

# The package `nicematrix`\*

F. Pantigny  
fpantigny@wanadoo.fr

August 20, 2021

## Abstract

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

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

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

The package `nicematrix` is entirely contained in the file `nicematrix.sty`. This file may be put in the current directory or in a `texmf` tree. However, the best is to install `nicematrix` with a TeX distribution such 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 6.1 of `nicematrix`, at the date of 2021/08/20.

<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>{NiceTabularX}</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.

L'environnement `{NiceTabularX}` is similar to the environment `{tabularx}` from the eponymous package.<sup>3</sup>

**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.**

## 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>4</sup>

```

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


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


```

<sup>3</sup>In fact, it's possible to use directly the `X` columns in the environnement `{NiceTabular}` (and the required width for the tabular is fixed by the key `width`): cf. p. 18

<sup>4</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`<sup>5</sup>: `\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}

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

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

<sup>5</sup>The extension `booktabs` is *not* loaded by `nicematrix`.

## 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>6</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>7</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>6</sup>The spaces after a command `\Block` are deleted.

<sup>7</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` or the key `hvlines` 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>8</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`;
- 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 `\\`);
- the keys `hvlines` draws all the vertical and horizontal rules in the block;
- **New 5.19** when the key `tikz` is used, the Tikz path corresponding of the rectangle which delimits the block is executed with Tikz<sup>9</sup> by using as options the value of that key `tikz` (which must be a list of keys allowed for a Tikz path). For examples, cf. p. 42.

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.

**New 6.0** In the columns with a fixed width (columns `w{...}{...}`, `p{...}`, `b{...}`, `m{...}` and `X`), the content of the block is formatted as a paragraph of that width.

<sup>8</sup>This value is the initial value of the *rounded corners* of Tikz.

<sup>9</sup>Tikz should be loaded (by default, `nicematrix` only loads PGF) and, if it's not, an error will be raised.

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

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

### 4.3 The mono-row blocks

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

### 4.4 The mono-cell blocks

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

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

- It's possible to use the command `\` in a (mono-cell) block.
- 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>10</sup>
- It's possible to draw one or several borders of the cell with the key `borders`.

<code>\begin{NiceTabular}{cc}</code>	
<code>\toprule</code>	
<code>Writer &amp; \Block[l]{year of birth} \\\</code>	<b>Writer</b> year of birth
<code>\midrule</code>	
<code>Hugo &amp; 1802 \\\</code>	Hugo 1802
<code>Balzac &amp; 1799 \\\</code>	Balzac 1799
<code>\bottomrule</code>	
<code>\end{NiceTabular}</code>	

We recall that if the first mandatory argument of `\Block` is left blank, the block is mono-cell.<sup>11</sup>

<sup>10</sup>If one simply wishes to color the background of a unique cell, there is no point using the command `\Block`: it's possible to use the command `\cellcolor` (when the key `colortbl-like` is used).

<sup>11</sup>One may consider that the default value of the first mandatory argument of `\Block` is 1-1.

## 4.5 Horizontal position of the content of the block

By default, the horizontal position of the content of a block is computed by using the positions of the *contents* of the columns implied in that block. That's why, in the following example, the header "First group" is correctly centered despite the instruction `!\qquad` in the preamble which has been used to increase the space between the columns (this is not the behaviour of `\multicolumn`).

```
\begin{NiceTabular}{@{}c!\qquad ccc!\qquad ccc@{}}
\toprule
Rank & \Block{1-3}{First group} & & \Block{1-3}{Second group} & \\
& 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

In order to have an horizontal positionning of the content of the block computed with the limits of the columns of the LaTeX array (and not with the contents of those columns), one may use the key `L`, `R` and `C` of the command `\Block`.

## 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 & & \\
Mary & George & \\
\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. 10).

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. 40):

```
\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	<u>B</u>	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 40.



### 5.3 The tools of nicematrix for the rules

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

- the keys `hlines`, `vlines`, `hvlines` and `hvlines-except-borders`;
- 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>12</sup> presented p. 4;
  - the blocks implicitly delimited by the continuous dotted lines created by `\Cdots`, `\Vdots`, etc. (cf. p. 20).
- The corners are created by the key `corners` explained below (see p. 10).

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 keys `hvlines` and `hvlines-except-borders`

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

The key `hvlines-except-borders` is similar to the key `hvlines` but does not draw the rules on the horizontal and vertical borders of the array.

<sup>12</sup>And also the command `\multicolumn` but it’s recommended to use instead `\Block` in the environments of `nicematrix`.

### 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>13</sup>

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

In the example on the right (where B is in the center of a block of size  $2 \times 2$ ), we have colored in blue the four (empty) corners of the array.


When the key **corners** is used, `nicematrix` computes the (empty) corners and these corners will be taken into account by the tools for drawing the rules (the rules won't be drawn in the corners).

*Remark:* In the previous versions of `nicematrix`, there was only a key `hvlines-except-corners` (now considered as obsolete).

```
\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 \\
  & \Block{2-2}{B} & & A \\
  & & & A \\
\end{NiceTabular}
```


It's also possible to provide to the key **corners** a (comma-separated) list of corners (designed by NW, SW, NE and SE).

```
\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}
```


▷ The corners are also taken into account by the tools provided by `nicematrix` to color cells, rows and columns. These tools don't color the cells which are in the corners (cf. p. 12).

<sup>13</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.

## 5.4 The command `\diagbox`

The command `\diagbox` (inspired by the package `diagbox`), allows, when it is used in a cell, to slash that cell diagonally downwards.<sup>14</sup>

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

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

It's possible to use the command `\diagbox` in a `\Block`.

## 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}
```

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

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

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

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

It's possible to change in `nicematrix` the letter used to specify a vertical dotted line with the option `letter-for-dotted-lines` available in `\NiceMatrixOptions`. Thus released, the letter “:” can be used otherwise (for example by the package `arydshln`<sup>15</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>16</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:

<sup>14</sup>The author of this document considers that type of construction as graphically poor.

<sup>15</sup>However, one should remark that the package `arydshln` is not fully compatible with `nicematrix`.

<sup>16</sup>In fact, with `array`, this is true only for `\hline` and “|” but not for `\cline`: cf p. 8

- 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.

## 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>17</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`, `\rowlistcolors`, `\chessboardcolors` and `arraycolor`.<sup>18</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.

These commands don’t color the cells which are in the “corners” if the key `corners` is used. This key has been described p. 10.

- 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.

<sup>17</sup>If you use Overleaf, Overleaf will do automatically the right number of compilations.

<sup>18</sup>Remark that, in the `\CodeBefore`, PGF/Tikz nodes of the form “(i-lj)” are also available to indicate the position to the potential rules: cf. p. 37.

```

\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

- 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. 18). 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.

```

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

```

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

We have used the key `r` which aligns all the columns rightwards (cf. p. 32).

- 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).

```

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

```

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

- 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>19</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`.

- 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>20</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}{lr}[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

<sup>19</sup>The command `\rowcolors` of `xcolor` is available when `xcolor` is loaded with the option `table`. That option also loads the package `colortbl`.

<sup>20</sup>Otherwise, the color of a given row relies only upon the parity of its absolute number.

- **New 6.0** The extension `nicematrix` provides also a command `\rowlistcolors`. This command generalises the command `\rowcolors`: instead of two successive arguments for the colors, this command takes in an argument which is a (comma-separated) list of colors. In that list, the symbol `=` represent a color identical to the previous one.

```
\begin{NiceTabular}{c}
\CodeBefore
  \rowlistcolors{1}{red!15,blue!15,green!15}
\Body
Peter \\
James \\
Abigail \\
Elisabeth \\
Claudius \\
Jane \\
Alexandra \\
\end{NiceTabular}
```

Peter
James
Abigail
Elisabeth
Claudius
Jane
Alexandra

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=NE` to require the determination of the corner *north east* (NE).

```
\begin{NiceTabular}{cccccc}[corners=NE,margin,hvlines,first-row,first-col]
\CodeBefore
  \rowlistcolors{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

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{1-4}
& 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>21</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

## 7 The command `\RowStyle`

**New 5.18** The command `\RowStyle` takes in as argument some formatting intructions that will be applied to each cell on the rest of the current row.

That command also takes in as optional argument (between square brackets) a list of key-value pairs.

- The keys `cell-space-top-limit`, `cell-space-bottom-limit` and `cell-space-limits` are available with the same meaning that the corresponding global keys (cf. p. 2).
- **New 6.1** The key `color` sets the color of the text.<sup>22</sup>

```
\begin{NiceTabular}{cccc}[colortbl-like]
\hline
\RowStyle[cell-space-limits=3pt]{\rotate}
first & second & third & fourth \\
\rowcolor{blue!50}\RowStyle[color=white]{\sffamily}
1 & 2 & 3 & 4 \\
\end{NiceTabular}
```

first	second	third	fourth
1	2	3	4

The command `\rotate` is described p. 33.

<sup>21</sup>Up to now, this key is *not* available in `\NiceMatrixOptions`.

<sup>22</sup>The key `color` uses the command `\color` but inserts also an instruction `\leavevmode` before. This instruction prevents a extra vertical space in the cells which belong to columns of type `p`, `b`, `m` and `X` (which start in vertical mode).



## 8 The width of the columns

### 8.1 Basic tools

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`, `W`, `p`, `b` and `m` 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 (excepted the potential exterior columns: cf. p. 18) 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>23</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 arrays 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>24</sup>. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 4).

<sup>23</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>24</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}

```

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

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

## 8.2 The columns X

### New 6.0

The environment `{NiceTabular}` provides `X` columns similar to those provided by the environment `{tabularx}` of the eponymous package.

The required width of the tabular may be specified with the key `width`. The initial value of this parameter is `\linewidth`.

For sake of similarity with the environment `{tabularx}`, `nicematrix` also provides an environment `{NiceTabularX}` with a first mandatory argument which is the width of the tabular.

As with the packages `tabu` and `tabularray`, the specifier `X` takes in an optional argument (between square brackets) which is a list of keys.

- It's possible to give a weight for the column by providing an integer directly as argument of the specifier `X`. For example, a column `X[2]` will have a width double of the width of a column `X` (which has a weight equal to 1).
- It's possible to specify an horizontal alignment with one of the letters `l`, `c` and `r` (which inserts respectively `\raggedright`, `\centering` and `\raggedleft` followed by `\arraybackslash`).
- It's possible to specify a vertical alignment with one of the keys `t` (alias `p`), `m` and `b` (which construct respectively columns of type `p`, `m` and `b`). The default value is `t`.

```

\begin{NiceTabular}[width=9cm]{X[2,l]X[l]}[hvlines]
a rather long text which fits on several lines
& a rather long text which fits on several lines \\
a shorter text & a shorter text
\end{NiceTabular}

```

a rather long text which fits on several lines	a rather long text which fits on several lines
a shorter text	a shorter text

## 9 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`. It's particularly interesting for the (mathematical) matrices.

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}

```

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

The dotted lines have been drawn with the tools presented p. 20.

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>25</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. 34) is used, `nicematrix` has, in any case, to load the whole body of the environment (and that’s why it’s not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows (but not the number of columns).
  - In the other cases, `nicematrix` compute the number of rows and columns during the first compilation and write the result in the `aux` file for the next run.

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

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

```

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

```

<sup>25</sup>The users wishing exteriors columns with another type of alignment should consider the command `\SubMatrix` available in the `\CodeAfter` (cf. p. 26).

```

& a_{31} & a_{32} & a_{33} & a_{34} & \\
L_4 & a_{41} & a_{42} & a_{43} & a_{44} & L_4 \\
& C_1 & \Cdots & & C_4 & \\
\end{pNiceArray}$

```

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

### Remarks

- As shown in the previous example, the horizontal and vertical rules don'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 40.
- A specification of color present in `code-for-first-row` also applies to a dotted line drawn in that 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. 17) 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. 26.

## 10 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>26</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>27</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>28</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}

```

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

<sup>26</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>27</sup>The precise definition of a “non-empty cell” is given below (cf. p. 41).

<sup>28</sup>It's also possible to change the color of all these dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 24.

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}
```

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

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

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

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

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

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

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.<sup>29</sup>

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

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

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

## 10.1 The option `nullify-dots`

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

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

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

<sup>29</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. 17

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 & x
\end{pmatrix}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Remark: Unlike the command `\hdotsfor` of `amsmath`, the command `\Hdotsfor` may be used even when the package `colortbl`<sup>30</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`:

<sup>30</sup>We recall that when `xcolor` is loaded with the option `table`, the package `colortbl` is loaded.

```

\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}

```

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

### 10.3 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>31</sup>

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>26</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}

```

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

<sup>31</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.

## 10.4 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. 25) accept two optional arguments specified by the tokens `_` and `^` for labels positionned below and above the line. The arguments are composed in math mode with `\scriptstyle`.

```


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


```

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

## 10.5 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. 25) 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. 18.

### 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>32</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 <`”).

<sup>32</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.



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

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

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

## 10.6 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`, `hvlines` and `hvlines-except-borders` are not drawn within the blocks).<sup>33</sup>

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

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

## 11 The `\CodeAfter`

The option `code-after` may be used to give some code that will be executed *after* the construction of the matrix.<sup>34</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. 35.

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

### 11.1 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. 24).

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

<sup>33</sup>On the other side, the command `\line` in the `\CodeAfter` (cf. p. 25) does *not* create block.

<sup>34</sup>There is also a key `code-before` described p. 12.

```

\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. 40).

```

\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}

```

$$\left[ \begin{array}{cc|cc} 1 & \cdots & 1 & 2 \\ 0 & \ddots & \vdots & \vdots \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \end{array} \right] \left[ \begin{array}{cc|cc} 2 & \cdots & 2 & 2 \\ \vdots & & \vdots & \vdots \\ \vdots & & \vdots & \vdots \\ 2 & \cdots & 2 & 2 \end{array} \right]$$

## 11.2 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 `.`;
- 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>35</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.

```

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

```

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

<sup>35</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}`).

**New 5.18** In fact, the command `\SubMatrix` also takes in two optional arguments specified by the traditional symbols `^` and `_` for material in superscript and subscript.

```

 $\begin{bNiceMatrix}[right-margin=1em]$ 
1 & 1 & 1 & \\
1 & a & b & \\
1 & c & d & \\
\CodeAfter
\SubMatrix[2-2]{3-3}^T
\end{bNiceMatrix}

```

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & b \\ 1 & c & d \end{bmatrix}^T$$

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

- `left-xshift` and `right-xshift` 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}

```

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

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

```

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

```

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

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. 38.

It's also possible to specify some delimiters<sup>36</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.

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

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

## 12 The notes in the tabulars

### 12.1 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`.

### 12.2 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@{}}
\toprule \RowStyle{\bfseries}
Last name & First name & Birth day \\
\midrule
Achard\tabularnote{Achard is an old family of the Poitou.}
```

---

<sup>36</sup>Those delimiters are `(`, `[`, `\{` and the closing ones. Of course, it's also possible to put `|` and `||` in the preamble of the environment.

```

& Jacques & 5 juin 1962 \\
Lefebvre\footnote{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. 30. 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\footnote{Considered as the first nurse of
history.}\footnote{Nicknamed ``the Lady with the Lamp''.}
& Florence & 90 \\
Schoelcher & Victor & 89\footnote{The label of the note is overlapping.}\\
\end{NiceTabular}
\end{table}

```

```

Touchet & Marie & 89 \\
Wallis & John & 87 \\
\bottomrule
\end{NiceTabular}
\end{table}

```

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

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

Some text before the notes.

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

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

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

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

## 12.3 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. 30).

- 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. 42.

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

If you wish to use the environment `{NiceTabular}`, `{NiceTabular*}` `{NiceTabularX}` 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*}\TPT@hookin{NiceTabularX}}
\makeatother
```

## 13 Other features

### 13.1 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`.

### 13.2 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).

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

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



### 13.3 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`.<sup>37</sup>

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

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

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

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

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

### 13.4 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`).

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

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

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`;

<sup>37</sup>It can also be used in `\RowStyle` (cf. p. 16).

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

### 13.5 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>38</sup>. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

In the `\CodeBefore` (cf. p. 12) and in the `\CodeAfter` (cf. p. 25), `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).

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

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

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).

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

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

### 13.6 The option `light-syntax`

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

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

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

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

<sup>38</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.

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>39</sup>

## 13.7 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`.

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

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

This colour also applies to the delimiters drawn by the command `\SubMatrix` (cf. p. 26).

## 13.8 The environment `{NiceArrayWithDelims}`

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

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

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

# 14 Use of Tikz with nicematrix

## 14.1 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>40</sup>

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

<sup>39</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>40</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. 20) and the computation of the “corners” (cf. p. 10).

PGF). However, one should remind that `nicematrix` doesn't load Tikz by default. In the following examples, we assume that Tikz has been loaded.

```

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

```

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

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

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

```

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

```

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

In the following example, we have underlined all the nodes of the matrix (we explain below the technic used : cf. p. 48).

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

## 14.2 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>41</sup>

These nodes are not used by `nicematrix` by default, and that's why they are not created by default.

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

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

The names of the “large nodes” are constructed by adding the suffix “-large” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.<sup>42</sup>

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

The “large nodes” of the first column and last column may appear too small for some usage. That's why it's possible to use the options `left-margin` and `right-margin` to add space on both sides of

<sup>41</sup>There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

<sup>42</sup>There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 18).

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>43</sup>

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

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

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

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

Here is an array composed with the following code:

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

The nodes we have described are not available by default in the `\CodeBefore` (described p. 12). It’s possible to have these nodes available in the `\CodeBefore` by using the key `create-cell-nodes` of the keyword `\CodeBefore` (in that case, the nodes are created first before the construction of the array by using informations written on the `aux` file and created a second time during the contruction of the array itself).

### 14.3 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`. 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`.

<sup>43</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.

		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).

```

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

```

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

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

The command `\SubMatrix` available in the `\CodeAfter` has been described p. 26.

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}\}`).

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

## 15 API for the developpers

The package `nicematrix` provides two variables which are internal but public<sup>44</sup>:

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

These variables contain the code of what we have called the “code-before” (usually specified at the beginning of the environment with the syntax using the keywords `\CodeBefore` and `\Body`) and the “code-after” (usually specified at the end of the environment after the keyword `\CodeAfter`). 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 `\crossbox` to draw a cross in the current cell. This command will take in an optional argument between square brackets for a list of pairs *key-value* which will be given to Tikz before the drawing.

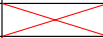
It’s possible to program such command `\crossbox` as follows, explicitly using the public variable `\g_nicematrix_code_after_tl`.

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

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

Here is an example of utilisation:

```
\begin{NiceTabular}{ccc}[hvlines]
merlan & requin & cabillaud \\
baleine & \crossbox[red] & morue \\
mante & raie & poule
\end{NiceTabular}
```

merlan	requin	cabillaud
baleine		morue
mante	raie	poule

<sup>44</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.

## 16 Technical remarks

### 16.1 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>45</sup>:

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

The heavy vertical rule won't extend in the exterior rows.<sup>46</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.

### 16.2 Diagonal lines

By default, all the diagonal lines<sup>47</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>45</sup>The command `\vrule` is a TeX (and not LaTeX) command.

<sup>46</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>47</sup>We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in the `\CodeAfter`.



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]`.

### 16.3 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`.
- A cell of a column of type `p`, `m` or `t` is always considered as not empty. *Caution* : One should not rely upon that point because it may change if a future version of `nicematrix`.

### 16.4 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>48</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>49</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).

### 16.5 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`:

<sup>48</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>49</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.

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

Up to 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.

The package `nicematrix` is not compatible with the class `ieeeaccess` (because that class is not compatible with PGF/Tikz).

## 17 Examples

### 17.1 Utilisation of the key “tikz” of the command `\Block`

```
\ttfamily \small
\begin{NiceTabular}{X[m]X[m]X[m]}[hvlines,cell-space-limits=3pt]
  \Block[tikz={pattern=grid,pattern color=lightgray}]{ }
    {pattern = grid,\ \ pattern color = lightgray}
& \Block[tikz={pattern = north west lines,pattern color=blue}]{ }
    {pattern = north west lines,\ \ pattern color = blue}
& \Block[tikz={outer color = red!50, inner color=white }]{2-1}
    {outer color = red!50,\ \ inner color = white} \ \
  \Block[tikz={pattern = sixpointed stars, pattern color = blue!15}]{ }
    {pattern = sixpointed stars,\ \ pattern color = blue!15}
& \Block[tikz={left color = blue!50}]{ }
    {left color = blue!50} \ \
\end{NiceTabular}
```

<pre>pattern = grid, pattern color = lightgray</pre>	<pre>pattern = north west lines, pattern color = blue</pre>	<pre>outer color = red!50, inner color = white</pre>
<pre>pattern = sixpointed stars, pattern color = blue!15</pre>	<pre>left color = blue!50</pre>	

### 17.2 Notes in the tabulars

The tools provided by `nicematrix` for the composition of the tabular notes have been presented in the section 12 p. 28.

Let’s consider that we wish to number the notes of a tabular with stars.<sup>50</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>51</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

<sup>50</sup>Of course, it’s realistic only when there is very few notes in the tabular.

<sup>51</sup>In fact: the value of its argument.

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

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

\*Achard is an old family of the Poitou.  
 \*\*The name Lefebvre is an alteration of the name Lefebure.

