

Command `\tabularnote`

## Changes between versions 5.0 and 5.1

The vertical rules specified by `|` in the preamble are not broken by `\hline\hline` (and other).

Environment `{NiceTabular*}`

Command `\Vdotsfor` similar to `\Hdotsfor`

The variable `\g_nicematrix_code_after_tl` is now public.

## Changes between versions 5.1 and 5.2

The vertical rules specified by `|` or `||` in the preamble respect the blocks.

Key `respect-blocks` for `\rowcolors` (with a *s*) in the `code-before`.

The variable `\g_nicematrix_code_before_tl` is now public.

The key `baseline` may take in as value an expression of the form *line-i* to align the `\hline` in the row *i*.

The key `hvlines-except-corners` may take in as value a list of corners (eg: NW,SE).

## Changes between versions 5.2 and 5.3

Keys `c`, `r` and `l` for the command `\Block`.

It's possible to use the key `draw-first` with `\Ddots` and `\Iddots` to specify which dotted line will be drawn first (the other lines will be drawn parallel to that one if parallelization is activated).

## Changes between versions 5.3 and 5.4

Key `tabularnote`.

Different behaviour for the mono-column blocks.

## Changes between versions 5.4 and 5.5

The user must never put `\omit` before `\CodeAfter`.

Correction of a bug: the tabular notes `\tabularnotes` were not composed when present in a block (except a mono-column block).

## Changes between versions 5.5 and 5.6

Different behaviour for the mono-row blocks.  
New command `\NotEmpty`.

## Changes between versions 5.6 and 5.7

New key `delimiters-color`  
Keys `fill`, `draw` and `line-width` for the command `\Block`.

## Changes between versions 5.7 and 5.8

Keys `cols` and `restart` of the command `\rowcolors` in the `code-before`.  
Modification of the behaviour of `\\` in the columns of type `p`, `m` or `b` (for a behaviour similar to the environments of `array`).  
Better error messages for the command `\Block`.

## Changes between versions 5.8 and 5.9

Correction of a bug: in the previous versions, it was not possible to use the key `line-style` for the continuous dotted lines when the Tikz library `babel` was loaded.  
New key `cell-space-limits`.

## Changes between versions 5.9 and 5.10

New command `\SubMatrix` available in the `\CodeAfter`.  
It's possible to provide options (between brackets) to the keyword `\CodeAfter`.  
A (non fatal) error is raised when the key `transparent`, which is deprecated, is used.

## Changes between versions 5.10 and 5.11

It's now possible, in the `code-before` and in the `\CodeAfter`, to use the syntax `|(i-|j)` for the Tikz node at the intersection of the (potential) horizontal rule number  $i$  and the (potential) vertical rule number  $j$ .

## Changes between versions 5.11 and 5.12

Keywords `\CodeBefore` and `\Body` (alternative syntax to the key `code-before`).  
New key `delimiters/max-width`.  
New keys `hlines`, `vlines` and `hvlines` for the command `\SubMatrix` in the `\CodeAfter`.  
New key `rounded-corners` for the command `\Block`.

## Changes between versions 5.12 and 5.13

New command `\arraycolor` in the `\CodeBefore` (with its key `except-corners`).  
New key `borders` for the command `\Block`.  
New command `\Hline` (for horizontal rules not drawn in the blocks).  
The keys `vlines` and `hlines` takes in as value a (comma-separated) list of numbers (for the rules to draw).

## Changes between versions 5.13 and 5.14

Nodes of the form  $(1.5)$ ,  $(2.5)$ ,  $(3.5)$ , etc.  
Keys `t` and `b` for the command `\Block`.  
Key `corners`.

## Changes between versions 5.14 and 5.15

Key hvlines for the command \Block.

The commands provided by nicematrix to color cells, rows and columns don't color the cells which are in the "corners" (when the key corner is used).

It's now possible to specify delimiters for submatrices in the preamble of an environment.

## 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:	
\@@_Block: .....	1158, 4822
\@@_Block_i .....	4827, 4828, 4832
\@@_Block_ii:nnnnn .....	4832, 4833
\@@_Block_iv:nnnnn .....	4863, 4867
\@@_Block_iv:nnnnnn .....	5047, 5049
\@@_Block_v:nnnnn .....	4864, 4968
\@@_Block_v:nnnnnn .....	5076, 5079
\@@_Cdots .....	1083, 1148, 3341
\g_@@_Cdots_lines_tl .....	1175, 2543
\@@_Cell: .....	201, 836, 1613, 1660, 1682, 2323, 3441, 3442, 3443, 3444, 3445, 3446, 3447, 3448, 3449, 3450, 3451, 3452
\@@_CodeAfter: .....	1162, 5599
\@@_CodeAfter_i:n .....	838, 2201, 2246, 5599, 5600, 5610
\@@_CodeAfter_ii:n .....	5603, 5605
\@@_CodeAfter_keys: .....	1247, 2481, 2504
\@@_Ddots .....	1085, 1150, 3373
\g_@@_Ddots_lines_tl .....	1178, 2541
\g_@@_HVDotsfor_lines_tl .....	1180, 2539, 3500, 3576, 6139
\@@_Hdotsfor: .....	1088, 1155, 3476
\@@_Hdotsfor:nnnn .....	3502, 3514
\@@_Hdotsfor_i .....	3485, 3491, 3498
\@@_Hline: .....	1153, 4319
\@@_Hline_i:n .....	4319, 4320, 4326
\@@_Hline_ii:nn .....	4323, 4326
\@@_Hline_iii:n .....	4324, 4327
\@@_Hspace: .....	1154, 3427
\@@_Iddots .....	1086, 1151, 3397
\g_@@_Iddots_lines_tl .....	1179, 2542
\@@_Ldots .....	1082, 1087, 1147, 3325
\g_@@_Ldots_lines_tl .....	1176, 2544
\l_@@_Matrix_bool .....	247, 1474, 1490, 2314
\g_@@_NiceArray_bool .....	244, 367, 1306, 1372, 1414, 1515, 1530, 1542, 1710, 1753, 2290, 2499, 4188, 4189, 4311, 4312, 4529, 4536, 5517, 5519
\g_@@_NiceMatrixBlock_int .....	235, 4588, 4593, 4596, 4607
\l_@@_NiceTabular_bool .....	156, 245, 843, 1009, 1245, 1249, 1347, 1447, 1451, 1531, 1543, 2343, 2352, 4895, 4977, 5525
\@@_NotEmpty: .....	1164, 2337
\@@_OnlyMainNiceMatrix:n .....	1160, 4049
\@@_OnlyMainNiceMatrix_i:n .....	4052, 4059, 4062
\@@_SubMatrix .....	2463, 5742
\@@_SubMatrix_in_code_before ...	1244, 5729
\@@_Vdots .....	1084, 1149, 3357
\g_@@_Vdots_lines_tl .....	1177, 2540
\@@_Vdotsfor: .....	1156, 3574
\@@_Vdotsfor:nnnn .....	3578, 3589
\@@_W: .....	1493, 1571
\@@_actually_color: .....	1248, 3714
\@@_actually_diagbox:nnnnnn .....	4874, 5133, 5537, 5553
\@@_actually_draw_Cdots: .....	2897, 2901
\@@_actually_draw_Ddots: .....	3047, 3051
\@@_actually_draw_Iddots: .....	3099, 3103
\@@_actually_draw_Ldots: .....	2858, 2862, 3565
\@@_actually_draw_Vdots: .....	2979, 2983, 3640
\@@_adapt_S_column: .....	171, 186, 1346
\@@_add_to_colors_seq:nn .....	3701, 3713, 3736, 3751, 3766, 3775
\@@_adjust_pos_of_blocks_seq: .....	2450, 2506
\@@_adjust_pos_of_blocks_seq_i:nnnn .....	2509, 2511
\@@_adjust_size_box: .....	914, 940, 1691, 2225, 2270
\@@_adjust_to_submatrix:nn .....	2742, 2845, 2884, 2965, 3040, 3092
\@@_adjust_to_submatrix:nnnnnn .....	2749, 2751
\@@_after_array: .....	1483, 2356
\g_@@_after_col_zero_bool .....	273, 1050, 2202, 3482
\@@_analyze_end:Nn .....	1999, 2044
\l_@@_argspec_tl .....	3323, 3324, 3325, 3341, 3357, 3373, 3397, 3496, 3497, 3498, 3572, 3573, 3574, 3650, 3651, 3652
\@@_array: .....	1004, 2000, 2027
\@@_arraycolor .....	1241, 3812
\c_@@_arydshln_loaded_bool ...	24, 31, 1599
\l_@@_auto_columns_width_bool .....	470, 606, 2111, 2115, 4583
\l_@@_baseline_tl .....	461, 462, 599, 600, 601, 602, 1017, 1416, 1812, 1824, 1829, 1831, 1836, 1841, 1923, 1924, 1928, 1933, 1935, 1940
\@@_begin_of_NiceMatrix:nn .....	2312, 2333
\@@_begin_of_row: .....	841, 864, 2203
\l_@@_block_auto_columns_width_bool .....	1360, 2116, 4576, 4581, 4591, 4601
\g_@@_block_box_int .....	311, 1340, 4869, 4883, 4885, 4921, 4933, 4945, 4954, 4964
\g_@@_blocks_dp_dim .....	241, 922, 925, 926, 4948, 4951

\g_@@_blocks_ht_dim .....	\@@_computations_for_large_nodes: ...
..... 240, 928, 931, 932, 4939, 4942	..... 4705, 4718, 4723
\g_@@_blocks_seq .....	\@@_computations_for_medium_nodes: ..
..... 287, 1362, 1862, 4958, 4970, 5047	..... 4625, 4694, 4704, 4715
\g_@@_blocks_wd_dim .....	\@@_compute_a_corner:nnnnnn .....
..... 239, 916, 919, 920, 4927, 4930	..... 4390, 4392, 4394, 4396, 4412
\c_@@_booktabs_loaded_bool 25, 34, 1098, 1894	\@@_compute_corners: .....
\l_@@_borders_clist .....	\@@_construct_preamble: .....
..... 301, 5026,	\l_@@_corners_cells_seq .....
5106, 5394, 5410, 5412, 5414, 5416, 5435, 5447	..... 291, 1191, 2455, 3934, 3974, 4123, 4129,
\@@_cartesian_path: .....	4136, 4248, 4254, 4261, 4384, 4400, 4407, 4468
..... 3745, 3760, 3884, 3896, 3989	\l_@@_corners_clist .....
\@@_cartesian_path:n .....	..... 466, 584, 589, 4089, 4215, 4385
\l_@@_cell_box 842, 888, 890, 896, 902, 905,	\@@_create_col_nodes: ....
909, 918, 919, 924, 925, 930, 931, 941, 942,	..... 2003, 2031, 2050
943, 944, 946, 949, 953, 955, 974, 1100,	\@@_create_diag_nodes: ...
1258, 1260, 1681, 1692, 2204, 2228, 2231,	..... 1226, 2426, 2561
2233, 2250, 2273, 2277, 5141, 5245, 5279, 5532	\@@_create_extra_nodes: .....
\l_@@_cell_space_bottom_limit_dim ...	\@@_create_large_nodes: .....
..... 450, 518, 944	\@@_create_medium_and_large_nodes: ..
\l_@@_cell_space_top_limit_dim 449, 516, 942	..... 4620, 4710
\l_@@_cell_type_tl .....	\@@_create_medium_nodes: .....
..... 237, 238, 1613, 1683, 4847, 4849	..... 4621, 4689
\@@_cellcolor .....	\@@_create_nodes: 4696, 4707, 4717, 4720, 4761
..... 1236, 3792, 3804, 3805	\@@_create_row_node: ....
\@@_cellcolor_tabular .....	..... 1020, 1053, 1099
..... 1092, 4014	\@@_cut_on_hyphen:w .....
\g_@@_cells_seq .....	..... 3727, 3782, 3787, 3852, 3944,
..... 2038, 2039, 2040, 2042	3945, 3967, 3968, 3998, 3999, 5312, 5321,
\@@_chessboardcolors .....	5352, 5355, 5399, 5402, 5731, 5736, 5749, 5752
..... 1243, 3797	\g_@@_ddots_int .....
\@@_cline .....	..... 2429, 3071, 3072
..... 135, 1146	\@@_def_env:nnn .....
\@@_cline_i:nn .....	..... 2296, 2307, 2308, 2309, 2310, 2311
..... 136, 137, 149, 152	\@@_define_L_C_R: .....
\@@_cline_i:w .....	..... 222, 1302, 6047
..... 137, 138	\@@_define_com:nnn .....
\l_@@_code_before_bool .....	..... 5502, 5510, 5511, 5512, 5513, 5514
..... 277, 596, 623, 1024, 1186, 1328, 1368,	\@@_delimiter:nnn .....
2058, 2075, 2093, 2124, 2150, 2177, 2402, 2497	..... 1720, 1726, 1746, 1761, 1769, 5613
\l_@@_code_before_tl .....	\l_@@_delimiters_color_tl 481, 720, 723,
..... 276, 595, 1247, 1327, 1369	778, 781, 800, 802, 1439, 1440, 1457, 1458,
\l_@@_code_for_first_col_tl ....	5593, 5651, 5652, 5679, 5983, 5984, 6007, 6008
..... 535, 2215	\l_@@_delimiters_max_width_bool .....
\l_@@_code_for_first_row_tl .	..... 482, 581, 719, 1462
..... 539, 852, 5217	\g_@@_delta_x_one_dim ....
\l_@@_code_for_last_col_tl ....	..... 2431, 3074, 3084
..... 537, 2259	\g_@@_delta_x_two_dim ....
\l_@@_code_for_last_row_tl .	..... 2433, 3122, 3132
..... 541, 859, 5220	\g_@@_delta_y_one_dim ....
\l_@@_code_tl .....	..... 2432, 3076, 3084
..... 268, 5724, 5931	\g_@@_delta_y_two_dim ....
\l_@@_col_max_int .....	..... 2434, 3124, 3132
..... 296, 2609, 2620, 2688, 2747, 2764	\@@_diagbox:nn .....
\l_@@_col_min_int .....	..... 1163, 5533
..... 295, 2614, 2677, 2682, 2745, 2762	\@@_dotfill: .....
\g_@@_col_total_int .....	..... 5522
..... 957,	\@@_dotfill_i: .....
1171, 1400, 2143, 2144, 2180, 2184, 2189,	..... 5527, 5529
2190, 2249, 2360, 2363, 2368, 2375, 2419,	\@@_dotfill_ii: .....
2932, 2950, 3010, 3472, 3473, 3635, 4039,	..... 5526, 5529, 5530
4634, 4644, 4678, 4765, 5059, 5252, 5757, 5806	\@@_dotfill_iii: .....
\l_@@_color_tl .....	..... 5530, 5531
..... 303, 4819, 4887, 4889	\@@_double_int_eval:n ....
\g_@@_colors_seq 1246, 3704, 3708, 3709, 3718	..... 3646, 3660, 3661
\@@_colortbl_like: .....	\g_@@_dp_ante_last_row_dim ....
..... 1090, 1165	..... 867, 1131
\l_@@_colortbl_like_bool 447, 622, 1165, 1518	\g_@@_dp_last_row_dim .....
\c_@@_colortbl_loaded_bool ...	..... 867, 868, 1134, 1135, 1259, 1260, 1433
..... 88, 92, 1115	\g_@@_dp_row_zero_dim .....
\l_@@_cols_tl .....	..... 887, 888, 1125, 1126, 1426, 1917, 1956
..... 3744, 3759, 3789, 3823, 3831, 3832, 3937, 3940	\@@_draw_Cdots:nnn .....
\g_@@_cols_vlism_seq ..	..... 2882
..... 256, 1511, 1590, 5834	\@@_draw_Ddots:nnn .....
\@@_columncolor .....	..... 3038
..... 1242, 3747	\@@_draw_Iddots:nnn .....
\@@_columncolor:n .....	..... 3090
..... 3753, 3756	\@@_draw_Ldots:nnn .....
\@@_columncolor_preamble .....	..... 2843
..... 1094, 4037	\@@_draw_Vdots:nnn .....
\c_@@_columncolor_regex .....	..... 2963
..... 210, 1521	\@@_draw_blocks: .....
\l_@@_columns_width_dim .....	..... 1862, 5044
..... 236, 607, 744, 2112, 2118, 4589, 4595	\@@_draw_dotted_lines: .....
\g_@@_com_or_env_str .....	..... 2448, 2528
..... 260, 263	\@@_draw_dotted_lines_i: .....
	..... 2531, 2535

\l_@@_draw_first_bool .	309, 3388, 3412, 3423	\l_@@_extra_left_margin_dim .....	476, 573, 1319, 2236
@@_draw_hlines: .....	2461, 4308	\l_@@_extra_right_margin_dim .....	477, 574, 1387, 2281, 3013
@@_draw_line: .....	2880,	@@_extract_brackets .....	5118, 5284
	2925, 3036, 3088, 3136, 3138, 3699, 4544, 4574	@@_extract_coords_values: ....	4786, 4793
@@_draw_line_ii:nn .....	3679, 3683	@@_fatal:n ....	15, 252, 1350, 1707, 1742,
@@_draw_line_iii:nn .....	3686, 3690		2008, 2012, 2014, 2047, 3487, 6129, 6132, 6135
@@_draw_non_standard_dotted_line: ..		@@_fatal:nn .....	16, 1604
	3144, 3146	\l_@@_fill_tl .....	299, 5018, 5116, 5118
@@_draw_non_standard_dotted_line:n ..		\l_@@_final_i_int .....	
	3149, 3152		2438, 2596, 2601, 2604, 2629,
@@_draw_non_standard_dotted_line:nnn			2637, 2641, 2650, 2658, 2738, 2793, 2874,
	3154, 3159, 3173		2947, 2953, 2956, 3519, 3547, 3615, 3625, 3627
@@_draw_standard_dotted_line: .	3143, 3174	\l_@@_final_j_int .....	
@@_draw_standard_dotted_line_i: 3237, 3241			2439, 2597, 2602, 2609, 2614, 2620, 2630,
\l_@@_draw_tl .....	300, 5020,		2638, 2642, 2651, 2659, 2739, 2794, 2827,
	5024, 5084, 5294, 5300, 5302, 5304, 5341, 5342		2830, 2838, 3064, 3540, 3550, 3552, 3594, 3623
@@_draw_vlines: .....	2462, 4185	\l_@@_final_open_bool .....	2441, 2603,
\g_@@_empty_cell_bool ....	284, 948, 958,		2607, 2610, 2617, 2623, 2627, 2643, 2871,
	2238, 2285, 3339, 3355, 3371, 3395, 3418, 3429		2906, 2911, 2922, 2986, 2996, 3001, 3022,
@@_end_Cell: .....	203, 935, 1615,		3061, 3111, 3245, 3260, 3291, 3292, 3517,
	1670, 1687, 2323, 3441, 3442, 3443, 3444,		3541, 3553, 3592, 3616, 3628, 3669, 4512, 4549
	3445, 3446, 3447, 3448, 3449, 3450, 3451, 3452	@@_find_extremities_of_line:nnnn ..	
\l_@@_end_of_row_tl .....			2591, 2848, 2887, 2968, 3043, 3095
	478, 479, 529, 2023, 2024, 6237	\l_@@_first_col_int .....	123, 136, 321,
\c_@@_endpgfortikzpicture_tl .....			322, 531, 810, 841, 1409, 1525, 2053, 2073,
	43, 47, 2532, 3687, 4515		2413, 2932, 2950, 3480, 3958, 4051, 4634,
\c_@@_enumitem_loaded_bool .....			4644, 4678, 4726, 4765, 5484, 5490, 5496, 5806
	26, 37, 340, 650, 655, 666, 671	\l_@@_first_i_tl .....	5745, 5750, 5769, 5800,
@@_env: ....	230, 234, 873, 879, 975, 981,		5809, 5811, 5866, 5873, 5875, 5956, 5967, 5971
	1029, 1035, 1041, 1206, 1207, 1213, 1214,	\l_@@_first_j_tl .....	5746, 5751, 5773,
	1221, 1222, 1233, 2059, 2062, 2064, 2080,		5775, 5836, 5849, 5856, 5858, 5957, 5968, 5972
	2086, 2089, 2098, 2104, 2107, 2129, 2135,	\l_@@_first_row_int .....	319, 320,
	2138, 2155, 2161, 2167, 2180, 2184, 2190,		532, 814, 1169, 1424, 1843, 1914, 1942,
	2472, 2578, 2585, 2587, 2588, 2649, 2717,		1953, 2411, 2802, 2824, 4627, 4641, 4668,
	2781, 2792, 2805, 2808, 2827, 2830, 2935,		4725, 4763, 5160, 5178, 5482, 5625, 5770, 6255
	2938, 2953, 2956, 3528, 3546, 3603, 3621,	\c_@@_footnote_bool .....	
	3672, 3674, 3693, 3696, 4423, 4442, 4460,		1335, 1485, 6024, 6060, 6083, 6086, 6096, 6102
	4647, 4649, 4657, 4768, 4777, 4795, 5156,	\c_@@_footnotehyper_bool .	6023, 6061, 6093
	5163, 5167, 5181, 5186, 5198, 5203, 5204,	@@_full_name_env: .....	
	5207, 5224, 5258, 5628, 5631, 5773, 5775,		261, 6108, 6145, 6152, 6160, 6168, 6172,
	5780, 5782, 5809, 5811, 5816, 5818, 5916, 5927		6269, 6282, 6285, 6298, 6303, 6308, 6310,
\g_@@_env_int .....	229,		6334, 6353, 6358, 6363, 6370, 6396, 6405, 6576
	230, 232, 1189, 1192, 1194, 1198, 1201,	@@_hdottedline: .....	1152, 4497
	1210, 1211, 1218, 1219, 1268, 1271, 1286,	@@_hdottedline:n .....	4505, 4509
	1289, 1359, 1366, 1370, 2367, 2388, 2406,	@@_hdottedline_i: .....	4500, 4502
	2409, 2422, 2493, 3839, 3842, 3867, 4406, 4804	@@_hdottedline_i:n .....	4514, 4518
@@_error:n ...	12, 343, 368, 491, 501, 551,	@@_hline:nn .....	4196, 4316, 4335
	676, 724, 727, 737, 743, 752, 760, 782, 784,	@@_hline_i:nn .....	2459, 4199, 4202
	791, 801, 804, 805, 806, 812, 817, 818, 819,	@@_hline_i_complete:nn .....	2459, 4306
	830, 832, 833, 834, 1395, 1405, 1477, 1846,	@@_hline_ii:nnnn ...	4224, 4235, 4268, 4307
	1899, 1945, 3811, 3826, 5042, 5392, 5597,	\l_@@_hlines_clist .....	315, 543,
	5709, 5720, 5727, 5758, 6057, 6062, 6089, 6099		557, 586, 1054, 1056, 1060, 2461, 4314, 4315
@@_error:nn .....	13, 614, 1579, 1580,	\l_@@_hpos_of_block_tl ..	305, 306, 4809,
	1581, 1734, 1747, 3328, 3331, 3344, 3347,		4811, 4813, 4848, 4849, 4851, 4894, 4900,
	3360, 3363, 3377, 3378, 3383, 3384, 3401,		4910, 4961, 4984, 4988, 5000, 5005, 5032,
	3402, 3407, 3408, 4398, 5397, 5714, 5788, 5791		5034, 5036, 5226, 5238, 5249, 5253, 5260, 5272
@@_error:nnn .....	14, 3677, 5863, 5898	\g_@@_ht_last_row_dim .....	
@@_error_too_much_cols: .....	1552, 6124		869, 1132, 1133, 1257, 1258, 1432
@@_everycr: .....	1046, 1120, 1123	\g_@@_ht_row_one_dim ..	895, 896, 1129, 1130
@@_everycr_i: .....	1046, 1047	\g_@@_ht_row_zero_dim .....	
@@_expand_clist:NN .....	3937, 3938, 3990		889, 890, 1127, 1128, 1427, 1916, 1955
\l_@@_exterior_arraycolsep_bool .....			
	463, 740, 1533, 1545		