### 17.3 Dotted lines

An example with the resultant of two polynoms:

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

$$\left| \begin{array}{ccc} a_0 & & \\ & \ddots & \\ & & a_p \\ & & & \ddots & \\ & & & & a_0 \\ & & & & & \ddots & \\ & & & & & & a_1 \\ & & & & & & & \ddots & \\ & & & & & & & & a_p \end{array} \right| \quad \left| \begin{array}{ccc} b_0 & & \\ & \ddots & \\ & & b_q \\ & & & \ddots & \\ & & & & b_0 \\ & & & & & \ddots & \\ & & & & & & b_1 \\ & & & & & & & \ddots & \\ & & & & & & & & b_q \end{array} \right|$$

An example for a linear system:

```

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

```

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

## 17.4 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{pmatrix} 1 & & \vdots & & \vdots \\ & \ddots & & & \vdots \\ & & 1 & & \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix} \begin{matrix} \\ L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - L_1 \\ \\ L_n \leftarrow L_n - L_1 \end{matrix}

```

$$\begin{pmatrix} 1 & \cdots & 1 \\ \cdots & 0 & 1 \\ \cdots & 1 & 0 \\ & & \ddots & \ddots \\ & & & 1 & \cdots & 1 \end{pmatrix} \begin{matrix} \\ \leftarrow i \\ \leftarrow j \\ \\ \\ \end{matrix}$$

$\begin{matrix} \uparrow \\ i \end{matrix}$ 
 $\begin{matrix} \uparrow \\ j \end{matrix}$

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

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

## 17.5 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}

```

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

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

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

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

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{pNiceArray}$`

...

`\end{NiceMatrixBlock}`

$$\begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{pmatrix} \\
 \begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{pmatrix} \begin{matrix} \\ L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{matrix} \\
 \begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} \\ \\ L_3 \leftarrow 3L_2 + L_3 \end{matrix} \\
 \begin{pmatrix} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{pmatrix}$$

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}
\[\begin{NiceMatrix}[ r, last-col=6, code-for-last-col = \scriptstyle \color{blue} ]
12 & -8 & & 7 & 5 & 3 \\
3 & -18 & & 12 & 1 & 4 \\
-3 & -46 & & 29 & -2 & -15 \\
9 & 10 & & -5 & 4 & 7 \\
12 & -8 & & 7 & 5 & 3 \\
0 & 64 & & -41 & 1 & 19 & L_2 \gets L_1 - 4L_2 \\
0 & -192 & & 123 & -3 & -57 & L_3 \gets L_1 + 4L_3 \\
0 & -64 & & 41 & -1 & -19 & L_4 \gets 3L_1 - 4L_4 \\
12 & -8 & & 7 & 5 & 3 \\
0 & 64 & & -41 & 1 & 19 \\
0 & 0 & & 0 & 0 & 0 & L_3 \gets 3L_2 + L_3 \\
12 & -8 & & 7 & 5 & 3 \\
0 & 64 & & -41 & 1 & 19 \\
\CodeAfter [sub-matrix/vlines=4]
\SubMatrix({1-1}{4-5})
\SubMatrix({5-1}{8-5})
\SubMatrix({9-1}{11-5})
\SubMatrix({12-1}{13-5})
\end{NiceMatrix}\]

```

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

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

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

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

## 17.6 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>52</sup>).

```

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

```

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

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the commands `\hline` and `\Hline`, the specifier “|” and the options `hlines`, `vlines`, `hvlines` and `hvlines-except-borders` spread the cells.<sup>53</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).

```

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

```

<sup>52</sup>We recall that, if the first mandatory argument of the command `\Block` is left empty, that means that the block is a mono-cell block

<sup>53</sup>For the command `\cline`, see the remark p. 8.



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

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>54</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}
```

We create a rectangular Tikz node which encompasses the nodes of the second row by using the tools of the Tikz library `fit`. Those nodes are not available by default in the `\CodeBefore` (for efficiency). We have to require their creation with the key `create-cell-nodes` of the keyword `\CodeBefore`.

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

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

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

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

```
\[\begin{pNiceArray}{ccc}[last-col]
\CodeBefore [create-cell-nodes]
  \begin{tikzpicture}
    \node [highlight = (1-1) (1-3)] {} ;
    \node [highlight = (2-1) (2-3)] {} ;
    \node [highlight = (3-1) (3-3)] {} ;
  \end{tikzpicture}
\Body
```

---

<sup>54</sup>In Overleaf, the “built-in” PDF viewer does not show transparency. You can switch to the “native” viewer in that case.

```

a & a + b & a + b + c & L_1 \\
a & a      & a + b      & L_2 \\
a & a      & a          & L_3
\end{pNiceArray}\]

```

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

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

```

\[\begin{pNiceArray}{ccc}[last-col,create-medium-nodes]
\CodeBefore [create-cell-nodes]
\begin{tikzpicture} [name suffix = -medium]
\node [highlight = (1-1) (1-3)] {} ;
\node [highlight = (2-1) (2-3)] {} ;
\node [highlight = (3-1) (3-3)] {} ;
\end{tikzpicture}
\Body
a & a + b & a + b + c & L_1 \\
a & a      & a + b      & L_2 \\
a & a      & a          & L_3
\end{pNiceArray}\]

```

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

## 17.7 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 `\CodeBefore`.

$$L_i \begin{pmatrix} a_{i1} & \dots & a_{in} \\ \vdots & & \vdots \\ a_{i1} & \dots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} b_{11} & \dots & b_{1j} & \dots & b_{1n} \\ \vdots & & \vdots & & \vdots \\ b_{k1} & \dots & b_{kj} & \dots & b_{kn} \\ \vdots & & \vdots & & \vdots \\ b_{n1} & \dots & b_{nj} & \dots & b_{nn} \end{pmatrix}$$

```

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

```

```

\[\begin{NiceArray}{*{6}{c}@{\hspace{6mm}}*{5}{c}}[nullify-dots]
\CodeBefore [create-cell-nodes]
  \SubMatrix({2-7}{6-11})
  \SubMatrix({7-2}{11-6})
  \SubMatrix({7-7}{11-11})
  \begin{tikzpicture}
    \node [highlight = (9-2) (9-6)] { } ;
    \node [highlight = (2-9) (6-9)] { } ;
  \end{tikzpicture}
\Body
  & & & & & & & \color{blue}\scriptstyle C_j \\\
  & & & & & & b_{11} & \Cdots & b_{1j} & \Cdots & b_{1n} \\\
  & & & & & & \Vdots & & \Vdots & & \Vdots \\\
  & & & & & & & & b_{kj} \\\
  & & & & & & & & \Vdots \\\
  & & & & & & b_{n1} & \Cdots & b_{nj} & \Cdots & b_{nn} \\\[3mm]
  & a_{11} & \Cdots & & & a_{1n} \\\
  & \Vdots & & & & \Vdots & & & \Vdots \\\
\color{blue}\scriptstyle L_i
  & a_{i1} & \Cdots & a_{ik} & \Cdots & a_{in} & \Cdots & & c_{ij} \\\
  & \Vdots & & & & \Vdots \\\
  & a_{n1} & \Cdots & & & a_{nn} \\\
\CodeAfter
\tikz \draw [gray,shorten > = 1mm, shorten < = 1mm] (9-4.north) to [bend left] (4-9.west) ;
\end{NiceArray}\]

```

## 18 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 registered for this package.

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

<@@=nicematrix>

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

```

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

```

We give the traditional declaration of a package written with the L3 programming layer.

```

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 }

```

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

```

59 \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.

```

60 \cs_new_protected:Npn \@@_provide_pgfsyspdfmark:
61   {
62     \iow_now:Nn \@mainaux
63     {
64       \ExplSyntaxOn
65       \cs_if_free:NT \pgfsyspdfmark
66         { \cs_set_eq:NN \pgfsyspdfmark \@gobblethree }
67       \ExplSyntaxOff
68     }
69     \cs_gset_eq:NN \@@_provide_pgfsyspdfmark: \prg_do_nothing:
70   }

```

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.

```

71 \ProvideDocumentCommand \iddots { }
72   {
73     \mathinner
74     {
75       \tex_mkern:D 1 mu
76       \box_move_up:nn { 1 pt } { \hbox:n { . } }
77       \tex_mkern:D 2 mu
78       \box_move_up:nn { 4 pt } { \hbox:n { . } }
79       \tex_mkern:D 2 mu
80       \box_move_up:nn { 7 pt }
81       { \vbox:n { \kern 7 pt \hbox:n { . } } }
82       \tex_mkern:D 1 mu
83     }
84   }

```

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.

```

85 \AtBeginDocument

```

```

86 {
87   \@ifpackageloaded { booktabs }
88   { \iow_now:Nn \@mainaux \nicematrix@redefine@check@rerun }
89   { }
90 }
91 \cs_set_protected:Npn \nicematrix@redefine@check@rerun
92 {
93   \cs_set_eq:NN \@_old_pgful@check@rerun \pgful@check@rerun

```

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

```

94   \cs_set_protected:Npn \pgful@check@rerun ##1 ##2
95   {
96     \str_if_eq:eeF { nm- } { \tl_range:nnn { ##1 } 1 3 }
97     { \@_old_pgful@check@rerun { ##1 } { ##2 } }
98   }
99 }

```

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

```

100 \bool_new:N \c_@@_colortbl_loaded_bool
101 \AtBeginDocument
102 {
103   \@ifpackageloaded { colortbl }
104   { \bool_set_true:N \c_@@_colortbl_loaded_bool }
105   {

```

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.

```

106   \cs_set_protected:Npn \CT@arc@ { }
107   \cs_set:Npn \arrayrulecolor #1 # { \CT@arc@ { #1 } }
108   \cs_set:Npn \CT@arc@ #1 #2
109   {
110     \dim_compare:nNt \baselineskip = \c_zero_dim \noalign
111     { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }
112   }

```

Idem for `\CT@drs@`.

```

113   \cs_set:Npn \doublerulesepcolor #1 # { \CT@drs@ { #1 } }
114   \cs_set:Npn \CT@drs@ #1 #2
115   {
116     \dim_compare:nNt \baselineskip = \c_zero_dim \noalign
117     { \cs_gset:Npn \CT@drsc@ { \color #1 { #2 } } }
118   }
119   \cs_set:Npn \hline
120   {
121     \noalign { \ifnum 0 = ` } \fi
122     \cs_set_eq:NN \hskip \vskip
123     \cs_set_eq:NN \vrule \hrule
124     \cs_set_eq:NN \@width \@height
125     { \CT@arc@ \vline }
126     \futurelet \reserved@a
127     \@xhline
128   }
129 }
130 }

```

We will use `\AtBeginEnvironment`. For version of LaTeX posterior to 2020-10-01, the command is available in the LaTeX kernel (`l3hooks`). For older versions, we load `etoolbox`.

```

131 \cs_if_exist:NF \AtBeginEnvironment { \RequirePackage { etoolbox } }

```

The command `\AtBeginDocument` will be used to patch `{tabular}` in order to come back to the original versions of `\multicolumn` in the `{tabular}` nested in the environments of `nicematrix`.

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.

```

132 \cs_set:Npn \@@_standard_cline #1 { \@@_standard_cline:w #1 \q_stop }
133 \cs_set:Npn \@@_standard_cline:w #1-#2 \q_stop
134 {
135   \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
136   \int_compare:nNnT { #1 } > 1 { \multispan { \@@_pred:n { #1 } } & }
137   \multispan { \int_eval:n { #2 - #1 + 1 } }
138   {
139     \CT@arc@
140     \leaders \hrule \@height \arrayrulewidth \hfill

```

The following `\skip_horizontal:N \c_zero_dim` is to prevent a potential `\unskip` to delete the `\leaders`<sup>55</sup>

```

141   \skip_horizontal:N \c_zero_dim
142 }

```

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`.

```

143   \everycr { }
144   \cr
145   \noalign { \skip_vertical:N -\arrayrulewidth }
146 }

```

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.

```

147 \cs_set:Npn \@@_cline

```

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

```

148 { \@@_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*.

```

149 \cs_set:Npn \@@_cline_i:nn #1 #2 { \@@_cline_i:w #1-#2 \q_stop }
150 \cs_set:Npn \@@_cline_i:w #1-#2-#3 \q_stop
151 {

```

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).

```

152   \int_compare:nNnT { #1 } < { #2 }
153   { \multispan { \int_eval:n { #2 - #1 } } & }
154   \multispan { \int_eval:n { #3 - #2 + 1 } }
155   {
156     \CT@arc@
157     \leaders \hrule \@height \arrayrulewidth \hfill
158     \skip_horizontal:N \c_zero_dim
159   }

```

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

```

160   \peek_meaning_remove_ignore_spaces:NTF \cline
161   { & \@@_cline_i:en { \@@_succ:n { #3 } } }
162   { \everycr { } \cr }
163 }
164 \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.

```

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

```

The following command is a small shortcut.

```

167 \cs_new:Npn \@@_math_toggle_token:

```

---

<sup>55</sup>See question 99041 on TeX StackExchange.

```

168 { \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token }

169 \cs_new_protected:Npn \@@_set_CT@arc@:
170 { \peek_meaning:NTF [ \@@_set_CT@arc@_i: \@@_set_CT@arc@_ii: }
171 \cs_new_protected:Npn \@@_set_CT@arc@_i: [ #1 ] #2 \q_stop
172 { \cs_set:Npn \CT@arc@ { \color [ #1 ] { #2 } } }
173 \cs_new_protected:Npn \@@_set_CT@arc@_ii: #1 \q_stop
174 { \cs_set:Npn \CT@arc@ { \color { #1 } } }

175 \cs_set_eq:NN \@@_old_pgfpintanchor \pgfpintanchor

```

## 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.

```

176 \bool_new:N \c_@@_siunitx_loaded_bool
177 \AtBeginDocument
178 {
179   \@ifpackageloaded { siunitx }
180   { \bool_set_true:N \c_@@_siunitx_loaded_bool }
181   { }
182 }

```

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

```

183 \AtBeginDocument
184 {
185   \bool_if:NTF { ! \c_@@_siunitx_loaded_bool }
186   { \cs_set_eq:NN \@@_renew_NC@rewrite@S: \prg_do_nothing: }
187   {
188     \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
189     {
190       \renewcommand*{\NC@rewrite@S}[1] []
191       {

```

\@temptokena is a toks (not supported by expl3).

```

192         \@temptokena \exp_after:wN
193         { \tex_the:D \@temptokena \@@_S: [ ##1 ] }
194         \NC@find
195       }
196     }
197   }
198 }

```

The following code is used to define \c\_@@\_table\_collect\_begin\_tl and \c\_@@\_table\_print\_tl when the version of siunitx is prior to 3.0. The command \@@\_adapt\_S\_column is used in the environment {NiceArrayWithDelims}.

```

199 \AtBeginDocument
200 {
201   \cs_set_eq:NN \@@_adapt_S_column: \prg_do_nothing:
202   \bool_lazy_and:nnT
203   { \c_@@_siunitx_loaded_bool }
204   { ! \cs_if_exist_p:N \siunitx_cell_begin:w }
205   {
206     \cs_set_protected:Npn \@@_adapt_S_column:
207     {
208       \group_begin:
209       \@temptokena = { }
210       \cs_set_eq:NN \NC@find \prg_do_nothing:
211       \NC@rewrite@S { }
212       \tl_gset:NV \g_tmpa_tl \@temptokena
213       \group_end:
214       \tl_new:N \c_@@_table_collect_begin_tl

```



```

215         \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
216         \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
217         \tl_new:N \c_@@_table_print_tl
218         \tl_gset:Nx \c_@@_table_print_tl { \tl_item:Nn \g_tmpa_tl { -1 } }
219         \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
220     }
221 }
222 }

```

## Parameters

The following counter will count the environments `{NiceArray}`. The value of this counter will be used to prefix the names of the Tikz nodes created in the array.

```

223 \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).

```

224 \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.

```

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

```

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

```

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

```

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

```

229 \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).

```

230 \dim_new:N \l_@@_columns_width_dim

```

The dimension `\l_@@_col_width_dim` will be available in each cell which belongs to a column of fixed width: `w{...}{...}`, `W{...}{...}`, `p{}`, `m{}`, `b{}` but also `X` (when the actual width of that column is known, that is to say after the first compilation). It's the width of that column. It will be used by some commands `\Block`. A non positive value means that the column has no fixed width (it's a column of type `c`, `r`, `l`, etc.).

```

231 \dim_new:N \l_@@_col_width_dim
232 \dim_set:Nn \l_@@_col_width_dim { -1 cm }

```

The following counters will be used to count the numbers of rows and columns of the array.

```

233 \int_new:N \g_@@_row_total_int
234 \int_new:N \g_@@_col_total_int

```

The following token list will contain the type of horizontal alignment of the current cell as provided by the corresponding column. The possible values are `r`, `l`, `c`. For exemple, a column `p[1]{3cm}` will provide the value `l` for all the cells of the column.

```

235 \str_new:N \l_@@_hpos_cell_str
236 \str_set:Nn \l_@@_hpos_cell_str { c }

```

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

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

Idem pour the mono-row blocks.

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

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

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

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

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

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

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

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

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.

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

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

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

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

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

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

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

In a cell, it will be possible to know whether we are in a cell of a column of type `X` thanks to that flag.

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

We will write in `\g_@@_aux_tl` all the instructions that we have to write on the `aux` file for the current environment. The content of that token list will be written on the `aux` file at the end of the environment (in an instruction `\tl_gset:cn { c_@@_ \int_use:N \g_@@_env_int _ tl }`).

```
248 \tl_new:N \g_@@_aux_tl
```

```
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 vlins 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
```

The sum of the weights of all the X-columns in the preamble. The weight of a X-column is given as optional argument between square brackets. The default value, of course, is 1.

```
273 \int_new:N \g_@@_total_X_weight_int
```

If there is at least one X-column in the preamble of the array, the following flag will be raised via the aux file. The length `l_@@_x_columns_dim` will be the width of X-columns of weight 1 (the width of a column of weight  $n$  will be that dimension multiplied by  $n$ ). That value is computed after the construction of the array during the first compilation in order to be used in the following run.

```
274 \bool_new:N \l_@@_X_columns_aux_bool
```

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

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.

```
276 \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).

```
277 \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).

```
278 \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`).

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

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

The following token list will contain the code inserted in each cell of the current row (this token list will be cleared at the beginning of each row).

```
281 \tl_new:N \g_@@_row_style_tl
```

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

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

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

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

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

The L3 programming layer provides scratch dimensions `\l_tmpa_dim` and `\l_tmpb_dim`. We create two more in the same spirit (if they don't exist yet: that's why we use `\dim_zero_new:N`).

```
286 \dim_zero_new:N \l_tmpc_dim
```

```
287 \dim_zero_new:N \l_tmpd_dim
```

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

```
288 \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”.

```
289 \dim_new:N \g_@@_width_last_col_dim
290 \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.

```
291 \seq_new:N \g_@@_blocks_seq
```

We also manage a sequence of the *positions* of the blocks. In that sequence, each block is represented by only the four first components: `{imin}{jmin}{imax}{jmax}`. A block with the key `hvlines` won’t appear in that sequence (otherwise, the lines in that block would not be drawn!).

```
292 \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}`.

```
293 \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.

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

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

```
295 \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`).

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

The following flag will be raised if the key `width` has been raised. You use it to raise an error when this key is used while no column `X` is used.

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

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).

```
298 \seq_new:N \g_@@_multicolumn_cells_seq
```

```
299 \seq_new:N \g_@@_multicolumn_sizes_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).

```
300 \int_new:N \l_@@_row_min_int
```

```
301 \int_new:N \l_@@_row_max_int
```

```
302 \int_new:N \l_@@_col_min_int
```

```
303 \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.

```
304 \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).

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

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

```
306 \tl_new:N \l_@@_fill_tl
307 \tl_new:N \l_@@_draw_tl
308 \seq_new:N \l_@@_tikz_seq
309 \clist_new:N \l_@@_borders_clist
310 \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`.

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

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

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

The parameters of the horizontal position of the label of a block. If the user uses the key `c` or `C`, the value is `c`. If the user uses the key `l` or `L`, the value is `l`. If the user uses the key `r` or `R`, the value is `r`. If the user has used a capital letter, the boolean `\l_@@_hpos_of_block_cap_bool` will be raised (in the second pass of the analyze of the keys of the command `\Block`).

```
313 \str_new:N \l_@@_hpos_block_str
314 \str_set:Nn \l_@@_hpos_block_str { c }
315 \bool_new:N \l_@@_hpos_of_block_cap_bool
```

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`.

```
316 \tl_new:N \l_@@_vpos_of_block_tl
317 \tl_set:Nn \l_@@_vpos_of_block_tl { c }
```

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

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

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

```
319 \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.

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

321 \dim_new:N \l_@@_submatrix_extra_height_dim
322 \dim_new:N \l_@@_submatrix_left_xshift_dim
323 \dim_new:N \l_@@_submatrix_right_xshift_dim
324 \clist_new:N \l_@@_hlines_clist
325 \clist_new:N \l_@@_vlines_clist
326 \clist_new:N \l_@@_submatrix_hlines_clist
327 \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.

```
328 \int_new:N \l_@@_first_row_int
329 \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.

```
330 \int_new:N \l_@@_first_col_int
331 \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).

```
332 \int_new:N \l_@@_last_row_int
333 \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>56</sup>

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

Idem for `\l_@@_last_col_without_value_bool`

```
335 \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.

```
336 \int_new:N \l_@@_last_col_int
337 \int_set:Nn \l_@@_last_col_int { -2 }
```

However, we have also a boolean. Consider the following code:

---

<sup>56</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.

```

\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.

```

338 \bool_new:N \g_@@_last_col_found_bool

```

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

## 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.

```

339 \newcounter { tabularnote }

```

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

```

340 \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.

```

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

```

342 \int_new:N \l_@@_number_of_notes_int

```

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

```

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

```

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

```

344 \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`.

```

345 \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`.

```

346 \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`).

```

347 \AtBeginDocument
348 {
349   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
350   {
351     \NewDocumentCommand \tabularnote { m }
352     { \@@_error:n { enumitem~not~loaded } }
353   }
354   {

```

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.

```

355   \newlist { tabularnotes } { enumerate } { 1 }
356   \setlist [ tabularnotes ]
357   {
358     topsep = 0pt ,
359     noitemsep ,
360     leftmargin = * ,
361     align = left ,
362     labelsep = 0pt ,
363     label =
364     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
365   }
366   \newlist { tabularnotes* } { enumerate* } { 1 }
367   \setlist [ tabularnotes* ]
368   {
369     afterlabel = \nobreak ,
370     itemjoin = \quad ,
371     label =
372     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
373   }

```

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>57</sup>

```

374   \NewDocumentCommand \tabularnote { m }
375   {
376     \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
377     { \@@_error:n { tabularnote~forbidden } }
378     {

```

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

```

379     \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`.

```

380     \seq_gput_right:Nn \g_@@_tabularnotes_seq { #1 }
381     \peek_meaning:NF \tabularnote
382     {

```

---

<sup>57</sup>We should try to find a solution to that problem.

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.

```
383         \hbox_set:Nn \l_tmpa_box
384         {
```

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

```
385         \@@_notes_label_in_tabular:n
386         {
387             \stepcounter { tabularnote }
388             \@@_notes_style:n { tabularnote }
389             \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
390             {
391                 ,
392                 \stepcounter { tabularnote }
393                 \@@_notes_style:n { tabularnote }
394             }
395         }
396     }
```

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.

```
397         \addtocounter { tabularnote } { -1 }
398         \refstepcounter { tabularnote }
399         \int_zero:N \l_@@_number_of_notes_int
400         \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).

```
401         \skip_horizontal:n { \box_wd:N \l_tmpa_box }
402     }
403 }
404 }
405 }
406 }
```

## 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.

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

```

423     {
424         \vfill
425         \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
426     }
427 }
428 { #1 }
429 { }
430 \end { pgfscope }
431 }

```

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

```

432 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
433 {
434     \begin { pgfscope }
435     \pgfset
436     {
437         outer-sep = \c_zero_dim ,
438         inner-sep = \c_zero_dim ,
439         minimum-size = \c_zero_dim
440     }
441     \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
442     \pgfpointdiff { #3 } { #2 }
443     \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
444     \pgfnode
445     { rectangle }
446     { center }
447     {
448         \vbox_to_ht:nn
449         { \dim_abs:n \l_tmpb_dim }
450         { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
451     }
452     { #1 }
453     { }
454     \end { pgfscope }
455 }

```

## 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.

```

456 \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`.

```

457 \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`).

```

458 \dim_new:N \l_@@_cell_space_top_limit_dim
459 \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.

```

460 \dim_new:N \l_@@_inter_dots_dim
461 \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).

```
462 \dim_new:N \l_@@_xdots_shorten_dim
463 \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.

```
464 \dim_new:N \l_@@_radius_dim
465 \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.

```
466 \tl_new:N \l_@@_xdots_line_style_tl
467 \tl_const:Nn \c_@@_standard_tl { standard }
468 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

```
469 \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).

```
470 \tl_new:N \l_@@_baseline_tl
471 \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`).

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

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

```
473 \bool_new:N \l_@@_parallelize_diags_bool
474 \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.

```
475 \clist_new:N \l_@@_corners_clist
```

```
476 \dim_new:N \l_@@_notes_above_space_dim
477 \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.).

```
478 \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).

```
479 \bool_new:N \l_@@_auto_columns_width_bool
```

The following boolean corresponds to the key `create-cell-nodes` of the keyword `\CodeBefore`.

```
480 \bool_new:N \g_@@_recreate_cell_nodes_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.

```
481 \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”.

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

```
483 \bool_new:N \l_@@_large_nodes_bool
```

The boolean `\l_@@_except_borders_bool` will be raised when the key `hvlines-except-borders` will be used (but that key has also other effects).

```
484 \bool_new:N \l_@@_except_borders_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`).

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

```
486 \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`.

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

```
488 \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.

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

```
490 \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 “:”.

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

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

```
492 \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.

```
493 \bool_new:N \l_@@_delimiters_max_width_bool
```

```

494 \keys_define:nn { NiceMatrix / xdots }
495 {
496   line-style .code:n =
497   {
498     \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.

```

499     { \cs_if_exist_p:N \tikzpicture }
500     { \str_if_eq_p:nn { #1 } { standard } }
501     { \tl_set:Nn \l_@@xdots_line_style_tl { #1 } }
502     { \@@_error:n { bad-option-for-line-style } }
503   } ,
504   line-style .value_required:n = true ,
505   color .tl_set:N = \l_@@xdots_color_tl ,
506   color .value_required:n = true ,
507   shorten .dim_set:N = \l_@@xdots_shorten_dim ,
508   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 `_`.

```

509   down .tl_set:N = \l_@@xdots_down_tl ,
510   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).

```

511   draw-first .code:n = \prg_do_nothing: ,
512   unknown .code:n = \@@_error:n { Unknown-key-for-xdots }
513 }

```

```

514 \keys_define:nn { NiceMatrix / rules }
515 {
516   color .tl_set:N = \l_@@_rules_color_tl ,
517   color .value_required:n = true ,
518   width .dim_set:N = \arrayrulewidth ,
519   width .value_required:n = true
520 }

```

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

```

521 \keys_define:nn { NiceMatrix / Global }
522 {
523   delimiters .code:n =
524     \keys_set:nn { NiceMatrix / delimiters } { #1 } ,
525   delimiters .value_required:n = true ,
526   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
527   rules .value_required:n = true ,
528   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
529   standard-cline .default:n = true ,
530   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
531   cell-space-top-limit .value_required:n = true ,
532   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
533   cell-space-bottom-limit .value_required:n = true ,
534   cell-space-limits .meta:n =
535   {
536     cell-space-top-limit = #1 ,
537     cell-space-bottom-limit = #1 ,
538   } ,
539   cell-space-limits .value_required:n = true ,
540   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
541   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,

```

```

542 light-syntax .default:n = true ,
543 end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
544 end-of-row .value_required:n = true ,
545 first-col .code:n = \int_zero:N \l_@@_first_col_int ,
546 first-row .code:n = \int_zero:N \l_@@_first_row_int ,
547 last-row .int_set:N = \l_@@_last_row_int ,
548 last-row .default:n = -1 ,
549 code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
550 code-for-first-col .value_required:n = true ,
551 code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
552 code-for-last-col .value_required:n = true ,
553 code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
554 code-for-first-row .value_required:n = true ,
555 code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
556 code-for-last-row .value_required:n = true ,
557 hlines .clist_set:N = \l_@@_hlines_clist ,
558 vlines .clist_set:N = \l_@@_vlines_clist ,
559 hlines .default:n = all ,
560 vlines .default:n = all ,
561 vlines-in-sub-matrix .code:n =
562 {
563   \tl_if_single_token:nTF { #1 }
564   { \tl_set:Nn \l_@@_letter_vlism_tl { #1 } }
565   { @@_error:n { One-letter~allowed } }
566 } ,
567 vlines-in-sub-matrix .value_required:n = true ,
568 hvlines .code:n =
569 {
570   \clist_set:Nn \l_@@_vlines_clist { all }
571   \clist_set:Nn \l_@@_hlines_clist { all }
572 } ,
573 hvlines-except-borders .code:n =
574 {
575   \clist_set:Nn \l_@@_vlines_clist { all }
576   \clist_set:Nn \l_@@_hlines_clist { all }
577   \bool_set_true:N \l_@@_except_borders_bool
578 } ,
579 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.

```

580 renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
581 renew-dots .value_forbidden:n = true ,
582 nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
583 create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
584 create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
585 create-extra-nodes .meta:n =
586 { create-medium-nodes , create-large-nodes } ,
587 left-margin .dim_set:N = \l_@@_left_margin_dim ,
588 left-margin .default:n = \arraycolsep ,
589 right-margin .dim_set:N = \l_@@_right_margin_dim ,
590 right-margin .default:n = \arraycolsep ,
591 margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
592 margin .default:n = \arraycolsep ,
593 extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
594 extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
595 extra-margin .meta:n =
596 { extra-left-margin = #1 , extra-right-margin = #1 } ,
597 extra-margin .value_required:n = true ,
598 }

```

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

```

599 \keys_define:nn { NiceMatrix / Env }
600 {

```

The key `hvlines-except-corners` is now deprecated (use `hvlines` and `corners` instead).

```

601 hvlines-except-corners .code:n =
602 {
603   \clist_set:Nn \l_@@_corners_clist { #1 }
604   \clist_set:Nn \l_@@_vlines_clist { all }
605   \clist_set:Nn \l_@@_hlines_clist { all }
606 } ,
607 hvlines-except-corners .default:n = { NW , SW , NE , SE } ,
608 corners .clist_set:N = \l_@@_corners_clist ,
609 corners .default:n = { NW , SW , NE , SE } ,
610 code-before .code:n =
611 {
612   \tl_if_empty:nF { #1 }
613   {
614     \tl_put_right:Nn \l_@@_code_before_tl { #1 }
615     \bool_set_true:N \l_@@_code_before_bool
616   }
617 } ,

```

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

```

618 c .code:n = \tl_set:Nn \l_@@_baseline_tl c ,
619 t .code:n = \tl_set:Nn \l_@@_baseline_tl t ,
620 b .code:n = \tl_set:Nn \l_@@_baseline_tl b ,
621 baseline .tl_set:N = \l_@@_baseline_tl ,
622 baseline .value_required:n = true ,
623 columns-width .code:n =
624   \tl_if_eq:nnTF { #1 } { auto }
625   { \bool_set_true:N \l_@@_auto_columns_width_bool }
626   { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
627 columns-width .value_required:n = true ,
628 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.

```

629 \legacy_if:nF { measuring@ }
630 {
631   \str_set:Nn \l_tmpa_str { #1 }
632   \seq_if_in:NVTf \g_@@_names_seq \l_tmpa_str
633   { \@@_error:nn { Duplicate-name } { #1 } }
634   { \seq_gput_left:Nv \g_@@_names_seq \l_tmpa_str }
635   \str_set_eq:NN \l_@@_name_str \l_tmpa_str
636 } ,
637 name .value_required:n = true ,
638 code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
639 code-after .value_required:n = true ,
640 colortbl-like .code:n =
641   \bool_set_true:N \l_@@_colortbl_like_bool
642   \bool_set_true:N \l_@@_code_before_bool ,
643 colortbl-like .value_forbidden:n = true
644 }
645 \keys_define:nn { NiceMatrix / notes }
646 {
647   para .bool_set:N = \l_@@_notes_para_bool ,
648   para .default:n = true ,
649   code-before .tl_set:N = \l_@@_notes_code_before_tl ,
650   code-before .value_required:n = true ,
651   code-after .tl_set:N = \l_@@_notes_code_after_tl ,
652   code-after .value_required:n = true ,
653   bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,

```



```

654 bottomrule .default:n = true ,
655 style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
656 style .value_required:n = true ,
657 label-in-tabular .code:n =
658   \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
659 label-in-tabular .value_required:n = true ,
660 label-in-list .code:n =
661   \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
662 label-in-list .value_required:n = true ,
663 enumitem-keys .code:n =
664   {
665     \bool_if:NTF \c_@@_in_preamble_bool
666     {
667       \AtBeginDocument
668       {
669         \bool_if:NT \c_@@_enumitem_loaded_bool
670         { \setlist* [ tabularnotes ] { #1 } }
671       }
672     }
673     {
674       \bool_if:NT \c_@@_enumitem_loaded_bool
675       { \setlist* [ tabularnotes ] { #1 } }
676     }
677   } ,
678 enumitem-keys .value_required:n = true ,
679 enumitem-keys-para .code:n =
680   {
681     \bool_if:NTF \c_@@_in_preamble_bool
682     {
683       \AtBeginDocument
684       {
685         \bool_if:NT \c_@@_enumitem_loaded_bool
686         { \setlist* [ tabularnotes* ] { #1 } }
687       }
688     }
689     {
690       \bool_if:NT \c_@@_enumitem_loaded_bool
691       { \setlist* [ tabularnotes* ] { #1 } }
692     }
693   } ,
694 enumitem-keys-para .value_required:n = true ,
695 unknown .code:n = \@@_error:n { Unknown~key~for~notes }
696 }
697 \keys_define:nn { NiceMatrix / delimiters }
698 {
699   max-width .bool_set:N = \l_@@_delimiters_max_width_bool ,
700   max-width .default:n = true ,
701   color .tl_set:N = \l_@@_delimiters_color_tl ,
702   color .value_required:n = true ,
703 }

```

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

```

704 \keys_define:nn { NiceMatrix }
705 {
706   NiceMatrixOptions .inherit:n =
707     { NiceMatrix / Global } ,
708   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
709   NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
710   NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
711   NiceMatrixOptions / delimiters .inherit:n = NiceMatrix / delimiters ,
712   NiceMatrixOptions / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,

```

```

713 SubMatrix / rules .inherit:n = NiceMatrix / rules ,
714 CodeAfter / xdots .inherit:n = NiceMatrix / xdots ,
715 NiceMatrix .inherit:n =
716 {
717     NiceMatrix / Global ,
718     NiceMatrix / Env ,
719 } ,
720 NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
721 NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
722 NiceMatrix / delimiters .inherit:n = NiceMatrix / delimiters ,
723 NiceTabular .inherit:n =
724 {
725     NiceMatrix / Global ,
726     NiceMatrix / Env
727 } ,
728 NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
729 NiceTabular / rules .inherit:n = NiceMatrix / rules ,
730 NiceTabular / delimiters .inherit:n = NiceMatrix / delimiters ,
731 NiceArray .inherit:n =
732 {
733     NiceMatrix / Global ,
734     NiceMatrix / Env ,
735 } ,
736 NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
737 NiceArray / rules .inherit:n = NiceMatrix / rules ,
738 NiceArray / delimiters .inherit:n = NiceMatrix / delimiters ,
739 pNiceArray .inherit:n =
740 {
741     NiceMatrix / Global ,
742     NiceMatrix / Env ,
743 } ,
744 pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
745 pNiceArray / rules .inherit:n = NiceMatrix / rules ,
746 pNiceArray / delimiters .inherit:n = NiceMatrix / delimiters ,
747 }

```

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

```

748 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
749 {
750     last-col .code:n = \tl_if_empty:nF { #1 }
751                 { \@@_error:n { last-col~non-empty~for~NiceMatrixOptions } }
752                 \int_zero:N \l_@@_last_col_int ,
753     small .bool_set:N = \l_@@_small_bool ,
754     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.

```

755     renew-matrix .code:n = \@@_renew_matrix: ,
756     renew-matrix .value_forbidden:n = true ,

```

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

```

757     transparent .code:n =
758     {
759         \@@_renew_matrix:
760         \bool_set_true:N \l_@@_renew_dots_bool
761         \@@_error:n { Key~transparent }
762     } ,
763     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}`.

```

764     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.

```

765 columns-width .code:n =
766   \tl_if_eq:nnTF { #1 } { auto }
767   { \@@_error:n { Option~auto~for~columns~width } }
768   { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

769 allow-duplicate-names .code:n =
770   \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
771 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`).

```

772 letter-for-dotted-lines .code:n =
773   {
774     \tl_if_single_token:nTF { #1 }
775     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
776     { \@@_error:n { One~letter~allowed } }
777   } ,
778 letter-for-dotted-lines .value_required:n = true ,
779 notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
780 notes .value_required:n = true ,
781 sub-matrix .code:n =
782   \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
783 sub-matrix .value_required:n = true ,
784 unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
785 }

786 \str_new:N \l_@@_letter_for_dotted_lines_str
787 \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.

```

788 \NewDocumentCommand \NiceMatrixOptions { m }
789   { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

790 \keys_define:nn { NiceMatrix / NiceMatrix }
791   {
792     last-col .code:n = \tl_if_empty:nTF {#1}
793       {
794         \bool_set_true:N \l_@@_last_col_without_value_bool
795         \int_set:Nn \l_@@_last_col_int { -1 }
796       }
797       { \int_set:Nn \l_@@_last_col_int { #1 } } ,
798     l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
799     r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
800     small .bool_set:N = \l_@@_small_bool ,
801     small .value_forbidden:n = true ,
802     unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrix }
803   }

```

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

```
804 \keys_define:nn { NiceMatrix / NiceArray }
805 {
```

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.

```
806     small .bool_set:N = \l_@@_small_bool ,
807     small .value_forbidden:n = true ,
808     last-col .code:n = \tl_if_empty:nF { #1 }
809         { \@@_error:n { last-col~non-empty~for~NiceArray } }
810         \int_zero:N \l_@@_last_col_int ,
811     notes / para .bool_set:N = \l_@@_notes_para_bool ,
812     notes / para .default:n = true ,
813     notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
814     notes / bottomrule .default:n = true ,
815     tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
816     tabularnote .value_required: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~key~for~NiceArray }
820 }

821 \keys_define:nn { NiceMatrix / pNiceArray }
822 {
823     first-col .code:n = \int_zero:N \l_@@_first_col_int ,
824     last-col .code:n = \tl_if_empty:nF { #1 }
825         { \@@_error:n { last-col~non-empty~for~NiceArray } }
826         \int_zero:N \l_@@_last_col_int ,
827     first-row .code:n = \int_zero:N \l_@@_first_row_int ,
828     small .bool_set:N = \l_@@_small_bool ,
829     small .value_forbidden:n = true ,
830     r .code:n = \@@_error:n { r~or~l~with~preamble } ,
831     l .code:n = \@@_error:n { r~or~l~with~preamble } ,
832     unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrix }
833 }
```

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

```
834 \keys_define:nn { NiceMatrix / NiceTabular }
835 {
```

The dimension width will be used if at least a column of type X is used.

```
836     width .code:n = \dim_set:Nn \l_@@_width_dim { #1 }
837         \bool_set_true:N \l_@@_width_used_bool ,
838     width .value_required:n = true ,
839     notes / para .bool_set:N = \l_@@_notes_para_bool ,
840     notes / para .default:n = true ,
841     notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
842     notes / bottomrule .default:n = true ,
843     tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
844     tabularnote .value_required:n = true ,
845     last-col .code:n = \tl_if_empty:nF { #1 }
846         { \@@_error:n { last-col~non-empty~for~NiceArray } }
847         \int_zero:N \l_@@_last_col_int ,
848     r .code:n = \@@_error:n { r~or~l~with~preamble } ,
849     l .code:n = \@@_error:n { r~or~l~with~preamble } ,
850     unknown .code:n = \@@_error:n { Unknown~key~for~NiceTabular }
851 }
```

## Important code used by {NiceArrayWithDelims}

The pseudo-environment `\@@_cell_begin:w-\@@_cell_end:` 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}`).

```
852 \cs_new_protected:Npn \@@_cell_begin:w
853 {
```

The token list `\g_@@_post_action_cell_tl` will be set during the composition of the box `\l_@@_cell_box` and will be used *after* the composition in order to modify that box (that's why it's called a *post-action*).

```
854 \tl_gclear:N \g_@@_post_action_cell_tl
```

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`).

```
855 \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n
```

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

```
856 \int_gincr:N \c@jCol
```

Now, we increment the counter of the rows. We don't do this incrementation in the `\everycr` because some packages, like `arydshln`, create special rows in the `\halign` that we don't want to take into account.

```
857 \int_compare:nNnT \c@jCol = 1
858 { \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 `\@@_cell_end:` (and the potential `\c_math_toggle_token` also).

```
859 \hbox_set:Nw \l_@@_cell_box
860 \bool_if:NF \l_@@_NiceTabular_bool
861 {
862   \c_math_toggle_token
863   \bool_if:NT \l_@@_small_bool \scriptstyle
864 }
```

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.

```
865 \g_@@_row_style_tl
866 \int_compare:nNnTF \c@iRow = 0
867 {
868   \int_compare:nNnT \c@jCol > 0
869   {
870     \l_@@_code_for_first_row_tl
871     \xglobal \colorlet { nicematrix-first-row } { . }
872   }
873 }
874 {
875   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
876   {
877     \l_@@_code_for_last_row_tl
878     \xglobal \colorlet { nicematrix-last-row } { . }
879   }
880 }
881 }
```

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.

```
882 \cs_new_protected:Npn \@@_begin_of_row:
883 {
884   \tl_gclear:N \g_@@_row_style_tl
885   \int_gincr:N \c@iRow
```

```

886 \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
887 \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
888 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
889 \pgfpicture
890 \pgfrememberpicturepositiononpagetrue
891 \pgfcoordinate
892 { \@@_env: - row - \int_use:N \c@iRow - base }
893 { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
894 \str_if_empty:NF \l_@@_name_str
895 {
896   \pgfnodealias
897   { \l_@@_name_str - row - \int_use:N \c@iRow - base }
898   { \@@_env: - row - \int_use:N \c@iRow - base }
899 }
900 \endpgfpicture
901 }

```

Remark: If the key `recreate-cell-nodes` of the `\CodeBefore` is used, then we will add some lines to that command.

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.

```

902 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
903 {
904   \int_compare:nNnTF \c@iRow = 0
905   {
906     \dim_gset:Nn \g_@@_dp_row_zero_dim
907     { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
908     \dim_gset:Nn \g_@@_ht_row_zero_dim
909     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
910   }
911   {
912     \int_compare:nNnT \c@iRow = 1
913     {
914       \dim_gset:Nn \g_@@_ht_row_one_dim
915       { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
916     }
917   }
918 }
919 \cs_new_protected:Npn \@@_rotate_cell_box:
920 {
921   \box_rotate:Nn \l_@@_cell_box { 90 }
922   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
923   {
924     \vbox_set_top:Nn \l_@@_cell_box
925     {
926       \vbox_to_zero:n { }
927       \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
928       \box_use:N \l_@@_cell_box
929     }
930   }
931   \bool_gset_false:N \g_@@_rotate_bool
932 }
933 \cs_new_protected:Npn \@@_adjust_size_box:
934 {
935   \dim_compare:nNnT \g_@@_blocks_wd_dim > \c_zero_dim
936   {
937     \box_set_wd:Nn \l_@@_cell_box
938     { \dim_max:nn { \box_wd:N \l_@@_cell_box } \g_@@_blocks_wd_dim }
939     \dim_gzero:N \g_@@_blocks_wd_dim
940   }

```

```

941 \dim_compare:nNnT \g_@@_blocks_dp_dim > \c_zero_dim
942 {
943   \box_set_dp:Nn \l_@@_cell_box
944   { \dim_max:nn { \box_dp:N \l_@@_cell_box } \g_@@_blocks_dp_dim }
945   \dim_gzero:N \g_@@_blocks_dp_dim
946 }
947 \dim_compare:nNnT \g_@@_blocks_ht_dim > \c_zero_dim
948 {
949   \box_set_ht:Nn \l_@@_cell_box
950   { \dim_max:nn { \box_ht:N \l_@@_cell_box } \g_@@_blocks_ht_dim }
951   \dim_gzero:N \g_@@_blocks_ht_dim
952 }
953 }
954 \cs_new_protected:Npn \@@_cell_end:
955 {
956   \@@_math_toggle_token:
957   \hbox_set_end:

```

The token list `\g_@@_post_action_cell_tl` is (potentially) set during the composition of the box `\l_@@_cell_box` and is used now *after* the composition in order to modify that box.