<code>\@@_hvlines_block:nnn</code> .....	5100, 5348	<code>\l_@@_last_i_tl</code> .....	5747,
<code>\l_@@_hvlines_block_bool</code> ..	310, 5028, 5096		5753, 5756, 5769, 5803, 5816, 5818, 5866, 5873
<code>\@@_i:</code> .....	4627, 4629,	<code>\l_@@_last_j_tl</code> .....	
	4630, 4631, 4632, 4641, 4647, 4649, 4650,		5748, 5754, 5757, 5780, 5782, 5839, 5849, 5856
	4651, 4652, 4657, 4658, 4659, 4660, 4668,	<code>\l_@@_last_row_int</code> .....	
	4671, 4673, 4674, 4675, 4727, 4729, 4732,		323, 324, 533, 857, 903, 1066, 1202, 1253,
	4733, 4737, 4738, 4763, 4768, 4770, 4772,		1263, 1270, 1277, 1389, 1393, 1396, 1408,
	4776, 4777, 4788, 4795, 4797, 4799, 4803, 4804		1430, 2025, 2026, 2211, 2212, 2256, 2257,
<code>\g_@@_iddots_int</code> .....	2430, 3119, 3120		2382, 2853, 2892, 3362, 3378, 3402, 4057,
<code>\l_@@_in_env_bool</code> ....	243, 367, 1350, 1351		4065, 5051, 5054, 5055, 5074, 5091, 5103,
<code>\c_@@_in_preamble_bool</code> .	21, 22, 23, 646, 662		5113, 5125, 5136, 5150, 5208, 5219, 5494, 6395
<code>\l_@@_initial_i_int</code> .....	2436,	<code>\l_@@_last_row_without_value_bool</code> ...	
	2594, 2669, 2672, 2697, 2705, 2709, 2718,		..... 325, 1265, 1391, 2380
	2726, 2736, 2782, 2867, 2913, 2915, 2929,	<code>\l_@@_left_delim_dim</code> .....	
	2935, 2938, 3518, 3519, 3529, 3597, 3607, 3609		..... 1304, 1308, 1313, 1991, 2234
<code>\l_@@_initial_j_int</code> .....		<code>\g_@@_left_delim_tl</code> .....	
	..... 2437, 2595, 2670, 2677,		..... 1312, 1337, 1441, 1465, 1712, 4539
	2682, 2688, 2698, 2706, 2710, 2719, 2727,	<code>\l_@@_left_margin_dim</code> .....	
	2737, 2783, 2805, 2808, 2816, 3003, 3005,		..... 474, 567, 1318, 2235, 4530, 4756
	3010, 3056, 3522, 3532, 3534, 3593, 3594, 3605	<code>\l_@@_letter_for_dotted_lines_str</code> ...	
<code>\l_@@_initial_open_bool</code> .....			..... 751, 762, 763, 1585
	..... 2440, 2671, 2675, 2678, 2685, 2691,	<code>\l_@@_letter_vlism_tl</code> ....	255, 550, 1588
	2695, 2711, 2864, 2903, 2910, 2920, 2986,	<code>\l_@@_light_syntax_bool</code> .....	
	2993, 2999, 3053, 3105, 3243, 3290, 3516,		..... 460, 527, 1321, 1382, 2383
	3523, 3535, 3591, 3598, 3610, 3668, 4511, 4548	<code>\@@_light_syntax_i</code> .....	2016, 2019
<code>\@@_insert_tabularnotes:</code> .....	1866, 1869	<code>\@@_line</code> .....	2476, 3652
<code>\@@_instruction_of_type:nnn</code> .....		<code>\@@_line_i:nn</code> .....	3659, 3666
	..... 985, 3333, 3349, 3365, 3388, 3412	<code>\l_@@_line_width_dim</code> .....	
<code>\c_@@_integers_alist_tl</code> .....	5941, 5952		304, 5030, 5295, 5333, 5344, 5350, 5362,
<code>\l_@@_inter_dots_dim</code> .....			5389, 5409, 5424, 5426, 5436, 5437, 5439, 5450
	. 451, 452, 2445, 3248, 3255, 3266, 3274,	<code>\@@_line_with_light_syntax:n</code> ...	2030, 2034
	3281, 3286, 3298, 3306, 4540, 4542, 4570, 4572	<code>\@@_line_with_light_syntax_i:n</code> .....	
<code>\g_@@_internal_code_after_tl</code> ....	269,		..... 2029, 2035, 2036
	1649, 1719, 1725, 1745, 1760, 1768, 1778,	<code>\@@_math_toggle_token:</code> .....	
	2477, 2478, 4334, 4504, 4872, 5131, 5535, 5739		155, 937, 2205, 2222, 2251, 2267, 5577, 5581
<code>\@@_intersect_our_row:nnnn</code> .....	3920	<code>\g_@@_max_cell_width_dim</code> .....	
<code>\@@_intersect_our_row_p:nnnn</code> .....	3871		..... 945, 946, 1361, 2117, 4582, 4608
<code>\@@_j:</code> .....	4634, 4636,	<code>\c_@@_max_l_dim</code> .....	3235, 3240
	4637, 4638, 4639, 4644, 4647, 4649, 4652,	<code>\l_@@_medium_nodes_bool</code>	472, 563, 4617, 5200
	4654, 4655, 4657, 4660, 4662, 4663, 4678,	<code>\@@_message_hdotsfor:</code>	6137, 6145, 6152, 6160
	4681, 4683, 4684, 4685, 4740, 4742, 4745,	<code>\@@_msg_new:nn</code> .....	
	4747, 4751, 4752, 4765, 4768, 4769, 4771,		. 17, 6039, 6065, 6074, 6104, 6111, 6142,
	4776, 4777, 4789, 4795, 4796, 4798, 4803, 4804		6149, 6157, 6165, 6170, 6176, 6181, 6186,
<code>\l_@@_l_dim</code> .....			6191, 6196, 6201, 6206, 6211, 6217, 6223,
	..... 3221, 3222, 3235, 3236, 3248, 3254,		6228, 6234, 6240, 6245, 6252, 6259, 6266,
	3265, 3273, 3281, 3286, 3298, 3299, 3306, 3307		6272, 6281, 6283, 6289, 6306, 6313, 6318,
<code>\l_@@_large_nodes_bool</code>	473, 564, 4619, 4623		6325, 6332, 6339, 6346, 6351, 6361, 6367,
<code>\g_@@_last_col_found_bool</code> ..	329, 1174,		6374, 6381, 6387, 6393, 6401, 6403, 6409, 6688
	1401, 1469, 2142, 2171, 2247, 2359, 2416, 5250	<code>\@@_msg_new:nnn</code> ...	18, 6025, 6295, 6414,
<code>\l_@@_last_col_int</code> .....	327,		6424, 6439, 6460, 6478, 6521, 6573, 6624, 6674
	328, 728, 771, 773, 792, 813, 831, 1204,	<code>\@@_msg_redirect_name:nn</code> .....	
	1282, 1288, 1295, 1404, 1537, 2319, 2321,		..... 19, 746, 1481, 5065, 5068
	2360, 2363, 2415, 2973, 3008, 3330, 3346,	<code>\@@_multicolumn:nnn</code> .....	1157, 3433
	3384, 3408, 5052, 5057, 5058, 5059, 5062,	<code>\g_@@_multicolumn_cells_seq</code> .....	
	5091, 5103, 5113, 5125, 5137, 5152, 5181,		... 1167, 3460, 4652, 4660, 4782, 5165, 5183
	5186, 5194, 5209, 5486, 5492, 5498, 6128, 6146	<code>\g_@@_multicolumn_sizes_seq</code>	1168, 3462, 4783
<code>\l_@@_last_col_without_value_bool</code> ...		<code>\g_@@_name_env_str</code> .....	259,
	..... 326, 770, 2361, 6131		264, 265, 1344, 1345, 2046, 2291, 2292,
<code>\l_@@_last_empty_column_int</code> .....			2300, 2301, 2330, 2341, 2349, 2500, 5506, 6126
	..... 4433, 4434, 4447, 4453, 4466	<code>\l_@@_name_str</code> ..	471, 616, 875, 878, 977,
<code>\l_@@_last_empty_row_int</code> .....			980, 1037, 1040, 1266, 1275, 1278, 1284,
	..... 4415, 4416, 4429, 4450		1293, 1296, 2063, 2064, 2088, 2089, 2106,