```

958 \g_@@_post_action_cell_tl
959 \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
960 \@@_adjust_size_box:
961 \box_set_ht:Nn \l_@@_cell_box
962 { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
963 \box_set_dp:Nn \l_@@_cell_box
964 { \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”).

```

965 \dim_gset:Nn \g_@@_max_cell_width_dim
966 { \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”.

```

967 \@@_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. Up to 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.

```

968 \bool_if:NNTF \g_@@_empty_cell_bool
969 { \box_use_drop:N \l_@@_cell_box }
970 {
971   \bool_lazy_or:nnTF
972   \g_@@_not_empty_cell_bool
973   { \dim_compare_p:nNn { \box_wd:N \l_@@_cell_box } > \c_zero_dim }
974   \@@_node_for_cell:
975   { \box_use_drop:N \l_@@_cell_box }

```

```

976     }
977     \int_gset:Nn \g_@@_col_total_int { \int_max:nn \g_@@_col_total_int \c@jCol }
978     \bool_gset_false:N \g_@@_empty_cell_bool
979     \bool_gset_false:N \g_@@_not_empty_cell_bool
980 }

```

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

```

981 \cs_new_protected:Npn \@@_node_for_cell:
982 {
983     \pgfpicture
984     \pgfsetbaseline \c_zero_dim
985     \pgfrememberpicturepositiononpagetrue
986     \pgfset
987     {
988         inner-sep = \c_zero_dim ,
989         minimum-width = \c_zero_dim
990     }
991     \pgfnode
992     { rectangle }
993     { base }
994     { \box_use_drop:N \l_@@_cell_box }
995     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
996     { }
997     \str_if_empty:NF \l_@@_name_str
998     {
999         \pgfnodealias
1000         { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
1001         { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
1002     }
1003     \endpgfpicture
1004 }

```

As its name says, the following command is a patch for the command `\@@_node_for_cell:`. This patch will be appended on the left of `\@@_node_for_the_cell:` when the construction of the cell nodes (of the form (i-j)) in the `\CodeBefore` is required.

```

1005 \cs_new_protected:Npn \@@_patch_node_for_cell:n #1
1006 {
1007     \cs_new_protected:Npn \@@_patch_node_for_cell:
1008     {
1009         \hbox_set:Nn \l_@@_cell_box
1010         {
1011             \box_move_up:nn { \box_ht:N \l_@@_cell_box}
1012             \hbox_overlap_left:n
1013             {
1014                 \pgfsys@markposition
1015                 { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol - NW }

```

I don't know why the following adjustment is needed when the compilation is done with XeLaTeX or with the classical way `latex`, `divps`, `ps2pdf` (or Adobe Distiller). However, it seems to work.

```

1016             #1
1017         }
1018         \box_use:N \l_@@_cell_box
1019         \box_move_down:nn { \box_dp:N \l_@@_cell_box }
1020         \hbox_overlap_left:n
1021         {
1022             \pgfsys@markposition
1023             { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol - SE }
1024             #1
1025         }
1026     }
1027 }
1028 }

```



We have no explanation for the different behaviour between the TeX engines...

```

1029 \bool_lazy_or:n\TF \sys_if_engine_xetex_p: \sys_if_output_dvi_p:
1030 {
1031   \@@_patch_node_for_cell:n
1032     { \skip_horizontal:n { 0.5 \box_wd:N \l_@@_cell_box } }
1033 }
1034 { \@@_patch_node_for_cell:n { } }

```

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}

```

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

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.

```

1035 \cs_new_protected:Npn \@@_instruction_of_type:nnn #1 #2 #3
1036 {
1037   \bool_if:nTF { #1 } \tl_gput_left:cx \tl_gput_right:cx
1038     { g_@@_ #2 _ lines _ tl }
1039   {
1040     \use:c { @@ _ draw _ #2 : nnn }
1041     { \int_use:N \c@iRow }
1042     { \int_use:N \c@jCol }
1043     { \exp_not:n { #3 } }
1044   }
1045 }

```

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`.

```

1046 \cs_new_protected:Npn \@@_revtex_array:
1047 {
1048   \cs_set_eq:NN \@acoll \@arrayacol
1049   \cs_set_eq:NN \@acolr \@arrayacol
1050   \cs_set_eq:NN \@acol \@arrayacol
1051   \cs_set_nopar:Npn \@halignto { }
1052   \@array@array
1053 }
1054 \cs_new_protected:Npn \@@_array:
1055 {
1056   \bool_if:NTF \c_@@_revtex_bool
1057     \@@_revtex_array:
1058   {
1059     \bool_if:NTF \l_@@_NiceTabular_bool
1060       { \dim_set_eq:NN \col@sep \tabcolsep }
1061       { \dim_set_eq:NN \col@sep \arraycolsep }
1062     \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim
1063       { \cs_set_nopar:Npn \@halignto { } }
1064       { \cs_set_nopar:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }

```

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

```
1065     \@tabarray
1066 }
```

`\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.

```
1067 [ \str_if_eq:VnTF \l_@@_baseline_tl c c t ]
1068 }
```

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.

```
1069 \cs_set_eq:NN \@_old_ialign: \ialign
```

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

```
1070 \cs_new_protected:Npn \@_create_row_node:
1071 {
```

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

```
1072   \hbox
1073   {
1074     \bool_if:NT \l_@@_code_before_bool
1075     {
1076       \vtop
1077       {
1078         \skip_vertical:N 0.5\arrayrulewidth
1079         \pgfsys@markposition { \@_env: - row - \@_succ:n \c@iRow }
1080         \skip_vertical:N -0.5\arrayrulewidth
1081       }
1082     }
1083     \pgfpicture
1084     \pgfrememberpicturerepositiononpagetrue
1085     \pgfcoordinate { \@_env: - row - \@_succ:n \c@iRow }
1086     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
1087     \str_if_empty:NF \l_@@_name_str
1088     {
1089       \pgfnodealias
1090       { \l_@@_name_str - row - \@_succ:n \c@iRow }
1091       { \@_env: - row - \@_succ:n \c@iRow }
1092     }
1093     \endpgfpicture
1094   }
1095 }
```

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

```
1096 \cs_new:Npn \@_everycr: { \noalign { \@_everycr_i: } }
1097 \cs_new_protected:Npn \@_everycr_i:
1098 {
1099   \int_gzero:N \c@jCol
1100   \bool_gset_false:N \g_@@_after_col_zero_bool
1101   \bool_if:NF \g_@@_row_of_col_done_bool
1102   {
1103     \@_create_row_node:
```

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

```
1104   \tl_if_empty:NF \l_@@_hlines_clist
1105   {
1106     \tl_if_eq:NnF \l_@@_hlines_clist { all }
1107     {
1108       \exp_args:NNx
1109       \clist_if_in:NnT
1110       \l_@@_hlines_clist
```

```

1111         { \@@_succ:n \c@iRow }
1112     }
1113 {

```

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.

```

1114     \int_compare:nNtT \c@iRow > { -1 }
1115     {
1116         \int_compare:nNf \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@`.

```

1117         { \hrule height \arrayrulewidth width \c_zero_dim }
1118     }
1119 }
1120 }
1121 }
1122 }

```

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`).

```

1123 \cs_set_protected:Npn \@@_newcolumnntype #1
1124 {
1125     \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
1126     \peek_meaning:NTF [
1127         { \newcol@ #1 }
1128         { \newcol@ #1 [ 0 ] }
1129     }

```

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

```

1130 \cs_set_protected:Npn \@@_renew_dots:
1131 {
1132     \cs_set_eq:NN \ldots \@@_Ldots
1133     \cs_set_eq:NN \cdots \@@_Cdots
1134     \cs_set_eq:NN \vdots \@@_Vdots
1135     \cs_set_eq:NN \ddots \@@_Ddots
1136     \cs_set_eq:NN \iddots \@@_Iddots
1137     \cs_set_eq:NN \dots \@@_Ldots
1138     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
1139 }

```

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

```

1140 \cs_new_protected:Npn \@@_colortbl_like:
1141 {
1142     \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
1143     \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
1144     \cs_set_eq:NN \columncolor \@@_columncolor_preamble
1145 }

```

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

```

1146 \cs_new_protected:Npn \@@_pre_array_ii:
1147 {

```

The number of letters `X` in the preamble of the array.

```

1148     \int_gzero:N \g_@@_total_X_weight_int

```

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's 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>58</sup>.

```

1149 \bool_if:NT \c_@@_booktabs_loaded_bool
1150 { \tl_put_left:Nn \@BTnormal \@_create_row_node: }
1151 \box_clear_new:N \l_@@_cell_box
1152 \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}`).

```

1153 \bool_if:NT \l_@@_small_bool
1154 {
1155     \cs_set_nopar:Npn \arraystretch { 0.47 }
1156     \dim_set:Nn \arraycolsep { 1.45 pt }
1157 }

1158 \bool_if:NT \g_@@_recreate_cell_nodes_bool
1159 {
1160     \tl_put_right:Nn \@_begin_of_row:
1161     {
1162         \pgfsys@markposition
1163         { \@_env: - row - \int_use:N \c@iRow - base }
1164     }
1165 }

```

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`.

```

1166 \cs_set_nopar:Npn \ialign
1167 {
1168     \bool_if:NTF \c_@@_colortbl_loaded_bool
1169     {
1170         \CT@everycr
1171         {
1172             \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
1173             \@_everycr:
1174         }
1175     }
1176     { \everycr { \@_everycr: } }
1177     \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>59</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`.

```

1178 \dim_gzero_new:N \g_@@_dp_row_zero_dim
1179 \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
1180 \dim_gzero_new:N \g_@@_ht_row_zero_dim
1181 \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }

```

---

<sup>58</sup>cf. `\nicematrix@redefine@check@rerun`

<sup>59</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

```

1182 \dim_gzero_new:N \g_@@_ht_row_one_dim
1183 \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
1184 \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
1185 \dim_gzero_new:N \g_@@_ht_last_row_dim
1186 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
1187 \dim_gzero_new:N \g_@@_dp_last_row_dim
1188 \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`.

```

1189 \cs_set_eq:NN \ialign \@@_old_ialign:
1190 \halign
1191 }

```

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).

```

1192 \cs_set_eq:NN \@@_old_ldots \ldots
1193 \cs_set_eq:NN \@@_old_cdots \cdots
1194 \cs_set_eq:NN \@@_old_vdots \vdots
1195 \cs_set_eq:NN \@@_old_ddots \ddots
1196 \cs_set_eq:NN \@@_old_iddots \iddots
1197 \bool_if:NTF \l_@@_standard_cline_bool
1198 { \cs_set_eq:NN \cline \@@_standard_cline }
1199 { \cs_set_eq:NN \cline \@@_cline }
1200 \cs_set_eq:NN \Ldots \@@_Ldots
1201 \cs_set_eq:NN \Cdots \@@_Cdots
1202 \cs_set_eq:NN \Vdots \@@_Vdots
1203 \cs_set_eq:NN \Ddots \@@_Ddots
1204 \cs_set_eq:NN \Iddots \@@_Iddots
1205 \cs_set_eq:NN \hdottedline \@@_hdottedline:
1206 \cs_set_eq:NN \Hline \@@_Hline:
1207 \cs_set_eq:NN \Hspace \@@_Hspace:
1208 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
1209 \cs_set_eq:NN \Vdotsfor \@@_Vdotsfor:
1210 \cs_set_eq:NN \Block \@@_Block:
1211 \cs_set_eq:NN \rotate \@@_rotate:
1212 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
1213 \cs_set_eq:NN \dotfill \@@_old_dotfill:
1214 \cs_set_eq:NN \CodeAfter \@@_CodeAfter:
1215 \cs_set_eq:NN \diagbox \@@_diagbox:nn
1216 \cs_set_eq:NN \NotEmpty \@@_NotEmpty:
1217 \cs_set_eq:NN \RowStyle \@@_RowStyle:n
1218 \bool_if:NT \l_@@_colortbl_like_bool \@@_colortbl_like:
1219 \bool_if:NT \l_@@_renew_dots_bool \@@_renew_dots:

```

We redefine `\multicolumn` and, since we want `\multicolumn` to be available in the potential environments `{tabular}` nested in the environments of `nicematrix`, we patch `{tabular}` to go back to the original definition. The command `\AtBeginEnvironment` is the command of `l3hooks` and, if this command is not available (versions of LaTeX prior to 2020-10-01), `etoolbox` is loaded and the command `\AtBeginDocument` of `etoolbox` is used.

```

1220 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
1221 \AtBeginEnvironment { tabular }
1222 { \cs_set_eq:NN \multicolumn \@@_old_multicolumn }

```

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

```

1223 \seq_gclear:N \g_@@_multicolumn_cells_seq
1224 \seq_gclear: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).

```

1225 \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`).

```
1226 \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_begin:w` executed at the beginning of each cell.

```
1227 \int_gzero_new:N \g_@@_col_total_int
1228 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1229 \@@_renew_NC@rewrite@S:
1230 \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.

```
1231 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1232 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1233 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1234 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1235 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1236 \tl_gclear_new:N \g_@@_HVDotsfor_lines_tl

1237 \tl_gclear_new:N \g_nicematrix_code_before_tl
1238 }
```

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

The command `\@@_pre_array:` will be executed after analyse of the keys of the environment.

```
1239 \cs_new_protected:Npn \@@_pre_array:
1240 {
1241 \cs_if_exist:NT \theiRow { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
1242 \int_gzero_new:N \c@iRow
1243 \cs_if_exist:NT \thejCol { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
1244 \int_gzero_new:N \c@jCol
```

We recall that `\l_@@_last_row_int` and `\l_@@_last_column_int` are *not* the numbers of the last row and last column of the array. There are only the values of the keys `last-row` and `last-column` (maybe the user has provided erroneous values). The meaning of that counters does not change during the environment of `nicematrix`. There is only a slight adjustment: if the user have used one of those keys without value, we provide now the right value as read on the `aux` file (of course, it's possible only after the first compilation).

```
1245 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1246 {
1247 \bool_set_true:N \l_@@_last_row_without_value_bool
1248 \bool_if:NT \g_@@_aux_found_bool
1249 { \int_set:Nn \l_@@_last_row_int { \seq_item:Nn \c_@@_size_seq 3 } }
1250 }
1251 \int_compare:nNnT \l_@@_last_col_int = { -1 }
1252 {
1253 \bool_if:NT \g_@@_aux_found_bool
1254 { \int_set:Nn \l_@@_last_col_int { \seq_item:Nn \c_@@_size_seq 6 } }
1255 }
```

If there is a exterior row, we patch a command used in `\@@_cell_begin:w` in order to keep track of some dimensions needed to the construction of that “last row”.

```
1256 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1257 {
1258 \tl_put_right:Nn \@@_update_for_first_and_last_row:
1259 {
```

```

1260      \dim_gset:Nn \g_@@_ht_last_row_dim
1261      { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1262      \dim_gset:Nn \g_@@_dp_last_row_dim
1263      { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1264    }
1265  }

```

```

1266  \seq_gclear:N \g_@@_submatrix_seq

```

Now the `\CodeBefore`.

```

1267  \bool_if:NT \l_@@_code_before_bool \@@_exec_code_before:

```

The value of `\g_@@_pos_of_blocks_seq` has been written on the `aux` file and loaded before the (potential) execution of the `\CodeBefore`. Now, we clear that variable because it will be reconstructed during the creation of the array.

```

1268  \seq_gclear:N \g_@@_pos_of_blocks_seq

```

Idem for other sequences written on the `aux` file.

```

1269  \seq_gclear_new:N \g_@@_multicolumn_cells_seq
1270  \seq_gclear_new:N \g_@@_multicolumn_sizes_seq

```

The code in `\@@_pre_array_ii:` is used only here.

```

1271  \@@_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.

```

1272  \box_clear_new:N \l_@@_the_array_box

```

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

```

1273  \@@_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]`).

```

1274  \dim_zero_new:N \l_@@_left_delim_dim
1275  \dim_zero_new:N \l_@@_right_delim_dim
1276  \bool_if:NTF \l_@@_NiceArray_bool
1277  {
1278    \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1279    \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1280  }
1281  {

```

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

```

1282  \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \g_@@_left_delim_tl $ }
1283  \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1284  \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \g_@@_right_delim_tl $ }
1285  \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1286  }

```

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).

```

1287  \hbox_set:Nw \l_@@_the_array_box

```

```

1288 \skip_horizontal:N \l_@@_left_margin_dim
1289 \skip_horizontal:N \l_@@_extra_left_margin_dim
1290 \c_math_toggle_token
1291 \bool_if:NTF \l_@@_light_syntax_bool
1292 { \use:c { @@-light-syntax } }
1293 { \use:c { @@-normal-syntax } }
1294 }

```

The following command `\@@_pre_array_i:w` will be used when the keyword `\CodeBefore` is present at the beginning of the environment.

```

1295 \cs_new_protected:Npn \@@_pre_array_i:w #1 \Body
1296 {
1297   \tl_put_right:Nn \l_@@_code_before_tl { #1 }
1298   \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...

```

1299   \@@_pre_array:
1300 }

```

```

1301 \keys_define:nn { NiceMatrix / RowStyle }
1302 {
1303   cell-space-top-limit .code:n =
1304   {
1305     \tl_gput_right:Nn \g_@@_row_style_tl
1306     {
1307       \tl_gput_right:Nn \g_@@_post_action_cell_tl
1308       { \dim_set:Nn \l_@@_cell_space_top_limit_dim { #1 } }
1309     }
1310   } ,
1311   cell-space-top-limit .value_required:n = true ,
1312   cell-space-bottom-limit .code:n =
1313   {
1314     \tl_gput_right:Nn \g_@@_row_style_tl
1315     {
1316       \tl_gput_right:Nn \g_@@_post_action_cell_tl
1317       { \dim_set:Nn \l_@@_cell_space_bottom_limit_dim { #1 } }
1318     }
1319   } ,
1320   cell-space-bottom-limit .value_required:n = true ,
1321   cell-space-limits .meta:n =
1322   {
1323     cell-space-top-limit = #1 ,
1324     cell-space-bottom-limit = #1 ,
1325   } ,
1326   color .code:n =
1327   {
1328     \leavevmode \color { #1 }
1329     \tl_gput_right:Nn \g_@@_row_style_tl
1330     { \leavevmode \color { #1 } }
1331   } ,
1332   color .value_required:n = true ,
1333   unknown .code:n = \@@_error:n { Unknown-key-for-RowStyle }
1334 }

```

```

1335 \NewDocumentCommand \@@_RowStyle:n { 0 { } m }
1336 {
1337   \tl_gset:Nn \g_@@_row_style_tl { #2 }
1338   \keys_set:nn { NiceMatrix / RowStyle } { #1 }
1339   #2
1340   \ignorespaces
1341 }

```



## The \CodeBefore

The following command will be executed if the \CodeBefore has to be actually executed.

```
1342 \cs_new_protected:Npn \@@_pre_code_before:
1343 {
```

First, we give values to the LaTeX counters `iRow` and `jCol`. We remind that, in the `\CodeBefore` (and in the `\CodeAfter`) they represent the numbers of rows and columns of the array (without the potential last row and last column). The value of `\g_@@_row_total_int` is the number of the last row (with potentially a last exterior row) and `\g_@@_col_total_int` is the number of the last column (with potentially a last exterior column).

```
1344 \int_set:Nn \c@iRow { \seq_item:Nn \c_@@_size_seq 2 }
1345 \int_set:Nn \c@jCol { \seq_item:Nn \c_@@_size_seq 5 }
1346 \int_set_eq:NN \g_@@_row_total_int { \seq_item:Nn \c_@@_size_seq 3 }
1347 \int_set_eq:NN \g_@@_col_total_int { \seq_item:Nn \c_@@_size_seq 6 }
```

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.

```
1348 \pgfsys@markposition { \@@_env: - position }
1349 \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1350 \pgfpicture
1351 \pgf@relevantforpicturesizefalse
```

First, the recreation of the row nodes.

```
1352 \int_step_inline:nnn \l_@@_first_row_int { \g_@@_row_total_int + 1 }
1353 {
1354   \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1355   \pgfcoordinate { \@@_env: - row - ##1 }
1356   { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1357 }
```

Now, the recreation of the `col` nodes.

```
1358 \int_step_inline:nnn \l_@@_first_col_int { \g_@@_col_total_int + 1 }
1359 {
1360   \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1361   \pgfcoordinate { \@@_env: - col - ##1 }
1362   { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1363 }
```

Now, you recreate the diagonal nodes by using the row nodes and the `col` nodes.

```
1364 \@@_create_diag_nodes:
```

Now, the creation of the cell nodes (`i-j`), and, maybe also the “medium nodes” and the “large nodes”.

```
1365 \bool_if:NT \g_@@_recreate_cell_nodes_bool \@@_recreate_cell_nodes:
1366 \endpgfpicture

1367 \bool_if:NT \c_@@_tikz_loaded_bool
1368 {
1369   \tikzset
1370   {
1371     every-picture / .style =
1372     { overlay , name-prefix = \@@_env: - }
1373   }
1374 }

1375 \cs_set_eq:NN \cellcolor \@@_cellcolor
1376 \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
1377 \cs_set_eq:NN \roundedrectanglecolor \@@_roundedrectanglecolor
1378 \cs_set_eq:NN \rowcolor \@@_rowcolor
1379 \cs_set_eq:NN \rowcolors \@@_rowcolors
1380 \cs_set_eq:NN \rowlistcolors \@@_rowlistcolors
1381 \cs_set_eq:NN \arraycolor \@@_arraycolor
1382 \cs_set_eq:NN \columncolor \@@_columncolor
1383 \cs_set_eq:NN \chessboardcolors \@@_chessboardcolors
1384 \cs_set_eq:NN \SubMatrix \@@_SubMatrix_in_code_before
1385 }
```

```

1386 \cs_new_protected:Npn \@@_exec_code_before:
1387 {
1388   \seq_gclear_new:N \g_@@_colors_seq
1389   \bool_gset_false:N \g_@@_recreate_cell_nodes_bool
1390   \group_begin:

```

We compose the `code-before` in math mode in order to nullify the spaces put by the user between instructions in the `code-before`.

```

1391   \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token

```

Here is the `\CodeBefore`. The construction is a bit complicated because `\l_@@_code_before_tl` may begin with keys between square brackets. Moreover, after the analyze of those keys, we sometimes have to decide to do *not* execute the rest of `\l_@@_code_before_tl` (when it is asked for the creation of cell nodes in the `\CodeBefore`). That's why we begin with a `\q_stop`: it will be used to discard the rest of `\l_@@_code_before_tl`.

```

1392   \exp_last_unbraced:NV \@@_CodeBefore_keys: \l_@@_code_before_tl \q_stop

```

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.

```

1393   \@@_actually_color:
1394   \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1395   \group_end:
1396   \bool_if:NT \g_@@_recreate_cell_nodes_bool
1397   { \tl_put_left:Nn \@@_node_for_cell: \@@_patch_node_for_cell: }
1398 }

```

```

1399 \keys_define:nn { NiceMatrix / CodeBefore }
1400 {
1401   create-cell-nodes .bool_gset:N = \g_@@_recreate_cell_nodes_bool ,
1402   create-cell-nodes .default:n = true ,
1403   sub-matrix .code:n = \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
1404   sub-matrix .value_required:n = true ,
1405   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
1406   delimiters / color .value_required:n = true ,
1407   unknown .code:n = \@@_error:n { Unknown-key-for-CodeAfter }
1408 }
1409 \NewDocumentCommand \@@_CodeBefore_keys: { 0 { } }
1410 {
1411   \keys_set:nn { NiceMatrix / CodeBefore } { #1 }
1412   \@@_CodeBefore:w
1413 }

```

We have extracted the options of the keyword `\CodeBefore` in order to see whether the key `create-cell-nodes` has been used. Now, you can execute the rest of the `\CodeAfter`, excepted, of course, if we are in the first compilation.

```

1414 \cs_new_protected:Npn \@@_CodeBefore:w #1 \q_stop
1415 {
1416   \bool_if:NT \g_@@_aux_found_bool
1417   {
1418     \@@_pre_code_before:
1419     #1
1420   }
1421 }

```

By default, if the user uses the `\CodeBefore`, only the `col` nodes, `row` nodes and `diag` nodes are available in that `\CodeBefore`. With the key `create-cell-nodes`, the cell nodes, that is to say the nodes of the form `(i-j)` (but not the extra nodes) are also available because those nodes also are recreated and that recreation is done by the following command.

```

1422 \cs_new_protected:Npn \@@_recreate_cell_nodes:
1423 {

```

```

1424 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
1425 {
1426   \pgfsys@getposition { \@@_env: - ##1 - base } \@@_node_position:
1427   \pgfcoordinate { \@@_env: - row - ##1 - base }
1428   { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1429   \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
1430   {
1431     \cs_if_exist:cT
1432     { pgf @ sys @ pdf @ mark @ pos @ \@@_env: - ##1 - #####1 - NW }
1433     {
1434       \pgfsys@getposition
1435       { \@@_env: - ##1 - #####1 - NW }
1436       \@@_node_position:
1437       \pgfsys@getposition
1438       { \@@_env: - ##1 - #####1 - SE }
1439       \@@_node_position_i:
1440       \@@_pgf_rect_node:nnn
1441       { \@@_env: - ##1 - #####1 }
1442       { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1443       { \pgfpointdiff \@@_picture_position: \@@_node_position_i: }
1444     }
1445   }
1446 }
1447 \@@_create_extra_nodes:
1448 }

```

## The environment {NiceArrayWithDelims}

```

1449 \NewDocumentEnvironment { NiceArrayWithDelims }
1450 { m m O { } m ! O { } t \CodeBefore }
1451 {
1452   \@@_provide_pgfsyspdfmark:
1453   \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.

```

1454   \bgroup

1455   \tl_gset:Nn \g_@@_left_delim_tl { #1 }
1456   \tl_gset:Nn \g_@@_right_delim_tl { #2 }
1457   \tl_gset:Nn \g_@@_preamble_tl { #4 }

1458   \int_gzero:N \g_@@_block_box_int
1459   \dim_zero:N \g_@@_width_last_col_dim
1460   \dim_zero:N \g_@@_width_first_col_dim
1461   \bool_gset_false:N \g_@@_row_of_col_done_bool
1462   \str_if_empty:NT \g_@@_name_env_str
1463   { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }

```

The following line will be deleted when we will consider that only versions of `siunitx` after v3.0 are compatible with `nicematrix`.

```

1464   \@@_adapt_S_column:
1465   \bool_if:NTF \l_@@_NiceTabular_bool
1466     \mode_leave_vertical:
1467     \@@_test_if_math_mode:
1468     \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
1469     \bool_set_true:N \l_@@_in_env_bool

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>60</sup>. This command is used by `\CT@arc@` but we use it also for compatibility with `colortbl`. But we want also to be able

---

<sup>60</sup>e.g. `\color[rgb]{0.5,0.5,0}`

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.

```
1470 \cs_gset_eq:NN \@@_old_CT@arc@ \CT@arc@
```

We deactivate Tikz externalization because we will use PGF pictures with the options `overlay` and `remember picture` (or equivalent forms). We deactivate with `\tikzexternaldisable` and not with `\tikzset{external/export=false}` which is *not* equivalent.

```
1471 \cs_if_exist:NT \tikz@library@external@loaded
1472 {
1473   \tikzexternaldisable
1474   \cs_if_exist:NT \ifstandalone
1475   { \tikzset { external / optimize = false } }
1476 }
```

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

```
1477 \int_gincr:N \g_@@_env_int
1478 \bool_if:NF \l_@@_block_auto_columns_width_bool
1479 { \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 (except the blocks with the key `hvlines`).

```
1480 \seq_gclear:N \g_@@_blocks_seq
1481 \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`.

```
1482 \seq_gclear:N \g_@@_pos_of_stroken_blocks_seq
1483 \seq_gclear:N \g_@@_pos_of_xdots_seq
1484 \tl_gclear_new:N \g_@@_code_before_tl
1485 \tl_gclear:N \g_@@_row_style_tl
```

We load all the informations written in the aux file during previous compilations corresponding to the current environment.

```
1486 \bool_gset_false:N \g_@@_aux_found_bool
1487 \tl_if_exist:cT { c_@@ _ \int_use:N \g_@@_env_int _ tl }
1488 {
1489   \bool_gset_true:N \g_@@_aux_found_bool
1490   \use:c { c_@@ _ \int_use:N \g_@@_env_int _ tl }
1491 }
```

Now, we prepare the token list for the instructions that we will have to write on the aux file at the end of the environment.

```
1492 \tl_gclear:N \g_@@_aux_tl
1493 \tl_if_empty:NF \g_@@_code_before_tl
1494 {
1495   \bool_set_true:N \l_@@_code_before_bool
1496   \tl_put_right:NV \l_@@_code_before_tl \g_@@_code_before_tl
1497 }
```

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`.

```
1498 \bool_if:NTF \l_@@_NiceArray_bool
1499 { \keys_set:nn { NiceMatrix / NiceArray } }
1500 { \keys_set:nn { NiceMatrix / pNiceArray } }
1501 { #3 , #5 }

1502 \tl_if_empty:NF \l_@@_rules_color_tl
1503 { \exp_after:wN \@@_set_CT@arc@: \l_@@_rules_color_tl \q_stop }
```

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 body

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:.`

```

1504   \IfBooleanTF { #6 } \@@_pre_array_i:w \@@_pre_array:
1505   }
1506   {
1507     \bool_if:NTF \l_@@_light_syntax_bool
1508     { \use:c { end @@-light-syntax } }
1509     { \use:c { end @@-normal-syntax } }
1510     \c_math_toggle_token
1511     \skip_horizontal:N \l_@@_right_margin_dim
1512     \skip_horizontal:N \l_@@_extra_right_margin_dim
1513     \hbox_set_end:

```

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

If the user has used the key `width` without any column `X`, we raise an error.

```

1514   \bool_if:NT \l_@@_width_used_bool
1515   {
1516     \int_compare:nNnT \g_@@_total_X_weight_int = 0
1517     { \@@_error:n { width~without~X~columns } }
1518   }

```

Now, if there is at least one `X`-column in the environment, we compute the width that those columns will have (in the next compilation). In fact, `\l_@@_X_columns_dim` will be the width of a column of weight 1. For a `X`-column of weight  $n$ , the width will be `\l_@@_X_columns_dim` multiplied by  $n$ .

```

1519   \int_compare:nNnT \g_@@_total_X_weight_int > 0
1520   {
1521     \tl_gput_right:Nx \g_@@_aux_tl
1522     {
1523       \bool_set_true:N \l_@@_X_columns_aux_bool
1524       \dim_set:Nn \l_@@_X_columns_dim
1525       {
1526         \dim_compare:nNnTF
1527         {
1528           \dim_abs:n
1529           { \l_@@_width_dim - \box_wd:N \l_@@_the_array_box }
1530         }
1531         <
1532         { 0.001 pt }
1533         { \dim_use:N \l_@@_X_columns_dim }
1534         {
1535           \dim_eval:n
1536           {
1537             ( \l_@@_width_dim - \box_wd:N \l_@@_the_array_box )
1538             / \int_use:N \g_@@_total_X_weight_int
1539             + \l_@@_X_columns_dim
1540           }
1541         }
1542       }
1543     }
1544   }

```

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).

```

1545   \int_compare:nNnT \l_@@_last_row_int > { -2 }
1546   {
1547     \bool_if:NF \l_@@_last_row_without_value_bool
1548     {
1549       \int_compare:nNnF \l_@@_last_row_int = \c_iRow
1550       {

```

```

1551         \@@_error:n { Wrong~last~row }
1552         \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1553     }
1554 }
1555 }

```

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>61</sup>

```

1556     \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1557     \bool_if:nTF \g_@@_last_col_found_bool
1558     { \int_gdecr:N \c@jCol }
1559     {
1560         \int_compare:nNnT \l_@@_last_col_int > { -1 }
1561         { \@@_error:n { last~col~not~used } }
1562     }

```

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

```

1563     \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1564     \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. 120).

```

1565     \int_compare:nNnT \l_@@_first_col_int = 0
1566     {
1567         \skip_horizontal:N \col@sep
1568         \skip_horizontal:N \g_@@_width_first_col_dim
1569     }

```

The construction of the real box is different when `\l_@@_NiceArray_bool` is true (`{NiceArray}` or `{NiceTabular}`) and in the other environments because, in `{NiceArray}` or `{NiceTabular}`, we have no delimiter to put (but we have tabular notes to put). We begin with this case.

```

1570     \bool_if:nTF \l_@@_NiceArray_bool
1571     {
1572         \str_case:VnF \l_@@_baseline_tl
1573         {
1574             b \@@_use_arraybox_with_notes_b:
1575             c \@@_use_arraybox_with_notes_c:
1576         }
1577         \@@_use_arraybox_with_notes:
1578     }

```

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).

```

1579     {
1580         \int_compare:nNnTF \l_@@_first_row_int = 0
1581         {
1582             \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1583             \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1584         }
1585         { \dim_zero:N \l_tmpa_dim }

```

We compute `\l_tmpb_dim` which is the total height of the “last row” below the array (when the key `last-row` is used). A value of `-2` for `\l_@@_last_row_int` means that there is no “last row”.<sup>62</sup>

```

1586     \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1587     {
1588         \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1589         \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1590     }
1591     { \dim_zero:N \l_tmpb_dim }

```

<sup>61</sup>We remind that the potential “first column” (exterior) has the number 0.

<sup>62</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).

```

1592     \hbox_set:Nn \l_tmpa_box
1593     {
1594         \c_math_toggle_token
1595         \tl_if_empty:NF \l_@@_delimiters_color_tl
1596         { \color { \l_@@_delimiters_color_tl } }
1597         \exp_after:wN \left \g_@@_left_delim_tl
1598         \vcenter
1599         {

```

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.

```

1600         \skip_vertical:N -\l_tmpa_dim
1601         \hbox
1602         {
1603             \bool_if:NTF \l_@@_NiceTabular_bool
1604             { \skip_horizontal:N -\tabcolsep }
1605             { \skip_horizontal:N -\arraycolsep }
1606             \@@_use_arraybox_with_notes_c:
1607             \bool_if:NTF \l_@@_NiceTabular_bool
1608             { \skip_horizontal:N -\tabcolsep }
1609             { \skip_horizontal:N -\arraycolsep }
1610         }

```

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

```

1611         \skip_vertical:N -\l_tmpb_dim
1612     }

```

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.

```

1613         \tl_if_empty:NF \l_@@_delimiters_color_tl
1614         { \color { \l_@@_delimiters_color_tl } }
1615         \exp_after:wN \right \g_@@_right_delim_tl
1616         \c_math_toggle_token
1617     }

```

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.

```

1618     \bool_if:NTF \l_@@_delimiters_max_width_bool
1619     {
1620         \@@_put_box_in_flow_bis:nn
1621         \g_@@_left_delim_tl \g_@@_right_delim_tl
1622     }
1623     \@@_put_box_in_flow:
1624 }

```

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. 121).

```

1625     \bool_if:NT \g_@@_last_col_found_bool
1626     {
1627         \skip_horizontal:N \g_@@_width_last_col_dim
1628         \skip_horizontal:N \col@sep
1629     }
1630     \bool_if:NF \l_@@_Matrix_bool
1631     {
1632         \int_compare:nNnT \c@jCol < \g_@@_static_num_of_col_int
1633         { \@@_error:n { columns-not-used } }
1634     }
1635     \group_begin:
1636     \globaldefs = 1
1637     \@@_msg_redirect_name:nn { columns-not-used } { error }
1638     \group_end:
1639     \@@_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.

```
1640 \egroup
```

We want to write on the aux file all the informations corresponding to the current environment.

```
1641 \iow_now:Nn \@mainaux { \ExplSyntaxOn }
1642 \iow_now:Nn \@mainaux { \char_set_catcode_space:n { 32 } }
1643 \iow_now:Nx \@mainaux
1644 {
1645     \tl_gset:cn { c_@@_ \int_use:N \g_@@_env_int _ tl }
1646     { \exp_not:V \g_@@_aux_tl }
1647 }
1648 \iow_now:Nn \@mainaux { \ExplSyntaxOff }

1649 \bool_if:NT \c_@@_footnote_bool \endsavenotes
1650 }
```

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.

```
1651 \cs_new_protected:Npn \@@_construct_preamble:
1652 {
```

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

Since we use the tools of `array` to do this expansion, we will have a programmation which is not in the style of the L3 programming layer.

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.

```
1653 \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).

```
1654 \bool_if:NF \l_@@_Matrix_bool
1655 {
1656     \@@_newcolumntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1657     \@@_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 the L3 programming layer).

```
1658     \exp_args:NV \@temptokena \g_@@_preamble_tl
```

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

```
1659     \@tempwatrue
```

The following line actually does the expansion (it’s has been copied from `array.sty`). The expanded version is still in `\@temptokena`.

```
1660     \@whiles w \if@tempwa \fi { \@tempwafalse \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`.

```

1661     \int_gzero:N \c@jCol
1662     \tl_gclear:N \g_@@_preamble_tl
\g_tmpb_bool will be raised if you have a | at the end of the preamble.
1663     \bool_gset_false:N \g_tmpb_bool
1664     \tl_if_eq:NnTF \l_@@_vlines_clist { all }
1665     {
1666         \tl_gset:Nn \g_@@_preamble_tl
1667         { ! { \skip_horizontal:N \arrayrulewidth } }
1668     }
1669     {
1670         \clist_if_in:NnT \l_@@_vlines_clist 1
1671         {
1672             \tl_gset:Nn \g_@@_preamble_tl
1673             { ! { \skip_horizontal:N \arrayrulewidth } }
1674         }
1675     }

```

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 `vlsim`).

```

1676     \seq_clear:N \g_@@_cols_vlsim_seq

```

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

```

1677     \int_zero:N \l_tmpa_int

```

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

```

1678     \exp_after:wN \@@_patch_preamble:n \the \temptokena \q_stop
1679     \int_gset_eq:NN \g_@@_static_num_of_col_int \c@jCol
1680 }

```

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

```

1681     \bool_if:NT \l_@@_colortbl_like_bool
1682     {
1683         \regex_replace_all:NnN
1684         \c_@@_columncolor_regex
1685         { \c { @@_columncolor_preamble } }
1686         \g_@@_preamble_tl
1687     }

```

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

```

1688     \group_end:

```

If there was delimiters at the beginning or at the end of the preamble, the environment `{NiceArray}` is transformed into an environment `{xNiceMatrix}`.

```

1689     \bool_lazy_or:nnT
1690     { ! \str_if_eq_p:Vn \g_@@_left_delim_tl { . } }
1691     { ! \str_if_eq_p:Vn \g_@@_right_delim_tl { . } }
1692     { \bool_set_false:N \l_@@_NiceArray_bool }

```

We want to remind whether there is a specifier `|` at the end of the preamble.

```

1693     \bool_if:NT \g_tmpb_bool { \bool_set_true:N \l_@@_bar_at_end_of_pream_bool }

```

We complete the preamble with the potential “exterior columns” (on both sides).

```

1694     \int_compare:nNnTF \l_@@_first_col_int = 0
1695     { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1696     {
1697         \bool_lazy_all:nT

```

```

1698     {
1699         \l_@@_NiceArray_bool
1700         { \bool_not_p:n \l_@@_NiceTabular_bool }
1701         { \tl_if_empty_p:N \l_@@_vlines_clist }
1702         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1703     }
1704     { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1705 }
1706 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1707 { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }
1708 {
1709     \bool_lazy_all:nT
1710     {
1711         \l_@@_NiceArray_bool
1712         { \bool_not_p:n \l_@@_NiceTabular_bool }
1713         { \tl_if_empty_p:N \l_@@_vlines_clist }
1714         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1715     }
1716     { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1717 }

```

We add a last column to raise a good error message when the user puts 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`).

```

1718     \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1719     {
1720         \tl_gput_right:Nn \g_@@_preamble_tl
1721         { > { \@@_error_too_much_cols: } 1 }
1722     }
1723 }

1724 \cs_new_protected:Npn \@@_patch_preamble:n #1
1725 {
1726     \str_case:nnF { #1 }
1727     {
1728         c { \@@_patch_preamble_i:n #1 }
1729         l { \@@_patch_preamble_i:n #1 }
1730         r { \@@_patch_preamble_i:n #1 }
1731         > { \@@_patch_preamble_ii:nn #1 }
1732         ! { \@@_patch_preamble_ii:nn #1 }
1733         @ { \@@_patch_preamble_ii:nn #1 }
1734         | { \@@_patch_preamble_iii:n #1 }
1735         p { \@@_patch_preamble_iv:n #1 }
1736         b { \@@_patch_preamble_iv:n #1 }
1737         m { \@@_patch_preamble_iv:n #1 }
1738         \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1739         \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1740         \@@_S: { \@@_patch_preamble_vi:n }
1741         ( { \@@_patch_preamble_vii:nn #1 }
1742         [ { \@@_patch_preamble_vii:nn #1 }
1743         \{ { \@@_patch_preamble_vii:nn #1 }
1744         ) { \@@_patch_preamble_viii:nn #1 }
1745         ] { \@@_patch_preamble_viii:nn #1 }
1746         \} { \@@_patch_preamble_viii:nn #1 }
1747         X { \@@_patch_preamble_ix:n }
1748         \q_stop { }
1749     }
1750     {
1751         \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1752         { \@@_patch_preamble_xi:n #1 }
1753         {
1754             \str_if_eq:VnTF \l_@@_letter_vlism_tl { #1 }
1755             {

```

```

1756         \seq_gput_right:Nx \g_@@_cols_vlism_seq
1757         { \int_eval:n { \c@jCol + 1 } }
1758     \tl_gput_right:Nx \g_@@_preamble_tl
1759     { \exp_not:N ! { \skip_horizontal:N \arrayrulewidth } }
1760     \@@_patch_preamble:n
1761 }
1762 {
1763     \bool_lazy_and:nnTF
1764     { \str_if_eq_p:nn { : } { #1 } }
1765     \c_@@_arydshln_loaded_bool
1766     {
1767         \tl_gput_right:Nn \g_@@_preamble_tl { : }
1768         \@@_patch_preamble:n
1769     }
1770     { \@@_fatal:nn { unknown~column~type } { #1 } }
1771 }
1772 }
1773 }
1774 }

```

For c, l and r

```

1775 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1776 {
1777     \tl_gput_right:Nn \g_@@_preamble_tl
1778     {
1779         > { \@@_cell_begin:w \str_set:Nn \l_@@_hpos_cell_str { #1 } }
1780         #1
1781         < \@@_cell_end:
1782     }

```

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

```

1783     \int_gincr:N \c@jCol
1784     \@@_patch_preamble_x:n
1785 }

```

For >, ! and @

```

1786 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1787 {
1788     \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
1789     \@@_patch_preamble:n
1790 }

```

For |

```

1791 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1792 {

```

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

```

1793     \int_incr:N \l_tmpa_int
1794     \@@_patch_preamble_iii_i:n
1795 }
1796 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1797 {
1798     \str_if_eq:nnTF { #1 } |
1799     { \@@_patch_preamble_iii:n | }
1800     {
1801         \tl_gput_right:Nx \g_@@_preamble_tl
1802         {
1803             \exp_not:N !
1804             {
1805                 \skip_horizontal:n
1806                 {
1807                     \dim_eval:n
1808                     {

```

```

1809         \arrayrulewidth * \l_tmpa_int
1810         + \doublerulesep * ( \l_tmpa_int - 1)
1811     }
1812 }
1813 }
1814 }
1815 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1816 {
1817     \@@_vline:nnnn
1818     { \@@_succ:n \c@jCol } { \int_use:N \l_tmpa_int } { 1 } { }
1819 }
1820 \int_zero:N \l_tmpa_int
1821 \str_if_eq:nnT { #1 } { \q_stop } { \bool_gset_true:N \g_tmpb_bool }
1822 \@@_patch_preamble:n #1
1823 }
1824 }
1825 \bool_new:N \l_@@_bar_at_end_of_pream_bool

```

The specifier *p* (and also the specifiers *m* and *b*) have an optional argument between square brackets for a list of *key-value* pairs. Here are the corresponding keys. This set of keys will also be used by the *X* columns.