2107, 2137, 2138, 2163, 2166, 2186, 2189, 2370, 2374, 2391, 2395, 4773, 4776, 4800, 4803	
\g_@@_names_seq .....	242, 613, 615, 6686
\l_@@_nb_cols_int .....	5473, 5478, 5481, 5485, 5491, 5497
\l_@@_nb_rows_int .....	5472, 5477, 5488
\@@_newcolumnmtype .....	1073, 1492, 1493
\@@_node_for_multicolumn:nn .....	4784, 4791
\@@_node_for_the_cell: .....	954, 961, 2232, 2282
\@@_node_left:nn .....	5915, 5916, 5975
\@@_node_position: ..	1213, 1215, 1221, 1223
\@@_node_right:nn .....	5925, 5927, 5999
\g_@@_not_empty_cell_bool .....	275, 952, 959, 2338
\@@_not_in_exterior:nnnn .....	3912
\@@_not_in_exterior_p:nnnn .....	3844
\l_@@_notes_above_space_dim .....	467, 468
\l_@@_notes_bottomrule_bool .....	634, 795, 825, 1892
\l_@@_notes_code_after_tl .....	632, 1901
\l_@@_notes_code_before_tl .....	630, 1873
\@@_notes_label_in_list:n .....	336, 355, 363, 642
\@@_notes_label_in_tabular:n ..	335, 376, 639
\l_@@_notes_para_bool ..	628, 793, 823, 1877
\@@_notes_style:n .....	334, 337, 355, 363, 379, 384, 636
\l_@@_nullify_dots_bool .....	469, 562, 3337, 3353, 3369, 3393, 3416
\l_@@_number_of_notes_int .....	333, 370, 380, 390
\@@_old_CT@arc@ .....	1352, 2502
\@@_old_cdots .....	1140, 3354
\@@_old_ddots .....	1142, 3394
\@@_old_dotfill .....	5521, 5524, 5532
\@@_old_dotfill: .....	1161
\l_@@_old_iRow_int .....	270, 1102, 2548
\@@_old_ialign: .....	1019, 1136, 5046
\@@_old_iddots .....	1143, 3417
\l_@@_old_jCol_int .....	271, 1105, 2549
\@@_old_ldots .....	1139, 3338
\@@_old_multicolumn .....	3432, 3436
\@@_old_pgfpntanchor .....	163, 5933, 5937
\@@_old_pgful@check@rerun .....	81, 85
\@@_old_vdots .....	1141, 3370
\@@_open_x_final_dim: .....	2821, 2873, 2907, 3065, 3114
\@@_open_x_initial_dim: .....	2799, 2866, 2904, 3058, 3108
\@@_open_y_final_dim: .....	2945, 2997, 3063, 3113
\@@_open_y_initial_dim: .....	2927, 2994, 3055, 3107
\l_@@_parallelize_diags_bool .....	464, 465, 559, 2427, 3069, 3117
\@@_patch_preamble:n .....	1514, 1556, 1594, 1602, 1623, 1652, 1715, 1735, 1737, 1748, 1762, 1770, 1780, 1800
\@@_patch_preamble_i:n .....	1560, 1561, 1562, 1609
\@@_patch_preamble_ii:nn .....	1563, 1564, 1565, 1620
\@@_patch_preamble_iii:n ..	1566, 1625, 1633
\@@_patch_preamble_iii_i:n .....	1628, 1630
\@@_patch_preamble_iv:nnn .....	1567, 1568, 1569, 1655
\@@_patch_preamble_ix:n .....	1785, 1803
\@@_patch_preamble_v:nnnn .....	1570, 1571, 1676
\@@_patch_preamble_vi:n .....	1572, 1698
\@@_patch_preamble_vii:n .....	1573, 1574, 1575, 1704
\@@_patch_preamble_vii_i:n .....	1721, 1727, 1730
\@@_patch_preamble_viii:nn .....	1576, 1577, 1578, 1739
\@@_patch_preamble_x:n .....	1618, 1674, 1696, 1702, 1782, 1806
\@@_patch_preamble_xi:n .....	1586, 1774
\@@_pgf_rect_node:nnn .....	423, 5202
\@@_pgf_rect_node:nnnn .....	398, 4767, 4794, 5155, 5197, 5902
\c_@@_pgfortikzpicture_tl .....	42, 46, 2530, 3685, 4513
\@@_pgfpntanchor:n .....	5929, 5934
\@@_pgfpntanchor_i:nn .....	5937, 5939
\@@_pgfpntanchor_ii:w .....	5940, 5948
\@@_pgfpntanchor_iii:w .....	5961, 5963
\@@_picture_position: .....	1207, 1215, 1223
\g_@@_pos_of_blocks_seq .....	288, 1363, 2423, 2453, 2508, 3463, 4083, 4209, 4480, 5081, 5545
\g_@@_pos_of_stroken_blocks_seq .....	290, 1364, 4087, 4213, 5093
\g_@@_pos_of_xdots_seq .....	289, 1365, 2454, 2734, 4085, 4211
\@@_pre_array: .....	1183, 1329, 1379
\@@_pre_array_i:w .....	1325, 1379
\@@_pre_array_ii: .....	1096, 1300
\c_@@_preamble_first_col_tl .....	1526, 2197
\c_@@_preamble_last_col_tl .....	1538, 2242
\g_@@_preamble_tl .....	1339, 1494, 1498, 1501, 1507, 1523, 1526, 1535, 1538, 1547, 1551, 1592, 1601, 1611, 1622, 1635, 1657, 1678, 1700, 1718, 1759, 1767, 1776, 1789, 1796, 1805, 2000, 2027
\@@_pred:n .....	124, 154, 2321, 4124, 4137, 4249, 4262
\@@_provide_pgfsyspdfmark: ..	211, 220, 1334
\@@_put_box_in_flow: .....	1467, 1808, 1993
\@@_put_box_in_flow_bis:nn .....	1464, 1960
\@@_put_box_in_flow_i: .....	1814, 1816
\@@_qpnt:n .....	233, 1819, 1821, 1833, 1849, 1908, 1910, 1926, 1937, 1948, 2567, 2569, 2571, 2573, 2581, 2583, 2816, 2838, 2867, 2874, 2913, 2915, 2929, 2947, 3003, 3005, 3056, 3064, 3693, 3696, 3957, 3961, 3977, 3979, 4147, 4149, 4151, 4272, 4274, 4276, 4521, 4525, 4532, 4566, 4569, 4571, 4673, 4683, 5146, 5148, 5150, 5152, 5174, 5194, 5222, 5317, 5319, 5326, 5328, 5363, 5365, 5369, 5374, 5376, 5380, 5423, 5425, 5427, 5434, 5438, 5440, 5558, 5560, 5563, 5565, 5618, 5620, 5800, 5803, 5841, 5858, 5875
\l_@@_radius_dim .....	455, 456, 1777, 2444, 2878, 2879, 3315, 4499, 4523, 4567, 4568
\l_@@_real_left_delim_dim .....	1962, 1977, 1992
\l_@@_real_right_delim_dim .....	1963, 1989, 1995
\@@_rectanglecolor ..	1237, 3762, 3795, 3815
\@@_rectanglecolor:nnn ..	3768, 3777, 3780
\@@_renew_NC@rewrite@S: .....	192, 194, 1173
\@@_renew_dots: .....	1080, 1166
\l_@@_renew_dots_bool .....	560, 736, 1166, 6049, 6056

\@@_renew_matrix:	731, 735, 5452, 6051, 6055	\@@_stroke_horizontal:n	5415, 5417, 5432
\l_@@_respect_blocks_bool	3821, 3836, 3864	\@@_stroke_vertical:n	5411, 5413, 5421
\@@_restore_iRow_jCol:	2501, 2546	\@@_sub_matrix_i:nn	5792, 5798
\@@_revtex_array:	996, 1007	\l_@@_submatrix_extra_height_dim	312, 5671, 5826
\c_@@_revtex_bool	50, 52, 55, 57, 1006	\l_@@_submatrix_hlines_clist	317, 5683, 5701, 5865, 5867
\l_@@_right_delim_dim	1305, 1309, 1315, 1994, 2279	\l_@@_submatrix_left_xshift_dim	313, 5673, 5878, 5911
\g_@@_right_delim_tl	1314, 1338, 1459, 1465, 1713, 1755, 4541	\l_@@_submatrix_name_str	5716, 5760, 5900, 5902, 5914, 5916, 5924, 5927
\l_@@_right_margin_dim	475, 569, 1386, 2280, 3012, 4537, 4759	\g_@@_submatrix_names_seq	292, 2480, 5713, 5717, 6304
\@@_rotate:	1159, 3645	\l_@@_submatrix_right_xshift_dim	314, 5675, 5887, 5921
\g_@@_rotate_bool	248, 912, 939, 1690, 2224, 2269, 3645, 4894, 4918, 4923, 4983, 4999, 5142	\g_@@_submatrix_seq	297, 1185, 2748, 5737
\@@_rotate_cell_box:	900, 939, 1690, 2224, 2269, 5142	\l_@@_submatrix_slim_bool	5681, 5768, 6277
\l_@@_rounded_corners_dim	302, 5022, 5126, 5309, 5310, 5345, 5391, 5448	\l_@@_submatrix_vlines_clist	318, 5685, 5703, 5848, 5850
\@@_roundedrectanglecolor	1238, 3771	\@@_succ:n	149, 153, 1029, 1035, 1061, 1650, 1720, 1726, 1794, 1821, 2155, 2161, 2166, 2167, 2180, 2184, 2189, 2190, 2417, 2838, 2915, 3005, 3064, 3916, 3961, 3977, 4120, 4151, 4189, 4245, 4276, 4312, 4335, 4485, 4487, 4489, 4491, 4532, 4571, 4733, 4737, 4747, 4751, 5150, 5152, 5194, 5326, 5328, 5563, 5565, 5620
\l_@@_row_max_int	294, 2604, 2746, 2763	\l_@@_suffix_tl	4695, 4706, 4716, 4719, 4768, 4776, 4777, 4795, 4803, 4804
\l_@@_row_min_int	293, 2672, 2744, 2761	\c_@@_table_collect_begin_tl	181, 183, 201
\g_@@_row_of_col_done_bool	274, 1051, 1343, 2072	\c_@@_table_print_tl	184, 185, 203
\g_@@_row_total_int	1170, 1407, 1844, 1943, 2382, 2389, 2396, 2412, 2802, 2824, 3560, 4627, 4641, 4668, 4763, 5074, 5160, 5178, 5625, 5756, 5770, 6256	\l_@@_tabular_width_dim	246, 1012, 1014, 1549, 2350
\@@_rowcolor	1239, 3732	\l_@@_tabularnote_tl	332, 797, 827, 1865, 1874
\@@_rowcolor:n	3738, 3741	\g_@@_tabularnotes_seq	331, 371, 1880, 1886, 1902
\@@_rowcolor_tabular	1093, 4025	\@@_test_hline_in_block:nnnn	4210, 4212, 4338
\@@_rowcolors	1240, 3828	\@@_test_hline_in_stroken_block:nnnn	4214, 4360
\@@_rowcolors_i:nnnn	3872, 3907	\@@_test_if_cell_in_a_block:nn	4419, 4437, 4455, 4475
\l_@@_rowcolors_restart_bool	3824, 3855	\@@_test_if_cell_in_block:nnnnnnn	4481, 4483
\g_@@_rows_seq	2022, 2024, 2026, 2028, 2030	\@@_test_if_math_mode:	249, 1349, 2302
\l_@@_rows_tl	3743, 3758, 3788, 3874, 3938, 3963	\@@_test_in_corner_h:	4215, 4243
\l_@@_rules_color_tl	272, 505, 1376, 1377, 5831, 5832	\@@_test_in_corner_v:	4090, 4118
\@@_set_CT@arc:	157, 1377, 5832	\@@_test_vline_in_block:nnnn	4084, 4086, 4349
\@@_set_CT@arc_i:	158, 159	\@@_test_vline_in_stroken_block:nnnn	4088, 4371
\@@_set_CT@arc_ii:	158, 161	\l_@@_the_array_box	1301, 1317, 1859, 1860
\@@_set_final_coords:	2772, 2797	\c_@@_tikz_loaded_bool	27, 41, 1228, 2464, 4550
\@@_set_final_coords_from_anchor:n	2788, 2877, 2908, 2989, 2998, 3068, 3116	\@@_true_c:	202, 1572
\@@_set_initial_coords:	2767, 2786	\l_@@_type_of_col_tl	774, 775, 2331, 2333
\@@_set_initial_coords_from_anchor:n	2777, 2870, 2905, 2988, 2995, 3060, 3110	\c_@@_types_of_matrix_seq	6116, 6117, 6122, 6126
\@@_set_size:n	5470, 5479	\@@_update_for_first_and_last_row:	883, 947, 1255, 2226, 2271
\c_@@_siunitx_loaded_bool	164, 168, 173, 191	\@@_use_arraybox_with_notes:	1421, 1921
\l_@@_small_bool	729, 776, 788, 815, 846, 1108, 1706, 1741, 2206, 2252, 2442	\@@_use_arraybox_with_notes_b:	1418, 1905
\@@_standard_cline	120, 1145		
\@@_standard_cline:w	120, 121		
\l_@@_standard_cline_bool	448, 514, 1144		
\c_@@_standard_tl	458, 459, 3142, 4543, 4573		
\g_@@_static_num_of_col_int	298, 1476, 1516, 5062, 6161, 6173, 6354		
\l_@@_stop_loop_bool	2598, 2599, 2631, 2644, 2653, 2666, 2667, 2699, 2712, 2721		
\@@_store_in_tmpb_tl	5287, 5289		
\@@_stroke_block:nnn	5088, 5291		
\@@_stroke_borders_block:nnn	5110, 5387		