```

1826 \keys_define:nn { WithArrows / p-column }
1827 {
1828     r .code:n = \str_set:Nn \l_@@_hpos_col_str { r } ,
1829     r .value_forbidden:n = true ,
1830     c .code:n = \str_set:Nn \l_@@_hpos_col_str { c } ,
1831     c .value_forbidden:n = true ,
1832     l .code:n = \str_set:Nn \l_@@_hpos_col_str { l } ,
1833     l .value_forbidden:n = true ,
1834     si .code:n = \str_set:Nn \l_@@_hpos_col_str { si } ,
1835     si .value_forbidden:n = true ,
1836     p .code:n = \str_set:Nn \l_@@_vpos_col_str { p } ,
1837     p .value_forbidden:n = true ,
1838     t .meta:n = p ,
1839     m .code:n = \str_set:Nn \l_@@_vpos_col_str { m } ,
1840     m .value_forbidden:n = true ,
1841     b .code:n = \str_set:Nn \l_@@_vpos_col_str { b } ,
1842     b .value_forbidden:n = true ,
1843 }

```

For *p*, *b* and *m*. The argument *#1* is that value : *p*, *b* or *m*.

```

1844 \cs_new_protected:Npn \@@_patch_preamble_iv:n #1
1845 {
1846     \str_set:Nn \l_@@_vpos_col_str { #1 }

```

Now, you look for a potential character [ after the letter of the specifier (for the options).

```

1847     \@@_patch_preamble_iv_i:n
1848 }
1849 \cs_new_protected:Npn \@@_patch_preamble_iv_i:n #1
1850 {
1851     \str_if_eq:nnTF { #1 } { [ ]
1852         { \@@_patch_preamble_iv_ii:w [ ]
1853             { \@@_patch_preamble_iv_ii:w [ ] { #1 } }
1854         }
1855     \cs_new_protected:Npn \@@_patch_preamble_iv_ii:w [ #1 ]
1856     { \@@_patch_preamble_iv_iii:nn { #1 } }

```

**#1** is the optional argument of the specifier (a list of *key-value* pairs).

**#2** is the mandatory argument of the specifier: the width of the column.

```

1857 \cs_new_protected:Npn \@@_patch_preamble_iv_iii:nn #1 #2
1858 {

```

The possible values of `\l_@@_hpos_col_str` are `j` (for *justified* which is the initial value), `l`, `c` and `r` (when the user has used the corresponding key in the optional argument of the specifier).

```

1859 \str_set:Nn \l_@@_hpos_col_str { j }
1860 \keys_set:nn { WithArrows / p-column } { #1 }
1861 \@@_patch_preamble_iv_iv:n { #2 }
1862 }

```

The argument is the width of the column.

```

1863 \cs_new_protected:Npn \@@_patch_preamble_iv_iv:n #1
1864 {
1865   \use:x
1866   {
1867     \@@_patch_preamble_iv_v:nnnnnn
1868     { \str_if_eq:VnTF \l_@@_vpos_col_str { p } { t } { b } }
1869     { \dim_eval:n { #1 } }
1870   }

```

The parameter `\l_@@_hpos_col_str` (as `\l_@@_vpos_col_str`) exists only during the construction of the preamble. During the composition of the array itself, you will have, in each cell, the parameter `\l_@@_hpos_cell_str` which will provide the horizontal alignment of the column to which belongs the cell.

```

1871 \str_if_eq:VnTF \l_@@_hpos_col_str j
1872 { \str_set:Nn \exp_not:N \l_@@_hpos_cell_str { c } }
1873 {
1874   \str_set:Nn \exp_not:N \l_@@_hpos_cell_str
1875   { \l_@@_hpos_col_str }
1876 }
1877 \str_case:Vn \l_@@_hpos_col_str
1878 {
1879   c { \exp_not:N \centering }
1880   l { \exp_not:N \raggedright }
1881   r { \exp_not:N \raggedleft }
1882 }
1883 }
1884 { \str_if_eq:VnT \l_@@_vpos_col_str { m } \@@_center_cell_box: }
1885 { \str_if_eq:VnT \l_@@_hpos_col_str { si } \siunitx_cell_begin:w }
1886 { \str_if_eq:VnT \l_@@_hpos_col_str { si } \siunitx_cell_end: }
1887 }

```

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

```

1888 \int_gincr:N \c@jCol
1889 \@@_patch_preamble_x:n
1890 }

```

**#1** is the optional argument of `{minipage}`: `t` of `b`. Indeed, for the columns of type `m`, we use the value `b` here because there is a special post-action in order to center vertically the box (see **#4**).

**#2** is the width of the `{minipage}`, that is to say also the width of the column.

**#3** is the coding for the horizontal position of the content of the cell (`\centering`, `\raggedright`, `\raggedleft` or nothing).

**#4** is an extra-code which contains `\l_@@_center_cell_box` (when the column is a `m` column) or nothing (in the other cases).

**#5** is a code put just before the `c`.

**#6** is a code put just after the `c`.

```

1891 \cs_new_protected:Npn \@@_patch_preamble_iv_v:nnnnnn #1 #2 #3 #4 #5 #6
1892 {
1893   \tl_gput_right:Nn \g_@@_preamble_tl
1894   {
1895     > {

```

The parameter `\l_@@_col_width_dim`, which is the width of the current column, will be available in each cell of the column. It will be used by the mono-column blocks.

```

1896 \dim_set:Nn \l_@@_col_width_dim { #2 }
1897 \@@_cell_begin:w
1898 \begin { minipage } [ #1 ] { #2 }

```

The following lines have been taken from `array.sty`.

```

1899     \everypar
1900     {
1901         \vrule height \box_ht:N \@arstrutbox width \c_zero_dim
1902         \everypar { }
1903     }

```

Now, the potential code for the horizontal position of the content of the cell (`\centering`, `\raggedright`, `\raggedleft` or nothing).

```

1904         #3

```

The following code is to allow something like `\centering` in `\RowStyle`.

```

1905         \g_@@_row_style_tl
1906         \arraybackslash
1907         #5
1908     }
1909     c
1910     < {
1911         #6

```

The following line has been taken from `array.sty`.

```

1912         \@finalstrut \@arstrutbox
1913         \end { minipage }

```

If the letter in the preamble is `m`, `#3` will be equal to `\@@_center_cell_box:` (see just below).

```

1914         #4
1915         \@@_cell_end:
1916     }
1917 }
1918 }

```

The following command will be used in `m`-columns in order to center vertically the box. In fact, despite its name, the command does not always center the cell. Indeed, if there is only one row in the cell, it should not be centered vertically. It's not possible to know the number of rows of the cell. However, we consider (as in `array`) that if the height of the cell is no more that the height of `\@arstrutbox`, there is only one row.

```

1919 \cs_new_protected:Npn \@@_center_cell_box:
1920 {

```

By putting instructions in `\g_@@_post_action_cell_tl`, we require a post-action of the box `\l_@@_cell_box`.

```

1921     \tl_gput_right:Nn \g_@@_post_action_cell_tl
1922     {
1923         \int_compare:nNnT
1924         { \box_ht:N \l_@@_cell_box }
1925         >
1926         { \box_ht:N \@arstrutbox }
1927         {
1928             \hbox_set:Nn \l_@@_cell_box
1929             {
1930                 \box_move_down:nn
1931                 {
1932                     ( \box_ht:N \l_@@_cell_box - \box_ht:N \@arstrutbox
1933                     + \baselineskip ) / 2
1934                 }
1935                 { \box_use:N \l_@@_cell_box }
1936             }
1937         }
1938     }
1939 }

```

For `w` and `W`

```

1940 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1941 {

```

```

1942 \tl_gput_right:Nn \g_@@_preamble_tl
1943 {
1944   > {

```

The parameter `\l_@@_col_width_dim`, which is the width of the current column, will be available in each cell of the column. It will be used by the mono-column blocks.

```

1945     \dim_set:Nn \l_@@_col_width_dim { #4 }
1946     \hbox_set:Nw \l_@@_cell_box
1947     \@@_cell_begin:w
1948     \str_set:Nn \l_@@_hpos_cell_str { #3 }
1949   }
1950   c
1951   < {
1952     \@@_cell_end:
1953     #1
1954     \hbox_set_end:
1955     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1956     \@@_adjust_size_box:
1957     \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
1958   }
1959 }

```

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

```

1960 \int_gincr:N \c@jCol
1961 \@@_patch_preamble_x:n
1962 }

```

For `\@@_S:`. If the user has used `S[...]`, `S` has been replaced by `\@@_S:` during the first expansion of the preamble (done with the tools of standard LaTeX and array).

```

1963 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1964 {
1965   \str_if_eq:nnTF { #1 } { [ ] }
1966   { \@@_patch_preamble_vi_i:w [ ] }
1967   { \@@_patch_preamble_vi_i:w [ ] { #1 } }
1968 }
1969 \cs_new_protected:Npn \@@_patch_preamble_vi_i:w [ #1 ]
1970 { \@@_patch_preamble_vi_ii:n { #1 } }

```

For version of `siunitx` at least equal to 3.0, the adaptation is different from previous ones. We test the version of `siunitx` by the existence of the control sequence `\siunitx_cell_begin:w`. When we will decide that only the previous posterior to 3.0 are supported by `nicematrix`, we will delete the second definition of `\@@_patch_preamble_vi_ii:n`.

```

1971 \AtBeginDocument
1972 {
1973   \cs_if_exist:NTF \siunitx_cell_begin:w
1974   {
1975     \cs_new_protected:Npn \@@_patch_preamble_vi_ii:n #1
1976     {
1977       \tl_gput_right:Nn \g_@@_preamble_tl
1978       {
1979         > {
1980           \@@_cell_begin:w
1981           \keys_set:nn { siunitx } { #1 }
1982           \siunitx_cell_begin:w
1983         }
1984         c
1985         < { \siunitx_cell_end: \@@_cell_end: }
1986       }

```

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

```

1987     \int_gincr:N \c@jCol
1988     \@@_patch_preamble_x:n
1989   }
1990 }

```

```

1991 {
1992   \cs_new_protected:Npn \@@_patch_preamble_vi_ii:n #1
1993   {
1994     \tl_gput_right:Nn \g_@@_preamble_tl
1995     {
1996       > { \@@_cell_begin:w \c_@@_table_collect_begin_tl S { #1 } }
1997       c
1998       < { \c_@@_table_print_tl \@@_cell_end: }
1999     }
2000     \int_gincr:N \c@jCol
2001     \@@_patch_preamble_x:n
2002   }
2003 }
2004 }

```

For (, [ and \{.

```

2005 \cs_new_protected:Npn \@@_patch_preamble_vii:nn #1 #2
2006 {
2007   \bool_if:NT \l_@@_small_bool { \@@_fatal:n { Delimiter~with~small } }

```

If we are before the column 1 and not in {NiceArray}, we reserve space for the left delimiter.

```

2008   \int_compare:nNnTF \c@jCol = \c_zero_int
2009   {
2010     \str_if_eq:VnTF \g_@@_left_delim_tl { . }
2011     {

```

In that case, in fact, the first letter of the preamble must be considered as the left delimiter of the array.

```

2012       \tl_gset:Nn \g_@@_left_delim_tl { #1 }
2013       \tl_gset:Nn \g_@@_right_delim_tl { . }
2014       \@@_patch_preamble:n #2
2015     }
2016   {
2017     \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } }
2018     \@@_patch_preamble_vii_i:nn { #1 } { #2 }
2019   }
2020 }
2021 { \@@_patch_preamble_vii_i:nn { #1 } { #2 } }
2022 }
2023 \cs_new_protected:Npn \@@_patch_preamble_vii_i:nn #1 #2
2024 {
2025   \tl_gput_right:Nx \g_@@_internal_code_after_tl
2026   { \@@_delimiter:nnn #1 { \@@_succ:n \c@jCol } \c_true_bool }
2027   \tl_if_in:nnTF { ( [ \{ ) ] \} } { #2 }
2028   {
2029     \@@_error:nn { delimiter~after~opening } { #2 }
2030     \@@_patch_preamble:n
2031   }
2032   { \@@_patch_preamble:n #2 }
2033 }

```

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 if the environment is {NiceArray}).

```

2034 \cs_new_protected:Npn \@@_patch_preamble_viii:nn #1 #2
2035 {
2036   \bool_if:NT \l_@@_small_bool { \@@_fatal:n { Delimiter~with~small } }
2037   \tl_if_in:nnTF { ) ] \} } { #2 }
2038   { \@@_patch_preamble_viii_i:nnn #1 #2 }
2039   {
2040     \tl_if_eq:nnTF { \q_stop } { #2 }
2041     {

```



```

2042 \str_if_eq:VnTF \g_@@_right_delim_tl { . }
2043 { \tl_gset:Nn \g_@@_right_delim_tl { #1 } }
2044 {
2045 \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } }
2046 \tl_gput_right:Nx \g_@@_internal_code_after_tl
2047 { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
2048 \@@_patch_preamble:n #2
2049 }
2050 }
2051 {
2052 \tl_if_in:nnT { ( [ \{ } { #2 }
2053 { \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } } }
2054 \tl_gput_right:Nx \g_@@_internal_code_after_tl
2055 { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
2056 \@@_patch_preamble:n #2
2057 }
2058 }
2059 }
2060 \cs_new_protected:Npn \@@_patch_preamble_viii_i:nnn #1 #2 #3
2061 {
2062 \tl_if_eq:nnTF { \q_stop } { #3 }
2063 {
2064 \str_if_eq:VnTF \g_@@_right_delim_tl { . }
2065 {
2066 \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } }
2067 \tl_gput_right:Nx \g_@@_internal_code_after_tl
2068 { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
2069 \tl_gset:Nn \g_@@_right_delim_tl { #2 }
2070 }
2071 {
2072 \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } }
2073 \tl_gput_right:Nx \g_@@_internal_code_after_tl
2074 { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
2075 \@@_error:nn { double-closing-delimiter } { #2 }
2076 }
2077 }
2078 {
2079 \tl_gput_right:Nx \g_@@_internal_code_after_tl
2080 { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
2081 \@@_error:nn { double-closing-delimiter } { #2 }
2082 \@@_patch_preamble:n #3
2083 }
2084 }

```

For the case of a letter X. This specifier may take in an optional argument (between square brackets). That's why we test whether there is a [ after the letter X.

```

2085 \cs_new_protected:Npn \@@_patch_preamble_ix:n #1
2086 {
2087 \str_if_eq:nnTF { #1 } { [ ]
2088 { \@@_patch_preamble_ix_i:w [ ]
2089 { \@@_patch_preamble_ix_i:w [ ] #1 }
2090 }
2091 \cs_new_protected:Npn \@@_patch_preamble_ix_i:w [ #1 ]
2092 { \@@_patch_preamble_ix_ii:n { #1 } }

```

#1 is the optional argument of the X specifier (a list of *key-value* pairs).

The following set of keys is for the specifier X in the preamble of the array. Such specifier may have as keys all the keys of { WithArrows / p-column } but also a key as 1, 2, 3, etc. The following set of keys will be used to retrieve that value (in the counter \l\_@@\_weight\_int).

```

2093 \keys_define:nn { WithArrows / X-column }
2094 { unknown .code:n = \int_set:Nn \l_@@_weight_int { \l_keys_key_str } }

```

In the following command, #1 is the list of the options of the specifier X.

```
2095 \cs_new_protected:Npn \@@_patch_preamble_ix_ii:n #1
2096 {
```

The possible values of \l\_@@\_hpos\_col\_str are j (for *justified* which is the initial value), l, c and r (when the user has used the corresponding key in the optional argument of the specifier X).

```
2097 \str_set:Nn \l_@@_hpos_col_str { j }
```

The possible values of \l\_@@\_vpos\_col\_str are p (the initial value), m and b (when the user has used the corresponding key in the optional argument of the specifier X).

```
2098 \tl_set:Nn \l_@@_vpos_col_str { p }
```

The integer \l\_@@\_weight\_int will be the weight of the X column (the initial value is 1). The user may specify a different value (such as 2, 3, etc.) by putting that value in the optional argument of the specifier. The weights of the X columns are used in the computation of the actual width of those columns as in tabu of tabularray.

```
2099 \int_zero_new:N \l_@@_weight_int
2100 \int_set:Nn \l_@@_weight_int { 1 }
2101 \keys_set_known:nnN { WithArrows / p-column } { #1 } \l_tmpa_tl
2102 \keys_set:nV { WithArrows / X-column } \l_tmpa_tl
2103 \int_gadd:Nn \g_@@_total_X_weight_int \l_@@_weight_int
```

We test whether we know the width of the X-columns by reading the aux file (after the first compilation, the width of the X-columns is computed and written in the aux file).

```
2104 \bool_if:NTF \l_@@_X_columns_aux_bool
2105 { \@@_patch_preamble_iv_iv:n { \l_@@_weight_int \l_@@_X_columns_dim } }
2106 {
2107   \tl_gput_right:Nn \g_@@_preamble_tl
2108   {
2109     > {
2110       \@@_cell_begin:w
2111       \bool_set_true:N \l_@@_X_column_bool
```

The following code will nullify the box of the cell.

```
2112 \tl_gput_right:Nn \g_@@_post_action_cell_tl
2113 { \hbox_set:Nn \l_@@_cell_box { } }
```

We put a {minipage} to give to the user the ability to put a command such as \centering in the \RowStyle.

```
2114 \begin { minipage } { 5 cm } \arraybackslash
2115 }
2116 c
2117 < {
2118   \end { minipage }
2119   \@@_cell_end:
2120 }
2121 }
2122 \int_gincr:N \c@jCol
2123 \@@_patch_preamble_x:n
2124 }
2125 }
```

```
2126 \cs_new_protected:Npn \@@_patch_preamble_xi:n #1
2127 {
2128   \tl_gput_right:Nn \g_@@_preamble_tl
2129   { ! { \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.

```
2130 \tl_gput_right:Nx \g_@@_internal_code_after_tl
2131 { \@@_vdottedline:n { \int_use:N \c@jCol } }
2132 \@@_patch_preamble:n
2133 }
```

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.

```

2134 \cs_new_protected:Npn \@@_patch_preamble_x:n #1
2135 {
2136   \str_if_eq:nnTF { #1 } { < }
2137     \@@_patch_preamble_xii:n
2138     {
2139       \tl_if_eq:NnTF \l_@@_vlines_clist { all }
2140       {
2141         \tl_gput_right:Nn \g_@@_preamble_tl
2142           { ! { \skip_horizontal:N \arrayrulewidth } }
2143       }
2144       {
2145         \exp_args:NNx
2146         \clist_if_in:NnT \l_@@_vlines_clist { \@@_succ:n \c@jCol }
2147         {
2148           \tl_gput_right:Nn \g_@@_preamble_tl
2149             { ! { \skip_horizontal:N \arrayrulewidth } }
2150         }
2151       }
2152       \@@_patch_preamble:n { #1 }
2153     }
2154 }
2155 \cs_new_protected:Npn \@@_patch_preamble_xii:n #1
2156 {
2157   \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
2158   \@@_patch_preamble_x:n
2159 }

```

## The redefinition of `\multicolumn`

The following command must *not* be protected since it begins with `\multispan` (a TeX primitive).

```

2160 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
2161 {

```

The following lines are from the definition of `\multicolumn` in `array` (and *not* in standard LaTeX). The first line aims to raise an error if the user has put more than one column specifier in the preamble of `\multicolumn`.

```

2162   \multispan { #1 }
2163   \begingroup
2164   \cs_set:Npn \@addamp { \if@firstamp \@firstampfalse \else \@preamerr 5 \fi }

```

You do the expansion of the (small) preamble with the tools of `array`.

```

2165   \@temptokena = { #2 }
2166   \@tempswatrue
2167   \@whilesw \if@tempswa \fi { \@tempswafalse \the \NC@list }

```

Now, we patch the (small) preamble as we have done with the main preamble of the `array`.

```

2168   \tl_gclear:N \g_@@_preamble_tl
2169   \exp_after:wN \@@_patch_m_preamble:n \the \@temptokena \q_stop

```

The following lines are an adaptation of the definition of `\multicolumn` in `array`.

```

2170   \exp_args:NV \@mkpream \g_@@_preamble_tl
2171   \@addtopreamble \@empty
2172   \endgroup

```

Now, you do a treatment specific to `nicematrix` which has no equivalent in the original definition of `\multicolumn`.

```

2173 \int_compare:nNnT { #1 } > 1
2174 {
2175   \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
2176   { \int_use:N \c@iRow - \@@_succ:n \c@jCol }
2177   \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
2178   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
2179   {
2180     { \int_use:N \c@iRow }
2181     { \int_eval:n { \c@jCol + 1 } }
2182     { \int_use:N \c@iRow }
2183     { \int_eval:n { \c@jCol + #1 } }
2184   }
2185 }

```

The following lines were in the original definition of `\multicolumn`.

```

2186 \cs_set:Npn \@sharp { #3 }
2187 \@arstrut
2188 \@preamble
2189 \null

```

We add some lines.

```

2190 \int_gadd:Nn \c@jCol { #1 - 1 }
2191 \int_compare:nNnT \c@jCol > \g_@@_col_total_int
2192 { \int_gset_eq:NN \g_@@_col_total_int \c@jCol }
2193 \ignorespaces
2194 }

```

The following commands will patch the (small) preamble of the `\multicolumn`. All those commands have a `m` in their name to recall that they deal with the redefinition of `\multicolumn`.

```

2195 \cs_new_protected:Npn \@@_patch_m_preamble:n #1
2196 {
2197   \str_case:nnF { #1 }
2198   {
2199     c { \@@_patch_m_preamble_i:n #1 }
2200     l { \@@_patch_m_preamble_i:n #1 }
2201     r { \@@_patch_m_preamble_i:n #1 }
2202     > { \@@_patch_m_preamble_ii:nn #1 }
2203     ! { \@@_patch_m_preamble_ii:nn #1 }
2204     @ { \@@_patch_m_preamble_ii:nn #1 }
2205     | { \@@_patch_m_preamble_iii:n #1 }
2206     p { \@@_patch_m_preamble_iv:nnn t #1 }
2207     m { \@@_patch_m_preamble_iv:nnn c #1 }
2208     b { \@@_patch_m_preamble_iv:nnn b #1 }
2209     \@@_w: { \@@_patch_m_preamble_v:nnnn { } #1 }
2210     \@@_W: { \@@_patch_m_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
2211     \@@_true_c: { \@@_patch_m_preamble_vi:n #1 }
2212     \q_stop { }
2213   }
2214   { \@@_fatal:nn { unknown~column~type } { #1 } }
2215 }

```

For `c`, `l` and `r`

```

2216 \cs_new_protected:Npn \@@_patch_m_preamble_i:n #1
2217 {
2218   \tl_gput_right:Nn \g_@@_preamble_tl
2219   {
2220     > { \@@_cell_begin:w \str_set:Nn \l_@@_hpos_cell_str { #1 } }
2221     #1
2222     < \@@_cell_end:
2223   }

```

We test for the presence of a <.

```
2224 \@@_patch_m_preamble_x:n
2225 }
```

For >, ! and @

```
2226 \cs_new_protected:Npn \@@_patch_m_preamble_ii:nn #1 #2
2227 {
2228   \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
2229   \@@_patch_m_preamble:n
2230 }
```

For |

```
2231 \cs_new_protected:Npn \@@_patch_m_preamble_iii:n #1
2232 {
2233   \tl_gput_right:Nn \g_@@_preamble_tl { #1 }
2234   \@@_patch_m_preamble:n
2235 }
```

For p, m and b

```
2236 \cs_new_protected:Npn \@@_patch_m_preamble_iv:nnn #1 #2 #3
2237 {
2238   \tl_gput_right:Nn \g_@@_preamble_tl
2239   {
2240     > {
2241       \@@_cell_begin:w
2242       \begin { minipage } [ #1 ] { \dim_eval:n { #3 } }
2243       \mode_leave_vertical:
2244       \arraybackslash
2245       \vrule height \box_ht:N \@arstrutbox depth 0 pt width 0 pt
2246     }
2247     c
2248     < {
2249       \vrule height 0 pt depth \box_dp:N \@arstrutbox width 0 pt
2250       \end { minipage }
2251       \@@_cell_end:
2252     }
2253   }
```

We test for the presence of a <.

```
2254 \@@_patch_m_preamble_x:n
2255 }
```

For w and W

```
2256 \cs_new_protected:Npn \@@_patch_m_preamble_v:nnnn #1 #2 #3 #4
2257 {
2258   \tl_gput_right:Nn \g_@@_preamble_tl
2259   {
2260     > {
2261       \hbox_set:Nw \l_@@_cell_box
2262       \@@_cell_begin:w
2263       \str_set:Nn \l_@@_hpos_cell_str { #3 }
2264     }
2265     c
2266     < {
2267       \@@_cell_end:
2268       #1
2269       \hbox_set_end:
2270       \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2271       \@@_adjust_size_box:
2272       \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
2273     }
2274   }
```

We test for the presence of a <.

```
2275 \@@_patch_m_preamble_x:n
2276 }
```

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

```
2277 \cs_new_protected:Npn \@@_patch_m_preamble_vi:n #1
2278 {
2279 \tl_gput_right:Nn \g_@@_preamble_tl { c }
```

We test for the presence of a <.

```
2280 \@@_patch_m_preamble_x:n
2281 }
```

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.

```
2282 \cs_new_protected:Npn \@@_patch_m_preamble_x:n #1
2283 {
2284 \str_if_eq:nnTF { #1 } { < }
2285 \@@_patch_m_preamble_ix:n
2286 {
2287 \tl_if_eq:NnTF \l_@@_vlines_clist { all }
2288 {
2289 \tl_gput_right:Nn \g_@@_preamble_tl
2290 { ! { \skip_horizontal:N \arrayrulewidth } }
2291 }
2292 {
2293 \exp_args:NNx
2294 \clist_if_in:NnT \l_@@_vlines_clist { \@@_succ:n \c@jCol }
2295 {
2296 \tl_gput_right:Nn \g_@@_preamble_tl
2297 { ! { \skip_horizontal:N \arrayrulewidth } }
2298 }
2299 }
2300 \@@_patch_m_preamble:n { #1 }
2301 }
2302 }

2303 \cs_new_protected:Npn \@@_patch_m_preamble_ix:n #1
2304 {
2305 \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
2306 \@@_patch_m_preamble_x:n
2307 }
```

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).

```
2308 \cs_new_protected:Npn \@@_put_box_in_flow:
2309 {
2310 \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
2311 \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
2312 \tl_if_eq:NnTF \l_@@_baseline_tl { c }
2313 { \box_use_drop:N \l_tmpa_box }
2314 \@@_put_box_in_flow_i:
2315 }
```

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).

```
2316 \cs_new_protected:Npn \@@_put_box_in_flow_i:
2317 {
2318 \pgfpicture
2319 \@@_qpoint:n { row - 1 }
2320 \dim_gset_eq:NN \g_tmpa_dim \pgf@y
```

```

2321 \@@_qpoint:n { row - \@@_succ:n \c@iRow }
2322 \dim_gadd:Nn \g_tmpa_dim \pgf@y
2323 \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).

```

2324 \str_if_in:NnTF \l_@@_baseline_tl { line- }
2325 {
2326   \int_set:Nn \l_tmpa_int
2327   {
2328     \str_range:Nnn
2329     \l_@@_baseline_tl
2330     6
2331     { \tl_count:V \l_@@_baseline_tl }
2332   }
2333   \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
2334 }
2335 {
2336   \str_case:VnF \l_@@_baseline_tl
2337   {
2338     { t } { \int_set:Nn \l_tmpa_int 1 }
2339     { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
2340   }
2341   { \int_set:Nn \l_tmpa_int \l_@@_baseline_tl }
2342   \bool_lazy_or:nnT
2343   { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
2344   { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
2345   {
2346     \@@_error:n { bad~value~for~baseline }
2347     \int_set:Nn \l_tmpa_int 1
2348   }
2349   \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

We take into account the position of the mathematical axis.

```

2350 \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
2351 }
2352 \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

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

```

2353 \endpgfpicture
2354 \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
2355 \box_use_drop:N \l_tmpa_box
2356 }

```

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).

```

2357 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
2358 {

```

With an environment `{Matrix}`, you want to remove the exterior `\arraycolsep` but we don't know the number of columns (since there is no preamble) and that's why we can't put `@{}` at the end of the preamble. That's why we remove a `\arraycolsep` now.

```

2359 \bool_lazy_and:nnT \l_@@_Matrix_bool \l_@@_NiceArray_bool
2360 {
2361   \box_set_wd:Nn \l_@@_the_array_box
2362   { \box_wd:N \l_@@_the_array_box - \arraycolsep }
2363 }

```

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

```

2364 \begin { minipage } [ t ] { \box_wd:N \l_@@_the_array_box }

```

The `\hbox` avoids that the `pgfpicture` inside `\@@_draw_blocks` adds a extra vertical space before the notes.

```

2365     \hbox:n
2366     {
2367         \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.

```

2368         \@@_create_extra_nodes:
2369         \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
2370     }
2371     \bool_lazy_or:nnT
2372     { \int_compare_p:nNn \c@tabularnote > 0 }
2373     { ! \tl_if_empty_p:V \l_@@_tabularnote_tl }
2374     \@@_insert_tabularnotes:
2375     \end { minipage }
2376 }
2377 \cs_new_protected:Npn \@@_insert_tabularnotes:
2378 {
2379     \skip_vertical:N 0.65ex

```

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

```

2380     \group_begin:
2381     \l_@@_notes_code_before_tl
2382     \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.

```

2383     \int_compare:nNnT \c@tabularnote > 0
2384     {
2385         \bool_if:NTF \l_@@_notes_para_bool
2386         {
2387             \begin { tabularnotes* }
2388             \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
2389             \end { tabularnotes* }

```

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

```

2390         \par
2391     }
2392     {
2393         \tabularnotes
2394         \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
2395         \endtabularnotes
2396     }
2397 }
2398 \unskip
2399 \group_end:
2400 \bool_if:NT \l_@@_notes_bottomrule_bool
2401 {
2402     \bool_if:NTF \c_@@_booktabs_loaded_bool
2403     {

```

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

```

2404         \skip_vertical:N \aboverulesep
\CT@arc@ is the specification of color defined by colortbl but you use it even if colortbl is not loaded.
2405         { \CT@arc@ \hrule height \heavyrulewidth }
2406     }
2407     { \@@_error:n { bottomrule~without~booktabs } }
2408 }
2409 \l_@@_notes_code_after_tl

```



```

2410 \seq_gclear:N \g_@@_tabularnotes_seq
2411 \int_gzero:N \c@tabularnote
2412 }

```

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.

```

2413 \cs_new_protected:Npn \@@_use_arraybox_with_notes_b:
2414 {
2415   \pgfpicture
2416     \@@_qpoint:n { row - 1 }
2417     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
2418     \@@_qpoint:n { row - \int_use:N \c@iRow - base }
2419     \dim_gsub:Nn \g_tmpa_dim \pgf@y
2420   \endpgfpicture
2421   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
2422   \int_compare:nNnT \l_@@_first_row_int = 0
2423   {
2424     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
2425     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
2426   }
2427   \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
2428 }

```

Now, the general case.

```

2429 \cs_new_protected:Npn \@@_use_arraybox_with_notes:
2430 {

```

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

```

2431   \tl_if_eq:NnT \l_@@_baseline_tl { t }
2432   { \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.

```

2433   \pgfpicture
2434   \@@_qpoint:n { row - 1 }
2435   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
2436   \str_if_in:NnTF \l_@@_baseline_tl { line- }
2437   {
2438     \int_set:Nn \l_tmpa_int
2439     {
2440       \str_range:Nnn
2441         \l_@@_baseline_tl
2442         6
2443       { \tl_count:V \l_@@_baseline_tl }
2444     }
2445     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
2446   }
2447   {
2448     \int_set:Nn \l_tmpa_int \l_@@_baseline_tl
2449     \bool_lazy_or:nnT
2450     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
2451     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
2452     {
2453       \@@_error:n { bad~value~for~baseline }
2454       \int_set:Nn \l_tmpa_int 1
2455     }
2456     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
2457   }
2458   \dim_gsub:Nn \g_tmpa_dim \pgf@y
2459   \endpgfpicture
2460   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
2461   \int_compare:nNnT \l_@@_first_row_int = 0
2462   {

```

```

2463     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
2464     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
2465   }
2466   \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
2467 }

```

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.

```

2468 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
2469 {

```

We will compute the real width of both delimiters used.

```

2470   \dim_zero_new:N \l_@@_real_left_delim_dim
2471   \dim_zero_new:N \l_@@_real_right_delim_dim
2472   \hbox_set:Nn \l_tmpb_box
2473   {
2474     \c_math_toggle_token
2475     \left #1
2476     \vcenter
2477     {
2478       \vbox_to_ht:nn

```

Here, you should use `\box_ht_plus_dp:N` when TeXLive 2021 will be available on Overleaf.

```

2479       { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
2480       { }
2481     }
2482     \right .
2483     \c_math_toggle_token
2484   }
2485   \dim_set:Nn \l_@@_real_left_delim_dim
2486   { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
2487   \hbox_set:Nn \l_tmpb_box
2488   {
2489     \c_math_toggle_token
2490     \left .
2491     \vbox_to_ht:nn

```

Here, you should use `\box_ht_plus_dp:N` when TeXLive 2021 will be available on Overleaf.

```

2492     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
2493     { }
2494     \right #2
2495     \c_math_toggle_token
2496   }
2497   \dim_set:Nn \l_@@_real_right_delim_dim
2498   { \box_wd:N \l_tmpb_box - \nulldelimiterspace }

```

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

```

2499   \skip_horizontal:N \l_@@_left_delim_dim
2500   \skip_horizontal:N -\l_@@_real_left_delim_dim
2501   \@@_put_box_in_flow:
2502   \skip_horizontal:N \l_@@_right_delim_dim
2503   \skip_horizontal:N -\l_@@_real_right_delim_dim
2504 }

```

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).

```

2505 \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).

```
2506 {
2507   \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`).

```
2508   { \exp_args:NV \@@_array: \g_@@_preamble_tl }
2509 }
2510 {
2511   \@@_create_col_nodes:
2512   \endarray
2513 }
```

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`).

```
2514 \NewDocumentEnvironment { @@-light-syntax } { b }
2515 {
```

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`.

```
2516   \tl_if_empty:nT { #1 } { \@@_fatal:n { empty~environment } }
2517   \tl_map_inline:nn { #1 }
2518   {
2519     \str_if_eq:nnT { ##1 } { & }
2520     { \@@_fatal:n { ampersand~in~light-syntax } }
2521     \str_if_eq:nnT { ##1 } { \ }
2522     { \@@_fatal:n { double-backslash~in~light-syntax } }
2523   }
```

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`.

```
2524   \@@_light_syntax_i #1 \CodeAfter \q_stop
2525 }
```

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`.

```
2526 { }
2527 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
2528 {
2529   \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).

```
2530   \seq_gclear_new:N \g_@@_rows_seq
2531   \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
2532   \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.

```
2533   \int_compare:nNnT \l_@@_last_row_int = { -1 }
2534   { \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`).

```
2535   \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.

```

2536 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
2537 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
2538 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
2539 \@@_create_col_nodes:
2540 \endarray
2541 }

2542 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
2543 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }

2544 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
2545 {
2546 \seq_gclear_new:N \g_@@_cells_seq
2547 \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
2548 \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
2549 \l_tmpa_tl
2550 \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
2551 }

```

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).

```

2552 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
2553 {
2554 \str_if_eq:VnT \g_@@_name_env_str { #2 }
2555 { \@@_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.

```

2556 \end { #2 }
2557 }

```

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

```

2558 \cs_new:Npn \@@_create_col_nodes:
2559 {
2560 \crrc
2561 \int_compare:nNnT \l_@@_first_col_int = 0
2562 {
2563 \omit
2564 \hbox_overlap_left:n
2565 {
2566 \bool_if:NT \l_@@_code_before_bool
2567 { \pgfsys@markposition { \@@_env: - col - 0 } }
2568 \pgfpicture
2569 \pgfrememberpicturepositiononpagetrue
2570 \pgfcoordinate { \@@_env: - col - 0 } \pgfpointorigin
2571 \str_if_empty:NF \l_@@_name_str
2572 { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
2573 \endpgfpicture
2574 \skip_horizontal:N 2\col@sep
2575 \skip_horizontal:N \g_@@_width_first_col_dim
2576 }
2577 &
2578 }
2579 \omit

```

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

```

2580 \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`).

```

2581 \int_compare:nNnTF \l_@@_first_col_int = 0
2582 {
2583   \bool_if:NT \l_@@_code_before_bool
2584   {
2585     \hbox
2586     {
2587       \skip_horizontal:N -0.5\arrayrulewidth
2588       \pgfsys@markposition { \@@_env: - col - 1 }
2589       \skip_horizontal:N 0.5\arrayrulewidth
2590     }
2591   }
2592   \pgfpicture
2593   \pgfrememberpicturepositiononpagetrue
2594   \pgfcoordinate { \@@_env: - col - 1 }
2595   { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2596   \str_if_empty:NF \l_@@_name_str
2597   { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
2598   \endpgfpicture
2599 }
2600 {
2601   \bool_if:NT \l_@@_code_before_bool
2602   {
2603     \hbox
2604     {
2605       \skip_horizontal:N 0.5\arrayrulewidth
2606       \pgfsys@markposition { \@@_env: - col - 1 }
2607       \skip_horizontal:N -0.5\arrayrulewidth
2608     }
2609   }
2610   \pgfpicture
2611   \pgfrememberpicturepositiononpagetrue
2612   \pgfcoordinate { \@@_env: - col - 1 }
2613   { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
2614   \str_if_empty:NF \l_@@_name_str
2615   { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
2616   \endpgfpicture
2617 }

```

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.

```

2618 \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
2619 \bool_if:NF \l_@@_auto_columns_width_bool
2620 { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
2621 {
2622   \bool_lazy_and:nnTF
2623   \l_@@_auto_columns_width_bool
2624   { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
2625   { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
2626   { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
2627   \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
2628 }
2629 \skip_horizontal:N \g_tmpa_skip
2630 \hbox
2631 {
2632   \bool_if:NT \l_@@_code_before_bool
2633   {
2634     \hbox

```

```

2635         {
2636             \skip_horizontal:N -0.5\arrayrulewidth
2637             \pgfsys@markposition { \@@_env: - col - 2 }
2638             \skip_horizontal:N 0.5\arrayrulewidth
2639         }
2640     }
2641     \pgfpicture
2642     \pgfrememberpicturepositiononpagetrue
2643     \pgfcoordinate { \@@_env: - col - 2 }
2644     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2645     \str_if_empty:NF \l_@@_name_str
2646     { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
2647     \endpgfpicture
2648 }

```

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.

```

2649 \int_gset:Nn \g_tmpa_int 1
2650 \bool_if:NTF \g_@@_last_col_found_bool
2651 { \prg_replicate:nn { \int_max:nn { \g_@@_col_total_int - 3 } 0 } }
2652 { \prg_replicate:nn { \int_max:nn { \g_@@_col_total_int - 2 } 0 } }
2653 {
2654     &
2655     \omit
2656     \int_gincr:N \g_tmpa_int

```

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

```

2657     \skip_horizontal:N \g_tmpa_skip
2658     \bool_if:NT \l_@@_code_before_bool
2659     {
2660         \hbox
2661         {
2662             \skip_horizontal:N -0.5\arrayrulewidth
2663             \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2664             \skip_horizontal:N 0.5\arrayrulewidth
2665         }
2666     }

```

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

```

2667     \pgfpicture
2668     \pgfrememberpicturepositiononpagetrue
2669     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2670     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2671     \str_if_empty:NF \l_@@_name_str
2672     {
2673         \pgfnodealias
2674         { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
2675         { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2676     }
2677     \endpgfpicture
2678 }

2679 &
2680 \omit
2681 \int_gincr:N \g_tmpa_int
2682 \skip_horizontal:N \g_tmpa_skip
2683 \bool_lazy_all:nT
2684 {
2685     \l_@@_NiceArray_bool
2686     { \bool_not_p:n \l_@@_NiceTabular_bool }
2687     { \clist_if_empty_p:N \l_@@_vlines_clist }
2688     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
2689     { ! \l_@@_bar_at_end_of_pream_bool }
2690 }

```

```

2691     { \skip_horizontal:N -\col@sep }
2692 \bool_if:NT \l_@@_code_before_bool
2693 {
2694     \hbox
2695     {
2696         \skip_horizontal:N -0.5\arrayrulewidth

```

With an environment `{Matrix}`, you want to remove the exterior `\arraycolsep` but we don't know the number of columns (since there is no preamble) and that's why we can't put `@{}` at the end of the preamble. That's why we remove a `\arraycolsep` now.

```

2697         \bool_lazy_and:nnT \l_@@_Matrix_bool \l_@@_NiceArray_bool
2698         { \skip_horizontal:N -\arraycolsep }
2699         \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2700         \skip_horizontal:N 0.5\arrayrulewidth
2701         \bool_lazy_and:nnT \l_@@_Matrix_bool \l_@@_NiceArray_bool
2702         { \skip_horizontal:N \arraycolsep }
2703     }
2704 }
2705 \pgfpicture
2706 \pgfrememberpicturepositiononpagetrue
2707 \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2708 {
2709     \bool_lazy_and:nnTF \l_@@_Matrix_bool \l_@@_NiceArray_bool
2710     {
2711         \pgfpoint
2712         { - 0.5 \arrayrulewidth - \arraycolsep }
2713         \c_zero_dim
2714     }
2715     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2716 }
2717 \str_if_empty:NF \l_@@_name_str
2718 {
2719     \pgfnodealias
2720     { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
2721     { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2722 }
2723 \endpgfpicture

```

```

2724 \bool_if:NT \g_@@_last_col_found_bool
2725 {
2726     \hbox_overlap_right:n
2727     {
2728         \skip_horizontal:N \g_@@_width_last_col_dim
2729         \bool_if:NT \l_@@_code_before_bool
2730         {
2731             \pgfsys@markposition
2732             { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2733         }
2734         \pgfpicture
2735         \pgfrememberpicturepositiononpagetrue
2736         \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2737         \pgfpointorigin
2738         \str_if_empty:NF \l_@@_name_str
2739         {
2740             \pgfnodealias
2741             { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
2742             { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
2743         }
2744         \endpgfpicture
2745     }
2746 }
2747 \cr
2748 }

```

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

```
2749 \tl_const:Nn \c_@@_preamble_first_col_tl
2750 {
2751   >
2752   {
```

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`).

```
2753     \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n
2754     \bool_gset_true:N \g_@@_after_col_zero_bool
2755     \@@_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.