\@@_use_arraybox_with_notes_c: .....	\} .....
..... 1419, 1450, 1857, 1919, 1958	265, 1578, 1732,
\@@_vdottedline:n .....	1743, 2309, 5514, 5892, 6291, 6371, 6524, 6627
\@@_vdottedline_i:n .....	\l .....
4553, 4558, 4562	2311, 5513
\@@_vline:nn .....	
1650, 4067, 4193	\sq ..
\@@_vline_i:nn .....	6107, 6140, 6145, 6152, 6160, 6161, 6172,
2458, 4072, 4076	6173, 6230, 6255, 6256, 6269, 6275, 6279,
\@@_vline_i_complete:nn .....	6282, 6285, 6297, 6303, 6315, 6353, 6354,
2458, 4183	6355, 6363, 6369, 6376, 6389, 6396, 6397, 6405
\@@_vline_ii:nnnn ...	
4099, 4110, 4143, 4184	
\l_@@_vlines_clist .....	
..... 316, 544, 556, 585, 1499,	
1505, 1532, 1544, 1787, 1794, 2462, 4191, 4192	
\l_@@_vpos_of_block_tl .....	
307, 308,	
4815, 4817, 4899, 4909, 4987, 5004, 5038, 5040	
\@@_w: .....	
1492, 1570	
\g_@@_width_first_col_dim .....	
..... 286, 1342, 1412, 2067, 2227, 2228	
\g_@@_width_last_col_dim .....	
..... 285, 1341, 1471, 2176, 2272, 2273	
\l_@@_x_final_dim .....	
280, 2774,	
2823, 2832, 2833, 2836, 2839, 2840, 2991,	
3007, 3015, 3019, 3023, 3025, 3030, 3032,	
3066, 3075, 3083, 3123, 3131, 3170, 3185,	
3194, 3228, 3280, 3296, 3697, 4533, 4542,	
4568, 5767, 5783, 5784, 5790, 5887, 5904, 5921	
\l_@@_x_initial_dim .....	
..... 278, 2769, 2801, 2810, 2811, 2814, 2817,	
2818, 2991, 3006, 3007, 3014, 3019, 3023,	
3025, 3027, 3030, 3032, 3057, 3075, 3083,	
3123, 3131, 3167, 3184, 3194, 3228, 3280,	
3294, 3296, 3314, 3316, 3694, 4526, 4540,	
4567, 5766, 5776, 5777, 5787, 5878, 5903, 5911	
\l_@@_xdots_color_tl .....	
480, 494, 2857, 2896,	
2977, 2978, 3046, 3098, 3150, 3564, 3639, 3656	
\l_@@_xdots_down_tl ...	
498, 3157, 3178, 3213	
\l_@@_xdots_line_style_tl .....	
..... 457, 459, 490, 3142, 3150, 4543, 4573	
\l_@@_xdots_shorten_dim .....	
453,	
454, 496, 2446, 3164, 3165, 3254, 3265, 3273	
\l_@@_xdots_up_tl ...	
499, 3156, 3177, 3203	
\l_@@_y_final_dim .....	
..... 281, 2775, 2875, 2879, 2917, 2921,	
2923, 2948, 2958, 2959, 3077, 3080, 3125,	
3128, 3170, 3185, 3193, 3230, 3285, 3304,	
3698, 4524, 4572, 5621, 5643, 5658, 5804,	
5819, 5820, 5825, 5843, 5860, 5904, 5912, 5922	
\l_@@_y_initial_dim .....	
..... 279, 2770, 2868, 2878, 2916,	
2917, 2921, 2923, 2930, 2940, 2941, 3077,	
3082, 3125, 3130, 3167, 3184, 3193, 3230,	
3285, 3302, 3304, 3314, 3317, 3695, 4522,	
4523, 4524, 4570, 5619, 5643, 5658, 5801,	
5812, 5813, 5825, 5842, 5859, 5903, 5912, 5922	
\ \ .....	
2013, 2035, 5486, 5492,	
5498, 6027, 6028, 6071, 6080, 6108, 6113,	
6173, 6178, 6183, 6188, 6193, 6231, 6236,	
6242, 6249, 6256, 6262, 6263, 6278, 6286,	
6292, 6298, 6299, 6310, 6322, 6329, 6335,	
6342, 6349, 6358, 6364, 6371, 6378, 6384,	
6390, 6402, 6406, 6411, 6417, 6426, 6427,	
6441, 6442, 6458, 6462, 6463, 6481, 6482,	
6524, 6525, 6576, 6577, 6627, 6628, 6681, 6682	
\{ .....	
265, 1575, 1732,	
1766, 2309, 5514, 5883, 6291, 6371, 6524, 6627	

## A

\A .....	5711
\aboverulesep .....	1896
\addtocounter .....	388
\alpha .....	334
\anchor .....	2558, 2559
\arraybackslash .....	1663
\arraycolor .....	1241, 6107
\arraycolsep .....	568, 570, 572,
1011, 1111, 1308, 1309, 1449, 1453, 4529, 4536	
\arrayrulecolor .....	95
\arrayrulewidth .....	128,
133, 145, 507, 874, 1028, 1030, 1036, 1067,	
1502, 1508, 1593, 1643, 1790, 1797, 1913,	
1952, 2079, 2081, 2087, 2097, 2099, 2105,	
2128, 2130, 2136, 2154, 2156, 2162, 3959,	
3960, 3962, 3978, 3980, 4159, 4160, 4162,	
4173, 4179, 4285, 4296, 4302, 4331, 4608,	
5295, 5350, 5375, 5377, 5389, 5643, 5826, 5830	
\arraystretch .....	
1110, 2931, 2949, 4891, 4980, 4996, 5802, 5805	
\AtBeginDocument .....	23, 28, 73,
89, 165, 189, 338, 452, 454, 456, 468, 648,	
664, 2526, 3321, 3494, 3570, 3648, 3681, 4507	
\AutoNiceMatrix .....	5515
\AutoNiceMatrixWithDelims ...	5475, 5507, 5518

## B

\baselineskip .....	98, 105
\bgroup .....	1336
\bigskip .....	1378
\Block .....	1158, 6328, 6376, 6417
\BNiceMatrix .....	5467
\bNiceMatrix .....	5464
\Body .....	1325
bool commands:	
\bool_do_until:Nn .....	2599, 2667
\bool_gset_false: .....	5519
\bool_gset_false:N .....	
.. 912, 958, 959, 1050, 1174, 1343, 1515,	
2238, 2285, 2499, 4347, 4358, 4369, 4380, 4923	
\bool_gset_true:N .....	
2072, 2202, 2247, 2290, 2338, 3339, 3355,	
3371, 3395, 3418, 3429, 3645, 4082, 4208, 5517	
\bool_if:NTF .....	156, 173,
650, 655, 666, 671, 843, 846, 939, 1024,	
1051, 1098, 1108, 1165, 1166, 1186, 1228,	
1245, 1249, 1302, 1335, 1350, 1360, 1391,	
1469, 1474, 1485, 1490, 1515, 1518, 1690,	
1706, 1741, 1892, 2058, 2075, 2093, 2111,	
2124, 2150, 2171, 2177, 2206, 2224, 2252,	
2269, 2359, 2361, 2380, 2383, 2402, 2427,	
2442, 2464, 2920, 2922, 3069, 3117, 3337,	
3353, 3369, 3393, 3416, 4428, 4446, 4464,	

4591, 4601, 4623, 4894, 4918, 4983, 4999,  
5096, 5142, 5200, 5525, 6083, 6093, 6131, 6277

\bool\_if:nTF ..... 191,  
340, 367, 987, 1401, 2753, 3670, 3922,  
4617, 5250, 5622, 5632, 5634, 5647, 5653, 5662

\bool\_lazy\_all:nTF .....  
... 1528, 1540, 2451, 4340, 4351, 4362, 4373

\bool\_lazy\_and:nnTF .....  
1597, 2114, 2207, 2414, 2909, 3176, 3478,  
3835, 3863, 3933, 4153, 4278, 5313, 5852, 5869

\bool\_lazy\_or:nnTF .....  
..... 487, 951, 1842, 1863, 1941,  
2255, 2986, 3234, 3914, 3946, 3950, 4000,  
4004, 4420, 4438, 4456, 4835, 4840, 4860, 5755

\bool\_lazy\_or\_p:nn ..... 2210

\bool\_not\_p:n .....  
..... 1531, 1533, 1543, 1545, 2116, 2416

\bool\_set:Nn ..... 2990, 3857

\c\_false\_bool .....  
..... 1746, 1761, 1769, 3333, 3349, 3365

\g\_tmpa\_bool .....  
..... 4082, 4091, 4125, 4133, 4138, 4208,  
4216, 4250, 4258, 4263, 4347, 4358, 4369, 4380

\l\_tmpb\_bool ... 4425, 4439, 4457, 4479, 4492

\c\_true\_bool ..... 1720, 1726

box commands:

\box\_clear\_new:N ..... 1100, 1301

\box\_dp:N . 868, 888, 925, 944, 1126, 1135,  
1260, 1668, 1811, 1971, 1984, 2949, 4953, 5805

\box\_gclear\_new:N ..... 4882

\box\_grotate:Nn ..... 4920

\box\_ht:N ..... 869,  
890, 896, 908, 931, 942, 1128, 1130, 1133,  
1258, 1664, 1810, 1971, 1984, 2931, 4944, 5802

\box\_move\_up:nn . 64, 66, 68, 1854, 1919, 1958

\box\_rotate:Nn ..... 902

\box\_set\_dp:Nn ..... 924, 943, 1811

\box\_set\_ht:Nn ..... 930, 941, 1810

\box\_set\_wd:Nn ..... 918

\box\_use:N ..... 391, 909

\box\_use\_drop:N .... 949, 955, 974, 1692,  
1813, 1854, 1855, 1860, 2233, 4963, 5245, 5279

\box\_wd:N .....  
... 392, 919, 946, 953, 1313, 1315, 1859,  
1978, 1990, 2228, 2231, 2273, 2277, 4932, 5532

\l\_tmpa\_box .....  
.. 374, 391, 392, 1312, 1313, 1314, 1315,  
1436, 1810, 1811, 1813, 1854, 1855, 1971, 1984

\l\_tmpb\_box ..... 1964, 1978, 1979, 1990

## C

\c ..... 210, 1522

\Cdots ..... 1148, 3344, 3347

\cdots ..... 1083, 1140

\cellcolor ..... 1092, 1236, 4020

\chessboardcolors ..... 1243

\cline ..... 148, 1145, 1146

clist commands:

\clist\_clear:N ..... 3993

\clist\_if\_empty:NnTF ..... 4089, 4215, 5106

\clist\_if\_in:NnTF ..... 1059, 1505,  
1794, 4192, 4315, 5410, 5412, 5414, 5416, 5435

\clist\_if\_in:nnTF ..... 5396

\clist\_map\_inline:Nn .....  
... 3940, 3963, 3994, 4385, 5394, 5850, 5867

\clist\_map\_inline:nn ..... 2326, 3794, 3848

\clist\_new:N .... 301, 315, 316, 317, 318, 466

\clist\_put\_right:Nn ..... 4011

\clist\_set:Nn ..... 556, 557, 584, 585, 586

\clist\_set\_eq:NN ..... 3992

\l\_tmpa\_clist ..... 3992, 3994

\CodeAfter .....  
838, 1162, 2016, 2019, 2201, 2246, 2479, 6429

\CodeBefore ..... 1332, 6107

\color ..... 99, 106, 160, 162, 1440,  
1458, 2851, 2854, 2857, 2890, 2893, 2896,  
2971, 2974, 2978, 3046, 3098, 3558, 3561,  
3564, 3633, 3636, 3639, 3656, 3720, 3881,  
3882, 3893, 3894, 4889, 5024, 5652, 5984, 6008

\colorlet ..... 257, 258, 853, 860, 2216, 2260

\columncolor ..... 1094, 1242, 2488, 4043

\cr ..... 132, 150, 2195

\crrc ..... 2052

cs commands:

\cs\_generate\_variant:Nn .....  
..... 58, 152, 3173, 3713, 4613, 4614

\cs\_gset:Npn .....  
..... 99, 106, 2367, 2374, 2388, 2395, 4606

\cs\_gset\_eq:NN ... 186, 220, 1119, 1352, 2502

\cs\_if\_exist:NnTF ..... 57, 1101, 1104,  
1268, 1275, 1286, 1293, 1353, 1356, 2548,  
2549, 2634, 2647, 2702, 2715, 2804, 2826,  
2934, 2952, 3526, 3544, 3601, 3619, 4593,  
4646, 5162, 5180, 5627, 5772, 5779, 5808, 5815

\cs\_if\_exist\_p:N .....  
..... 488, 3838, 3866, 4422, 4441, 4459

\cs\_if\_free:NnTF .....  
..... 216, 2846, 2885, 2966, 3041, 3093

\cs\_if\_free\_p:N ..... 3672, 3674

\cs\_new\_protected:Npx .... 2528, 3683, 4509

\cs\_set:Nn ..... 636, 639, 642

\cs\_set:Npn ..... 95, 96,  
102, 103, 108, 120, 121, 135, 137, 138, 160,  
162, 337, 1075, 2593, 2655, 2723, 3568,  
3643, 4319, 4320, 4326, 4327, 4891, 4980, 4996

\cs\_set\_nopar:Npn .....  
..... 1001, 1013, 1110, 1113, 4788, 4789

\cs\_set\_nopar:Npx ..... 1014

\cs\_set\_protected:Npn ..... 5504

\cs\_set\_protected\_nopar:Npn .... 4870, 5129

## D

\Ddots ..... 1150, 3377, 3378, 3383, 3384

\ddots ..... 1085, 1142

\diagbox ..... 1163, 4870, 5129

dim commands:

\dim\_add:Nn ..... 4757

\dim\_compare:nNnTF .....  
98, 105, 916, 922, 928, 1549, 2112, 2277,  
2814, 2836, 3025, 4670, 4680, 5172, 5192, 5532

\dim\_gzero:N ..... 920, 926, 932

\dim\_max:nn ..... 4659, 4663

\dim\_min:nn ..... 4651, 4655

\dim\_ratio:nn ..... 3084, 3132,  
3248, 3253, 3264, 3272, 3281, 3286, 3297, 3305

\dim\_set:Nn ..... 4632, 4639, 4650, 4654,  
4658, 4662, 4674, 4675, 4684, 4685, 4729, 4742



<code>\dim_set_eq:Nn</code> . . . . .	4630, 4637, 4737, 4751	<code>\extrarowheight</code> . . .	2931, 4892, 4981, 4997, 5802
<code>\dim_sub:Nn</code> . . . . .	4754		
<code>\dim_use:N</code> . . . . .		<b>F</b>	
. . . . .	4671, 4681, 4732, 4733, 4745, 4746, 4769, 4770, 4771, 4772, 4796, 4797, 4798, 4799	<code>\fi</code> . . . . .	110, 1496, 4319, 4336
<code>\dim_zero_new:N</code> . . . . .	4629, 4631, 4636, 4638	fi commands:	
<code>\c_max_dim</code> . . . . .	2801, 2814, 2823, 2836, 4630, 4632, 4637, 4639, 4671, 4681, 5159, 5172, 5177, 5192, 5623, 5624, 5766, 5767, 5787, 5790	<code>\fi:</code> . . . . .	253
<code>\l_tmpc_dim</code> . . . . .	282, 2572, 2576, 3959, 3960, 3983, 4152, 4160, 4165, 4170, 4176, 4277, 4288, 4293, 4299, 5151, 5157, 5199, 5320, 5331, 5426, 5429, 5564, 5567, 5575	<code>\five</code> . . . . .	2553, 2558
<code>\l_tmpd_dim</code> . . . . .		flag commands:	
. . . . .	283, 2574, 2577, 3980, 3983, 4161, 4165, 4284, 4288, 5153, 5157, 5177, 5188, 5192, 5195, 5199, 5329, 5332, 5566, 5567, 5579	<code>\flag_clear_new:n</code> . . . . .	5930
<code>\dotfill</code> . . . . .	1161, 5521	<code>\flag_height:n</code> . . . . .	5955
<code>\dots</code> . . . . .	1087	<code>\flag_raise:n</code> . . . . .	5954
<code>\doublerulesep</code> . . . . .	1644, 4162, 4174, 4285, 4297, 4332	<code>\fontdimen</code> . . . . .	1850
<code>\doublerulesepcolor</code> . . . . .	102	fp commands:	
<code>\draw</code> . . . . .	3161	<code>\fp_eval:n</code> . . . . .	3189
		<code>\fp_to_dim:n</code> . . . . .	3224
		<code>\futurelet</code> . . . . .	115
<b>E</b>		<b>G</b>	
<code>\egroup</code> . . . . .	1484	<code>\globaldefs</code> . . . . .	1480, 5067
else commands:		group commands:	
<code>\else:</code> . . . . .	251	<code>\group_insert_after:N</code> . . . . .	5526, 5527, 5529, 5530
<code>\endarray</code> . . . . .	2004, 2032		
<code>\endBNiceMatrix</code> . . . . .	5468	<b>H</b>	
<code>\endbNiceMatrix</code> . . . . .	5465	<code>\halign</code> . . . . .	1137
<code>\endNiceArray</code> . . . . .	2346, 2355	<code>\hbox</code> . . . . .	1022, 1445, 1919, 1958, 2077, 2095, 2122, 2126, 2152
<code>\endNiceArrayWithDelims</code> . . . . .	2295, 2305	hbox commands:	
<code>\endpgfscope</code> . . . . .	3218, 5578	<code>\hbox:n</code> . . . . .	64, 66, 69
<code>\endpNiceMatrix</code> . . . . .	5456	<code>\hbox_gset:Nn</code> . . . . .	4884
<code>\endsavenotes</code> . . . . .	1485	<code>\hbox_overlap_left:n</code> . . . . .	2056, 2229
<code>\endtabularnotes</code> . . . . .	1887	<code>\hbox_overlap_right:n</code> . . . . .	391, 2173, 2275
<code>\endVNiceMatrix</code> . . . . .	5462	<code>\hbox_set:Nn</code> . . . . .	
<code>\endvNiceMatrix</code> . . . . .	5459	. . . . .	374, 1312, 1314, 1436, 1964, 1979, 5141
<code>\enskip</code> . . . . .	1718, 1759, 1767	<code>\hbox_set:Nw</code> . . . . .	842, 1317, 1681, 2204, 2250
<code>\ensuremath</code> . . . . .	3338, 3354, 3370, 3394, 3417	<code>\hbox_set_end:</code> . . . . .	938, 1388, 1689, 2223, 2268
<code>\everycr</code> . . . . .	131, 150, 1123	<code>\hbox_to_wd:nn</code> . . . . .	416, 441
exp commands:		<code>\Hdotsfor</code> . . . . .	1155, 6140, 6315
<code>\exp_after:wN</code> . . . . .	198, 1377, 1441, 1459, 1514, 5832	<code>\hdotsfor</code> . . . . .	1088
<code>\exp_args:Ne</code> . . . . .	3439	<code>\hdottedline</code> . . . . .	1152
<code>\exp_args:NNc</code> . . . . .	4595	<code>\heavyrulewidth</code> . . . . .	1897
<code>\exp_args:NNe</code> . . . . .	3435	<code>\hfil</code> . . . . .	1571
<code>\exp_args:Nne</code> . . . . .	2333	<code>\hfill</code> . . . . .	128, 145
<code>\exp_args:NNV</code> . . . . .	2024, 3325, 3341, 3357, 3373, 3397, 3498, 3574, 3652	<code>\Hline</code> . . . . .	1153, 4322
<code>\exp_args:NNv</code> . . . . .	1369	<code>\hline</code> . . . . .	108
<code>\exp_args:NNx</code> . . . . .	1058, 1793	<code>\hrule</code> . . . . .	112, 128, 145, 1067, 1897, 5657, 5989, 6013
<code>\exp_args:No</code> . . . . .	3149	<code>\hskip</code> . . . . .	111
<code>\exp_args:NV</code> . . . . .	1494, 2000, 2027, 2029, 5304	<code>\Hspace</code> . . . . .	1154
<code>\exp_args:Nxx</code> . . . . .	4863, 4864	<code>\hspace</code> . . . . .	3430
<code>\exp_last_unbraced:NV</code> . . . . .	1247, 2481, 5118	<code>\hss</code> . . . . .	1571
<code>\exp_not:N</code> . . . . .			
. . . . .	42, 43, 46, 47, 1593, 1637, 4031, 4043, 4511, 4512, 4899, 4909, 4987, 5004, 5121, 5937	<b>I</b>	
<code>\exp_not:n</code> . . . . .	993, 2494, 3508, 3584, 4020, 4031, 4033, 4044, 4879, 4961, 4973, 4974, 5089, 5101, 5111, 5123, 5138, 5542, 5543	<code>\ialign</code> . . . . .	1019, 1113, 1136, 5046
<code>\ExplSyntaxOff</code> . . . . .	218, 2378, 2399, 2496, 4409, 4610	<code>\Iddots</code> . . . . .	1151, 3401, 3402, 3407, 3408
<code>\ExplSyntaxOn</code> . . . . .		<code>\iddots</code> . . . . .	59, 1086, 1143
. . . . .	215, 2364, 2385, 2404, 2489, 4402, 4603	if commands:	
		<code>\if_mode_math:</code> . . . . .	251
		<code>\IfBooleanTF</code> . . . . .	1379
		<code>\ifnum</code> . . . . .	110, 4319, 4336
		<code>\ifstandalone</code> . . . . .	1356
		int commands:	
		<code>\int_case:nnTF</code> . . . . .	3375, 3381, 3399, 3405
		<code>\int_compare:nNnTF</code> . . . . .	123, 124, 140, 840, 841, 850, 857, 893, 903, 1064, 1066, 1202, 1204, 1253, 1263, 1282, 1389, 1393, 1404, 1408,

1409, 1476, 1875, 1914, 1953, 2025, 2053, 2253, 2609, 2616, 2620, 2622, 2677, 2684, 2688, 2690, 2853, 2892, 2973, 3008, 3010, 3458, 3472, 3560, 3635, 3909, 3954, 3972, 4008, 4039, 4056, 4057, 4064, 4065, 4069, 4485, 4487, 4489, 4491, 4888, 4925, 4937, 5219, 5248, 5252, 5322, 5324, 5482, 5484, 5486, 5490, 5492, 5494, 5496, 5498, 5836, 5838	
\int_compare_p:n . . . . .	
2755, 2756, 2757, 2758, 3924, 3926, 5314, 5315	
\int_do_until:nNnn . . . . .	3860
\int_gadd:Nn . . . . .	3471
\int_gincr:N . . . . .	839, 866, 1359, 1617, 1673, 1695, 1701, 2148, 2248, 3071, 3119, 4588, 4869
\int_if_even:nTF . . . . .	3803, 5955
\int_if_odd_p:n . . . . .	3857
\int_incr:N . . . . .	370, 1627
\int_min:nn . . . . .	
2567, 2569, 2571, 2573, 2581, 2583, 2763, 2764	
\int_step_inline:nn . . . . .	
2565, 3799, 3801, 5849, 5866	
\int_step_inline:nnnn . . . . .	4417, 4435, 4450, 4453
\l_tmpc_int . . . . .	3858, 3859, 3860
\c_zero_int . . . . .	1708, 3706, 3915
iow commands:	
\iow_now:Nn . . . . .	76, 213, 2404, 2405, 2407, 2489, 2490, 2496, 4402, 4403, 4409
\iow_shipout:Nn . . . . .	2364, 2365, 2372, 2378, 2385, 2386, 2393, 2399, 4603, 4604, 4610
\item . . . . .	1880, 1886
<b>K</b>	
\kern . . . . .	69
keys commands:	
\keys_define:nn . . . . .	483, 503, 510, 579, 626, 678, 717, 766, 786, 808, 821, 3421, 3810, 3819, 4577, 4807, 5016, 5339, 5445, 5584, 5589, 5669, 5690, 5699, 6045
\l_keys_key_str . . . . .	
6027, 6214, 6220, 6308, 6411, 6416, 6426, 6441, 6462, 6480, 6523, 6575, 6626	
\keys_set:nn . . . . .	512, 526, 755, 758, 765, 1373, 1374, 2332, 2342, 2351, 2505, 2856, 2895, 2976, 3045, 3097, 3563, 3638, 3655, 3814, 3833, 4590, 5083, 5591, 5595, 5722, 5761
\keys_set_known:nn . . . . .	
3387, 3411, 4852, 5296, 5351, 5390	
\l_keys_value_tl . . . . .	6261, 6337, 6344, 6676
<b>L</b>	
\Ldots . . . . .	1147, 3328, 3331
\ldots . . . . .	1082, 1139
\leaders . . . . .	128, 145
\left . . . . .	1441, 1967, 1982, 5653, 5985, 6009
legacy commands:	
\legacy_if:nTF . . . . .	610
\line . . . . .	2476, 6291, 6437
<b>M</b>	
\makebox . . . . .	1692
\mathinner . . . . .	61
mode commands:	
\mode_leave_vertical: . . . . .	1348, 1662
msg commands:	
\msg_error:nn . . . . .	12
\msg_error:nnn . . . . .	13
\msg_error:nnnn . . . . .	14, 5064, 5071, 5075
\msg_fatal:nn . . . . .	15
\msg_fatal:nnn . . . . .	16
\msg_new:nnn . . . . .	17
\msg_new:nnnn . . . . .	18
\msg_redirect_name:nnn . . . . .	20
\multicolumn . . . . .	1157, 3432, 3484, 3490, 3511
\multispan . . . . .	124, 125, 141, 142
\myfiledate . . . . .	6
\myfileversion . . . . .	7
<b>N</b>	
\newcolumnntype . . . . .	225, 226, 227
\newcounter . . . . .	330
\NewDocumentCommand . . . . .	
342, 365, 764, 2504, 3325, 3341, 3357, 3373, 3397, 3498, 3574, 3652, 3732, 3747, 3762, 3771, 3792, 3797, 3812, 3828, 4014, 4025, 4037, 5284, 5475, 5515, 5729, 5742	
\NewDocumentEnvironment . . . . .	1331, 1997, 2006, 2288, 2298, 2328, 2339, 2347, 4586
\NewExpandableDocumentCommand . . . . .	231, 4822
\newlist . . . . .	346, 357
\NiceArray . . . . .	2344, 2353
\NiceArrayWithDelims . . . . .	2293, 2303
nicematrix commands:	
\g_nicematrix_code_after_tl . . . . .	267, 619, 2021, 2481, 2483, 5086, 5098, 5108, 5602, 5609
\g_nicematrix_code_before_tl . . . . .	
1181, 2485, 2494, 4018, 4029, 4041, 5119	
\NiceMatrixLastEnv . . . . .	231
\NiceMatrixOptions . . . . .	764, 6481, 6681
\NiceMatrixoptions . . . . .	6341
\noalign . . . . .	98, 105, 110, 133, 1046, 1119, 4319, 4499
\nobreak . . . . .	360
\normalbaselines . . . . .	1107
\NotEmpty . . . . .	1164
\nulldelimiterspace . . . . .	1978, 1990, 5638, 5828
\nullfont . . . . .	5649, 5656, 5981, 5988, 6005, 6012
\numexpr . . . . .	153, 154
<b>O</b>	
\omit . . . . .	123, 2055, 2071, 2147, 5599
\OnlyMainNiceMatrix . . . . .	1160, 4048
<b>P</b>	
\par . . . . .	1874, 1882
peek commands:	
\peek_meaning:NTF . . . . .	158, 372, 1076
\peek_meaning_ignore_spaces:NTF . . . . .	1999, 4322
\peek_meaning_remove_ignore_spaces:NTF . . . . .	148
\peek_remove_spaces:n . . . . .	3456, 4016, 4027, 4824
\pgfdeclareshape . . . . .	2551
\pgfextracty . . . . .	5222
\pgfgetlastxy . . . . .	434
\pgfpathcircle . . . . .	3313
\pgfpathlineto . . . . .	4170, 4176, 4293, 4299, 5371, 5382, 5429, 5442, 5567, 5843, 5860, 5894
\pgfpathmoveto . . . . .	4169, 4175, 4292, 4298, 5370, 5381, 5428, 5441, 5562, 5842, 5859, 5885
\pgfpathrectanglecorners . . . . .	3982, 4163, 4286, 5330





1777, 1790, 1797, 1991, 1992, 1994, 1995, 2066, 2067, 2079, 2081, 2097, 2099, 2121, 2128, 2130, 2149, 2154, 2156, 2175, 2176, 2234, 2235, 2236, 2239, 2274, 2279, 2280, 2281	
\skip_horizontal:n	392, 1639
\skip_vertical:N	
133, 1028, 1030, 1444, 1455, 1871, 1896, 4499	
\skip_vertical:n	908, 4329
\g_tmpa_skip	2110, 2117, 2118, 2119, 2121, 2149
\c_zero_skip	1124
\space	264, 265
\stepcounter	378, 383
str commands:	
\c_backslash_str	264
\c_colon_str	763
\str_case:nn	
3439, 5226, 5238, 5260, 5272, 5879, 5888	
\str_case:nnTF	
1416, 1558, 1836, 4387, 5952, 5965	
\str_clear_new:N	5760
\str_foldcase:n	3439
\str_gclear:N	2500
\str_gset:Nn	
1345, 2292, 2301, 2330, 2341, 2349, 5506	
\str_if_empty:NTF	
875, 977, 1037, 1266, 1284, 1344, 2063, 2088, 2106, 2137, 2163, 2186, 2291, 2300, 2370, 2391, 4773, 4800, 5900, 5914, 5924	
\str_if_eq:nnTF	84, 263, 1017, 1585, 1588, 1632, 1784, 2011, 2013, 2046, 5302, 5607
\str_if_eq_p:nn	
489, 1598, 3948, 3952, 4002, 4006, 4837, 4842	
\str_if_in:NnTF	1824, 1928
\str_new:N	259, 471, 762
\str_range:Nnn	1828, 1932
\str_set:Nn	612, 751, 5716
\str_set_eq:NN	616, 763
\l_tmpa_str	612, 613, 615, 616
\strut	1880, 1886
\strutbox	2931, 2949, 5802, 5805
\SubMatrix	1244, 2463, 5740, 6230, 6262, 6269, 6275, 6279, 6297, 6444

## T

\tabcolsep	1010, 1448, 1452
\tabskip	1124
\tabularnote	342, 365, 372, 6369, 6389
\tabularnotes	1885
T <sub>E</sub> X and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\@BTnormal	1099
\@acol	1000
\@acoll	998
\@acolr	999
\@array@array	1002
\@arrayacol	998, 999, 1000
\@arstrutbox	868, 869, 908, 1126, 1128, 1130, 1133, 1135, 1664, 1668
\@currenenvir	5607
\@depth	5659, 5990, 6014
\@gobblethree	217
\@halignto	1001, 1013, 1014
\@height	113, 128, 145, 5657, 5989, 6013
\@ifclassloaded	51, 54, 6085, 6095
\@ifnextchar	1172

\@ifpackageloaded	
30, 33, 36, 39, 75, 91, 167, 6088, 6098	
\@mainaux	
76, 213, 2364, 2365, 2372, 2378, 2385, 2386, 2393, 2399, 2404, 2405, 2407, 2489, 2490, 2496, 4402, 4403, 4409, 4603, 4604, 4610	
\@tabarray	1015
\@tempswafalse	1496
\@tempswatrue	1495
\@emptokena	176, 179, 198, 200, 1494, 1514
\@whilesw	1496
\@width	113, 5660, 5991, 6015
\@xhline	116
\bBigg@	1312, 1314
\c@MaxMatrixCols	2320, 6154
\c@tabularnote	1864, 1875, 1903
\col@sep	1010, 1011, 1411, 1472, 2066, 2119, 2175, 2239, 2274, 2818, 2840
\CT@arc	95, 96
\CT@arc@	94, 99, 114, 127, 144, 160, 162, 1352, 1897, 2502, 4178, 4301, 4564, 5303, 5361, 5408, 5569, 5833
\CT@drs	102, 103
\CT@drsc@	101, 106, 4155, 4158, 4280, 4283
\CT@everycr	1117
\CT@row@color	1119
\if@tempswa	1496
\NC@	1075
\NC@find	177, 205
\NC@list	1496
\NC@rewrite@S	178, 196
\new@ifnextchar	1172
\newcol@	1077, 1078
\nicematrix@redefine@check@rerun	76, 79
\pgf@relevantforpicturesizefalse	
2538, 3141, 3310, 3717, 3847, 4146, 4271, 4693, 4703, 4714, 5145, 5299, 5360, 5407, 5556, 5617, 5764	
\pgfsys@getposition	1207, 1213, 1221
\pgfsys@markposition	
1029, 1206, 2059, 2080, 2098, 2129, 2155, 2179	
\pgfutil@check@rerun	81, 82
\reserved@a	115
\rvtx@ifformat@geq	57
\set@color	4888, 5141
\tikz@library@external@loaded	1353
tex commands:	
\tex_mkern:D	63, 65, 67, 70
\tex_the:D	200
\textfont	1850
\textit	334
\textsuperscript	335, 336
\the	153, 154, 1496, 1514
\thetabularnote	337
\tikzexternaldisable	1355
\tikzset	1230, 1357, 2466
tl commands:	
\tl_clear:N	4104, 4115, 4229, 4240, 5294
\tl_clear_new:N	3783, 3784, 3831, 4079, 4205, 5732, 5733, 5745, 5746, 5747, 5748
\tl_const:Nn	
42, 43, 46, 47, 458, 2197, 2242, 5941	
\tl_count:n	1831, 1935

<code>\tl_gclear:N</code> . . . . .	1498, 2478, 2483, 3722	<code>\tl_set_rescan:Nnn</code> . . . . .	2023, 3324, 3497, 3573, 3651
<code>\tl_gclear_new:N</code> . . . . .	1175, 1176, 1177, 1178, 1179, 1180, 1181	<code>\tl_to_str:n</code> . . . . .	6123
<code>\tl_gput_left:Nn</code> . . . . .	987, 1526, 4041	<code>\g_tmpa_tl</code> . . . . .	179, 182, 185
<code>\tl_gput_right:Nn</code> . . . . .	987, 1538, 1592, 1635, 1649, 1719, 1725, 1745, 1760, 1768, 1778, 3500, 3576, 3711, 4018, 4029, 4334, 4504, 4872, 5086, 5098, 5108, 5119, 5131, 5535	<code>\l_tmpc_tl</code> . . . . .	3783, 3785, 3788, 3956, 3975, 4079, 4093, 4094, 4097, 4102, 4104, 4108, 4113, 4115, 4205, 4218, 4219, 4222, 4227, 4229, 4233, 4238, 4240, 5353, 5365, 5378, 5400, 5417, 5423, 5732, 5734, 5738
<code>\tl_gset:Nn</code> . . . . .	179, 183, 185, 1337, 1338, 1339, 1501, 1507, 1712, 1713, 1755, 2492, 3709	<code>\l_tmpd_tl</code> . . . . .	3784, 3786, 3789, 5354, 5367, 5376, 5401, 5413, 5434, 5733, 5735, 5738
<code>\tl_if_blank:nTF</code> . . . . .	3734, 3737, 3749, 3752, 3764, 3767, 3773, 3776, 3878, 3890, 4826	token commands:	
<code>\tl_if_blank_p:n</code> . . . . .	3947, 3951, 4001, 4005, 4155, 4280, 4836, 4841	<code>\token_to_str:N</code> . . . . .	6106, 6107, 6140, 6225, 6230, 6236, 6261, 6269, 6275, 6279, 6291, 6297, 6315, 6328, 6369, 6376, 6389, 6406, 6416, 6428, 6437, 6443, 6481, 6681
<code>\tl_if_empty:NnTF</code> . . . . .	1054, 1376, 1439, 1457, 1874, 2461, 2462, 2485, 3969, 3970, 4093, 4097, 4108, 4218, 4222, 4233, 4847, 4887, 5084, 5116, 5300, 5651, 5831, 5983, 6007	U	
<code>\tl_if_empty:nTF</code> . . . . .	593, 726, 768, 790, 811, 829, 2008, 2035, 2857, 2896, 2977, 3046, 3098, 3564, 3639, 3656, 3880, 3892, 5708, 5950, 6139	<code>\unskip</code> . . . . .	1890
<code>\tl_if_empty_p:N</code> . . . . .	1532, 1544, 3177, 3178	use commands:	
<code>\tl_if_empty_p:n</code> . . . . .	1865	<code>\use:N</code> . . . . .	990, 1271, 1278, 1289, 1296, 1322, 1323, 1383, 1384, 2315, 2335, 3721
<code>\tl_if_eq:NnTF</code> . . . . .	3142	<code>\use:n</code> . . . . .	3657, 4048, 4897, 4907, 4985, 5002, 5936
<code>\tl_if_eq:NnTF</code> . . . . .	1056, 1499, 1787, 1812, 1923, 4191, 4314, 4539, 4541, 5848, 5865	<code>\usepackage</code> . . . . .	6090, 6100
<code>\tl_if_eq:nnTF</code> . . . . .	605, 742, 1751, 3705	<code>\usepgfmodule</code> . . . . .	2
<code>\tl_if_exist:NnTF</code> . . . . .	1366	V	
<code>\tl_if_in:NnTF</code> . . . . .	3851, 3943, 3966, 3997	vbox commands:	
<code>\tl_if_in:nnTF</code> . . . . .	1732, 1743, 1766	<code>\vbox:n</code> . . . . .	69
<code>\tl_if_single_token:nTF</code> . . . . .	549, 750	<code>\vbox_set_top:Nn</code> . . . . .	905
<code>\tl_if_single_token_p:n</code> . . . . .	58	<code>\vbox_to_ht:nn</code> . . . . .	412, 439, 1970, 1983
<code>\tl_item:Nn</code> . . . . .	182, 183, 185	<code>\vbox_to_zero:n</code> . . . . .	907
<code>\tl_map_inline:nn</code> . . . . .	2009	<code>\vcenter</code> . . . . .	1442, 1968, 5654, 5986, 6010, 6406
<code>\tl_new:N</code> . . . . .	181, 184, 237, 255, 267, 268, 269, 272, 276, 299, 300, 303, 305, 307, 332, 457, 461, 478, 480, 481	<code>\Vdots</code> . . . . .	1149, 3360, 3363
<code>\tl_put_left:Nn</code> . . . . .	1099	<code>\vdots</code> . . . . .	1084, 1141
<code>\tl_put_right:Nn</code> . . . . .	595, 1255, 1327, 1369	<code>\Vdotsfor</code> . . . . .	1156
<code>\tl_range:nnn</code> . . . . .	84	<code>\vfill</code> . . . . .	415, 441
<code>\tl_set:Nn</code> . . . . .	182, 3788, 3789, 3853, 3874, 3953, 3955, 3971, 3973, 4007, 4009, 4078, 4853, 5323, 5325, 5356, 5357, 5403, 5404	<code>\vline</code> . . . . .	114
<code>\tl_set_eq:NN</code> . . . . .	459, 3785, 3786, 3956, 4094, 4219, 4543, 4573, 4849, 5353, 5354, 5400, 5401, 5734, 5735, 5750, 5751, 5753, 5754	<code>\VNiceMatrix</code> . . . . .	5461
		<code>\vNiceMatrix</code> . . . . .	5458
		<code>\vrule</code> . . . . .	112, 1664, 1668
		<code>\vskip</code> . . . . .	111
		<code>\vtop</code> . . . . .	1026
		X	
		<code>\xglobal</code> . . . . .	853, 860, 2216, 2260
		Z	
		<code>\Z</code> . . . . .	5711

## Contents

1	The environment of this package	2
2	The vertical space between the rows	2
3	The vertical position of the arrays	3

<b>4</b>	<b>The blocks</b>	<b>4</b>
4.1	General case . . . . .	4
4.2	The mono-column blocks . . . . .	5
4.3	The mono-row blocks . . . . .	6
4.4	The mono-cell blocks . . . . .	6
4.5	A small remark . . . . .	6
<b>5</b>	<b>The rules</b>	<b>7</b>
5.1	Some differences with the classical environments . . . . .	7
5.1.1	The vertical rules . . . . .	7
5.1.2	The command <code>\cline</code> . . . . .	8
5.2	The thickness and the color of the rules . . . . .	8
5.3	The tools of <code>nicematrix</code> for the rules . . . . .	8
5.3.1	The <code>keyshlinesandvlines</code> . . . . .	9
5.3.2	The <code>keyhvlines</code> . . . . .	9
5.3.3	The <code>(empty)corners</code> . . . . .	9
5.4	The command <code>\diagbox</code> . . . . .	10
5.5	Dotted rules . . . . .	11
<b>6</b>	<b>The color of the rows and columns</b>	<b>11</b>
6.1	Use of <code>colortbl</code> . . . . .	11
6.2	The tools of <code>nicematrix</code> in the <code>\CodeBefore</code> . . . . .	12
6.3	Color tools with the syntax of <code>colortbl</code> . . . . .	15
<b>7</b>	<b>The width of the columns</b>	<b>16</b>
<b>8</b>	<b>The exterior rows and columns</b>	<b>17</b>
<b>9</b>	<b>The continuous dotted lines</b>	<b>18</b>
9.1	The option <code>nullify-dots</code> . . . . .	20
9.2	The commands <code>\Hdotsfor</code> and <code>\Vdotsfor</code> . . . . .	20
9.3	How to generate the continuous dotted lines transparently . . . . .	21
9.4	The labels of the dotted lines . . . . .	22
9.5	Customisation of the dotted lines . . . . .	22
9.6	The dotted lines and the rules . . . . .	23
<b>10</b>	<b>The <code>\CodeAfter</code></b>	<b>24</b>
10.1	The command <code>\line</code> in the <code>\CodeAfter</code> . . . . .	24
10.2	The command <code>\SubMatrix</code> in the <code>\CodeAfter</code> . . . . .	24
<b>11</b>	<b>The notes in the tabulars</b>	<b>26</b>
11.1	The footnotes . . . . .	26
11.2	The notes of tabular . . . . .	27
11.3	Customisation of the tabular notes . . . . .	28
11.4	Use of <code>{NiceTabular}</code> with <code>threeparttable</code> . . . . .	30
<b>12</b>	<b>Other features</b>	<b>30</b>
12.1	Use of the column type <code>S</code> of <code>siunitx</code> . . . . .	30
12.2	Alignment option in <code>{NiceMatrix}</code> . . . . .	31
12.3	The command <code>\rotate</code> . . . . .	31
12.4	The option <code>small</code> . . . . .	31
12.5	The counters <code>iRow</code> and <code>jCol</code> . . . . .	32
12.6	The option <code>light-syntax</code> . . . . .	33
12.7	Color of the delimiters . . . . .	33
12.8	The environment <code>{NiceArrayWithDelims}</code> . . . . .	33

<b>13</b>	<b>Use of Tikz with nicematrix</b>	<b>33</b>
13.1	The nodes corresponding to the contents of the cells . . . . .	33
13.2	The “medium nodes” and the “large nodes” . . . . .	34
13.3	The nodes which indicate the position of the rules . . . . .	36
13.4	The nodes corresponding to the command <code>\SubMatrix</code> . . . . .	37
<b>14</b>	<b>API for the developers</b>	<b>37</b>
<b>15</b>	<b>Technical remarks</b>	<b>38</b>
15.1	Definition of new column types . . . . .	38
15.2	Diagonal lines . . . . .	38
15.3	The “empty” cells . . . . .	39
15.4	The option <code>exterior-arraycolsep</code> . . . . .	39
15.5	Incompatibilities . . . . .	39
<b>16</b>	<b>Examples</b>	<b>40</b>
16.1	Notes in the tabulars . . . . .	40
16.2	Dotted lines . . . . .	41
16.3	Dotted lines which are no longer dotted . . . . .	42
16.4	Stacks of matrices . . . . .	43
16.5	How to highlight cells of a matrix . . . . .	45
16.6	Utilisation of <code>\SubMatrix</code> in the <code>\CodeBefore</code> . . . . .	48
<b>17</b>	<b>Implementation</b>	<b>48</b>
<b>18</b>	<b>History</b>	<b>199</b>
	<b>Index</b>	<b>206</b>