```
2756     \hbox_set:Nw \l_@@_cell_box
2757     \@@_math_toggle_token:
2758     \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”.

```
2759     \bool_lazy_and:nnT
2760     { \int_compare_p:nNn \c@iRow > 0 }
2761     {
2762       \bool_lazy_or_p:nn
2763       { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2764       { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2765     }
2766     {
2767       \l_@@_code_for_first_col_tl
2768       \xglobal \colorlet { nicematrix-first-col } { . }
2769     }
2770   }
```

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`.

```
2771   l
2772   <
2773   {
2774     \@@_math_toggle_token:
2775     \hbox_set_end:
2776     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2777     \@@_adjust_size_box:
2778     \@@_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.

```
2779     \dim_gset:Nn \g_@@_width_first_col_dim
2780     { \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.

```
2781     \hbox_overlap_left:n
2782     {
2783       \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2784       \@@_node_for_cell:
2785       { \box_use_drop:N \l_@@_cell_box }
2786       \skip_horizontal:N \l_@@_left_delim_dim
2787       \skip_horizontal:N \l_@@_left_margin_dim
2788       \skip_horizontal:N \l_@@_extra_left_margin_dim
2789     }
2790     \bool_gset_false:N \g_@@_empty_cell_bool
2791     \skip_horizontal:N -2\col@sep
2792   }
2793 }
```



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

```

2794 \tl_const:Nn \c_@@_preamble_last_col_tl
2795 {
2796   >
2797   {

```

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`).

```

2798   \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.

```

2799   \bool_gset_true:N \g_@@_last_col_found_bool
2800   \int_gincr:N \c@jCol
2801   \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.

```

2802   \hbox_set:Nw \l_@@_cell_box
2803   \@@_math_toggle_token:
2804   \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”.

```

2805   \int_compare:nNnT \c@iRow > 0
2806   {
2807     \bool_lazy_or:nnT
2808     { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2809     { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2810     {
2811       \l_@@_code_for_last_col_tl
2812       \xglobal \colorlet { nicematrix-last-col } { . }
2813     }
2814   }
2815 }
2816 1
2817 <
2818 {
2819   \@@_math_toggle_token:
2820   \hbox_set_end:
2821   \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2822   \@@_adjust_size_box:
2823   \@@_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.

```

2824   \dim_gset:Nn \g_@@_width_last_col_dim
2825   { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
2826   \skip_horizontal:N -2\col@sep

```

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

```

2827   \hbox_overlap_right:n
2828   {
2829     \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2830     {
2831       \skip_horizontal:N \l_@@_right_delim_dim
2832       \skip_horizontal:N \l_@@_right_margin_dim
2833       \skip_horizontal:N \l_@@_extra_right_margin_dim
2834       \@@_node_for_cell:
2835     }
2836   }
2837   \bool_gset_false:N \g_@@_empty_cell_bool
2838 }
2839 }

```

The environment `{NiceArray}` is constructed upon the environment `{NiceArrayWithDelims}` but, in fact, there is a flag `\l_@@_NiceArray_bool`. In `{NiceArrayWithDelims}`, some special code will be executed if this flag is raised.

```

2840 \NewDocumentEnvironment { NiceArray } { }
2841 {
2842   \bool_set_true:N \l_@@_NiceArray_bool
2843   \str_if_empty:NT \g_@@_name_env_str
2844     { \str_gset:Nn \g_@@_name_env_str { NiceArray } }

```

We put `.` and `.` for the delimiters but, in fact, that doesn't matter because these arguments won't be used in `{NiceArrayWithDelims}` (because the flag `\l_@@_NiceArray_bool` is raised).

```

2845   \NiceArrayWithDelims . .
2846 }
2847 { \endNiceArrayWithDelims }

```

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

```

2848 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
2849 {
2850   \NewDocumentEnvironment { #1 NiceArray } { }
2851   {
2852     \str_if_empty:NT \g_@@_name_env_str
2853       { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
2854     \@@_test_if_math_mode:
2855     \NiceArrayWithDelims #2 #3
2856   }
2857   { \endNiceArrayWithDelims }
2858 }
2859 \@@_def_env:nnn p ( )
2860 \@@_def_env:nnn b [ ]
2861 \@@_def_env:nnn B \{ \}
2862 \@@_def_env:nnn v | |
2863 \@@_def_env:nnn V \| \|

```

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

```

2864 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
2865 {
2866   \bool_set_true:N \l_@@_Matrix_bool
2867   \use:c { #1 NiceArray }
2868   {
2869     *
2870     {
2871       \int_compare:nNnTF \l_@@_last_col_int < 0
2872         \c@MaxMatrixCols
2873         { \@@_pred:n \l_@@_last_col_int }
2874     }
2875     { > \@@_cell_begin:w #2 < \@@_cell_end: }
2876   }
2877 }
2878 \clist_map_inline:nn { { } , p , b , B , v , V }
2879 {
2880   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
2881   {
2882     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
2883     \tl_set:Nn \l_@@_type_of_col_tl c
2884     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
2885     \exp_args:Nne \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
2886   }
2887   { \use:c { end #1 NiceArray } }

```

```
2888 }
```

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

```
2889 \cs_new_protected:Npn \l_@@_NotEmpty:
2890 { \bool_gset_true:N \g_@@_not_empty_cell_bool }
```

## {NiceTabular}, {NiceTabularX} and {NiceTabular\*}

```
2891 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
2892 {
```

The dimension \l\_@@\_width\_dim will be used if at least one column of type X is used.

```
2893 \dim_zero_new:N \l_@@_width_dim
2894 \dim_set_eq:NN \l_@@_width_dim \linewidth
2895 \str_gset:Nn \g_@@_name_env_str { NiceTabular }
2896 \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
2897 \bool_set_true:N \l_@@_NiceTabular_bool
2898 \NiceArray { #2 }
2899 }
2900 { \endNiceArray }
```

```
2901 \NewDocumentEnvironment { NiceTabularX } { m 0 { } m ! 0 { } }
2902 {
2903 \str_gset:Nn \g_@@_name_env_str { NiceTabularX }
2904 \dim_zero_new:N \l_@@_width_dim
2905 \dim_set:NN \l_@@_width_dim { #1 }
2906 \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2907 \bool_set_true:N \l_@@_NiceTabular_bool
2908 \NiceArray { #3 }
2909 }
2910 { \endNiceArray }
```

```
2911 \NewDocumentEnvironment { NiceTabular* } { m 0 { } m ! 0 { } }
2912 {
2913 \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
2914 \dim_set:NN \l_@@_tabular_width_dim { #1 }
2915 \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2916 \bool_set_true:N \l_@@_NiceTabular_bool
2917 \NiceArray { #3 }
2918 }
2919 { \endNiceArray }
```

## After the construction of the array

```
2920 \cs_new_protected:Npn \l_@@_after_array:
2921 {
2922 \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.

```
2923 \bool_if:NT \g_@@_last_col_found_bool
2924 { \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 also fix the real value of \l\_@@\_last\_col\_int.

```
2925 \bool_if:NT \l_@@_last_col_without_value_bool
2926 { \int_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int }
```

It's also time to give to `\l_@@_last_row_int` its real value.

```

2927 \bool_if:NT \l_@@_last_row_without_value_bool
2928 { \int_set_eq:NN \l_@@_last_row_int \g_@@_row_total_int }

2929 \tl_gput_right:Nx \g_@@_aux_tl
2930 {
2931   \seq_gset_from_clist:Nn \exp_not:N \c_@@_size_seq
2932   {
2933     \int_use:N \l_@@_first_row_int ,
2934     \int_use:N \c@iRow ,
2935     \int_use:N \g_@@_row_total_int ,
2936     \int_use:N \l_@@_first_col_int ,
2937     \int_use:N \c@jCol ,
2938     \int_use:N \g_@@_col_total_int
2939   }
2940 }

```

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`).

```

2941 \seq_if_empty:NF \g_@@_pos_of_blocks_seq
2942 {
2943   \tl_gput_right:Nx \g_@@_aux_tl
2944   {
2945     \seq_gset_from_clist:Nn \exp_not:N \g_@@_pos_of_blocks_seq
2946     { \seq_use:Nnnn \g_@@_pos_of_blocks_seq , , , }
2947   }
2948 }
2949 \seq_if_empty:NF \g_@@_multicolumn_cells_seq
2950 {
2951   \tl_gput_right:Nx \g_@@_aux_tl
2952   {
2953     \seq_gset_from_clist:Nn \exp_not:N \g_@@_multicolumn_cells_seq
2954     { \seq_use:Nnnn \g_@@_multicolumn_cells_seq , , , }
2955     \seq_gset_from_clist:Nn \exp_not:N \g_@@_multicolumn_sizes_seq
2956     { \seq_use:Nnnn \g_@@_multicolumn_sizes_seq , , , }
2957   }
2958 }

```

Now, you create the diagonal nodes by using the `row` nodes and the `col` nodes.

```

2959 \@@_create_diag_nodes:

```

By default, the diagonal lines will be parallelized<sup>63</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.

```

2960 \bool_if:NT \l_@@_parallelize_diags_bool
2961 {
2962   \int_gzero_new:N \g_@@_ddots_int
2963   \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.

```

2964   \dim_gzero_new:N \g_@@_delta_x_one_dim
2965   \dim_gzero_new:N \g_@@_delta_y_one_dim
2966   \dim_gzero_new:N \g_@@_delta_x_two_dim
2967   \dim_gzero_new:N \g_@@_delta_y_two_dim
2968 }
2969 \int_zero_new:N \l_@@_initial_i_int
2970 \int_zero_new:N \l_@@_initial_j_int
2971 \int_zero_new:N \l_@@_final_i_int
2972 \int_zero_new:N \l_@@_final_j_int

```

---

<sup>63</sup>It's possible to use the option `parallelize-diags` to disable this parallelization.

```

2973 \bool_set_false:N \l_@@_initial_open_bool
2974 \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.

```

2975 \bool_if:NT \l_@@_small_bool
2976 {
2977   \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2978   \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.

```

2979   \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2980 }

```

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

```

2981 \@@_draw_dotted_lines:

```

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.

```

2982 \@@_compute_corners:

```

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

```

2983 \@@_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`.

```

2984 % \bool_lazy_all:nT
2985 % {
2986 %   { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
2987 %   { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
2988 %   { \seq_if_empty_p:N \l_@@_corners_cells_seq }
2989 % }
2990 % {
2991 %   \cs_set_eq:NN \@@_vline_i:nn \@@_vline_i_complete:nn
2992 %   \cs_set_eq:NN \@@_hline_i:nn \@@_hline_i_complete:nn
2993 % }
2994 \tl_if_empty:NF \l_@@_hlines_clist \@@_draw_hlines:
2995 \tl_if_empty:NF \l_@@_vlines_clist \@@_draw_vlines:
2996 \cs_set_eq:NN \SubMatrix \@@_SubMatrix

```

Now, the internal code-after and then, the `\CodeAfter`.

```

2997 \bool_if:NT \c_@@_tikz_loaded_bool
2998 {
2999   \tikzset
3000   {
3001     every~picture / .style =
3002     {
3003       overlay ,
3004       remember~picture ,
3005       name~prefix = \@@_env: -
3006     }
3007   }
3008 }
3009 \cs_set_eq:NN \line \@@_line

```

```

3010 \g_@@_internal_code_after_tl
3011 \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.

```

3012 \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).

```

3013 \seq_gclear:N \g_@@_submatrix_names_seq

```

We compose the `code-after` in math mode in order to nullify the spaces put by the user between instructions in the `code-after`.

```

3014 % \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token

```

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:`.

```

3015 \exp_last_unbraced:NV \@@_CodeAfter_keys: \g_nicematrix_code_after_tl
3016 \scan_stop:
3017 % \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
3018 \tl_gclear:N \g_nicematrix_code_after_tl
3019 \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.

```

3020 \tl_if_empty:NF \g_nicematrix_code_before_tl
3021 {

```

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`.

```

3022 \cs_set_protected:Npn \rectanglecolor { }
3023 \cs_set_protected:Npn \columncolor { }
3024 \tl_gput_right:Nx \g_@@_aux_tl
3025 {
3026   \tl_gset:Nn \exp_not:N \g_@@_code_before_tl
3027   { \exp_not:V \g_nicematrix_code_before_tl }
3028 }
3029 \bool_set_true:N \l_@@_code_before_bool
3030 }
3031 % \bool_if:NT \l_@@_code_before_bool \@@_write_aux_for_cell_nodes:

3032 \str_gclear:N \g_@@_name_env_str
3033 \@@_restore_iRow_jCol:

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>64</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`.

```

3034 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
3035 }

```

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`). Idem for the `\CodeBefore`.

```

3036 \NewDocumentCommand \@@_CodeAfter_keys: { 0 { } }
3037 { \keys_set:nn { NiceMatrix / CodeAfter } { #1 } }

```

---

<sup>64</sup>e.g. `\color[rgb]{0.5,0.5,0}`

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).

```

3038 \cs_new_protected:Npn \@@_adjust_pos_of_blocks_seq:
3039 {
3040   \seq_gset_map_x:NNn \g_@@_pos_of_blocks_seq \g_@@_pos_of_blocks_seq
3041   { \@@_adjust_pos_of_blocks_seq_i:nnnn #1 }
3042 }

```

The following command must *not* be protected.

```

3043 \cs_new:Npn \@@_adjust_pos_of_blocks_seq_i:nnnn #1 #2 #3 #4
3044 {
3045   { #1 }
3046   { #2 }
3047   {
3048     \int_compare:nNnTF { #3 } > { 99 }
3049     { \int_use:N \c@iRow }
3050     { #3 }
3051   }
3052   {
3053     \int_compare:nNnTF { #4 } > { 99 }
3054     { \int_use:N \c@jCol }
3055     { #4 }
3056   }
3057 }

```

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).

```

3058 \AtBeginDocument
3059 {
3060   \cs_new_protected:Npx \@@_draw_dotted_lines:
3061   {
3062     \c_@@_pgfortikzpicture_tl
3063     \@@_draw_dotted_lines_i:
3064     \c_@@_endpgfortikzpicture_tl
3065   }
3066 }

```

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

```

3067 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
3068 {
3069   \pgfrememberpicturerepositiononpagetrue
3070   \pgf@relevantforpicturesizefalse
3071   \g_@@_HVdotsfor_lines_tl
3072   \g_@@_Vdots_lines_tl
3073   \g_@@_Ddots_lines_tl
3074   \g_@@_Iddots_lines_tl
3075   \g_@@_Cdots_lines_tl
3076   \g_@@_Ldots_lines_tl
3077 }

3078 \cs_new_protected:Npn \@@_restore_iRow_jCol:
3079 {
3080   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
3081   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
3082 }

```

We define a new PGF shape for the diag nodes because we want to provide a anchor called .5 for those nodes.

```

3083 \pgfdeclareshape { @@_diag_node }
3084 {
3085     \savedanchor { \five }
3086     {
3087         \dim_gset_eq:NN \pgf@x \l_tmpa_dim
3088         \dim_gset_eq:NN \pgf@y \l_tmpb_dim
3089     }
3090     \anchor { 5 } { \five }
3091     \anchor { center } { \pgfpointorigin }
3092 }

3093 \cs_new_protected:Npn @@_write_aux_for_cell_nodes:
3094 {
3095     \pgfpicture
3096     \pgfrememberpicturepositiononpagetrue
3097     \pgf@relevantforpicturesizefalse
3098     \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
3099     {
3100         \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
3101         {
3102             \cs_if_exist:cT { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
3103             {
3104                 \pgfscope
3105                 \pgftransformshift
3106                 { \pgfpointanchor { \@@_env: - ##1 - #####1 } { north-west } }
3107                 \pgfnode
3108                 { rectangle }
3109                 { center }
3110                 {
3111                     \hbox
3112                     { \pgfsys@markposition { \@@_env: - ##1 - #####1 - NW } }
3113                 }
3114                 { }
3115                 { }
3116                 \endpgfscope
3117                 \pgfscope
3118                 \pgftransformshift
3119                 { \pgfpointanchor { \@@_env: - ##1 - #####1 } { south-east } }
3120                 \pgfnode
3121                 { rectangle }
3122                 { center }
3123                 {
3124                     \hbox
3125                     { \pgfsys@markposition { \@@_env: - ##1 - #####1 - SE } }
3126                 }
3127                 { }
3128                 { }
3129                 \endpgfscope
3130             }
3131         }
3132     }
3133     \endpgfpicture
3134     \@@_create_extra_nodes:
3135 }
3136 % \end{macrocode}
3137 %
3138 % \bigskip
3139 % The following command creates the diagonal nodes (in fact, if the matrix is
3140 % not a square matrix, not all the nodes are on the diagonal).
3141 % \begin{macrocode}
3142 \cs_new_protected:Npn @@_create_diag_nodes:

```



```

3143 {
3144   \pgfpicture
3145   \pgfrememberpicturepositiononpagetrue
3146   \int_step_inline:nn { \int_max:nn \c@iRow \c@jCol }
3147   {
3148     \@@_qpoint:n { col - \int_min:nn { ##1 } { \c@jCol + 1 } }
3149     \dim_set_eq:NN \l_tmpa_dim \pgf@x
3150     \@@_qpoint:n { row - \int_min:nn { ##1 } { \c@iRow + 1 } }
3151     \dim_set_eq:NN \l_tmpb_dim \pgf@y
3152     \@@_qpoint:n { col - \int_min:nn { ##1 + 1 } { \c@jCol + 1 } }
3153     \dim_set_eq:NN \l_tmpc_dim \pgf@x
3154     \@@_qpoint:n { row - \int_min:nn { ##1 + 1 } { \c@iRow + 1 } }
3155     \dim_set_eq:NN \l_tmpd_dim \pgf@y
3156     \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 `@ä_diag_node`) that we will construct.

```

3157     \dim_set:Nn \l_tmpa_dim { ( \l_tmpc_dim - \l_tmpa_dim ) / 2 }
3158     \dim_set:Nn \l_tmpb_dim { ( \l_tmpd_dim - \l_tmpb_dim ) / 2 }
3159     \pgfnode { @@_diag_node } { center } { } { \@@_env: - ##1 } { }
3160     \str_if_empty:NF \l_@@_name_str
3161     { \pgfnodealias { \l_@@_name_str - ##1 } { \@@_env: - ##1 } }
3162   }

```

Now, the last node. Of course, that is only a coordinate because there is not `.5` anchor for that node.

```

3163   \int_set:Nn \l_tmpa_int { \int_max:nn \c@iRow \c@jCol + 1 }
3164   \@@_qpoint:n { row - \int_min:nn { \l_tmpa_int } { \c@iRow + 1 } }
3165   \dim_set_eq:NN \l_tmpa_dim \pgf@y
3166   \@@_qpoint:n { col - \int_min:nn { \l_tmpa_int } { \c@jCol + 1 } }
3167   \pgfcoordinate
3168   { \@@_env: - \int_use:N \l_tmpa_int } { \pgfpoint \pgf@x \l_tmpa_dim }
3169   \pgfnodealias
3170   { \@@_env: - last }
3171   { \@@_env: - \int_eval:n { \int_max:nn \c@iRow \c@jCol + 1 } }
3172   \str_if_empty:NF \l_@@_name_str
3173   {
3174     \pgfnodealias
3175     { \l_@@_name_str - \int_use:N \l_tmpa_int }
3176     { \@@_env: - \int_use:N \l_tmpa_int }
3177     \pgfnodealias
3178     { \l_@@_name_str - last }
3179     { \@@_env: - last }
3180   }
3181   \endpgfpicture
3182 }

```

## We draw the dotted lines

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

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

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

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

- the fourth argument is the  $y$ -value of the orientation vector of the line.

This command computes:

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

```
3183 \cs_new_protected:Npn \l_@@_find_extremities_of_line:nnnn #1 #2 #3 #4
3184 {
```

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

```
3185 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
3186 \int_set:Nn \l_@@_initial_i_int { #1 }
3187 \int_set:Nn \l_@@_initial_j_int { #2 }
3188 \int_set:Nn \l_@@_final_i_int { #1 }
3189 \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.

```
3190 \bool_set_false:N \l_@@_stop_loop_bool
3191 \bool_do_until:Nn \l_@@_stop_loop_bool
3192 {
3193   \int_add:Nn \l_@@_final_i_int { #3 }
3194   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
3195 \bool_set_false:N \l_@@_final_open_bool
3196 \int_compare:nNnTF \l_@@_final_i_int > \l_@@_row_max_int
3197 {
3198   \int_compare:nNnTF { #3 } = 1
3199   { \bool_set_true:N \l_@@_final_open_bool }
3200   {
3201     \int_compare:nNnTF \l_@@_final_j_int > \l_@@_col_max_int
3202     { \bool_set_true:N \l_@@_final_open_bool }
3203   }
3204 }
3205 {
3206   \int_compare:nNnTF \l_@@_final_j_int < \l_@@_col_min_int
3207   {
3208     \int_compare:nNnTF { #4 } = { -1 }
3209     { \bool_set_true:N \l_@@_final_open_bool }
3210   }
3211   {
3212     \int_compare:nNnTF \l_@@_final_j_int > \l_@@_col_max_int
3213     {
3214       \int_compare:nNnTF { #4 } = 1
3215       { \bool_set_true:N \l_@@_final_open_bool }
3216     }
3217   }
3218 }
3219 \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.

```
3220 {
```

We do a step backwards.

```

3221     \int_sub:Nn \l_@@_final_i_int { #3 }
3222     \int_sub:Nn \l_@@_final_j_int { #4 }
3223     \bool_set_true:N \l_@@_stop_loop_bool
3224 }

```

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`.

```

3225 {
3226   \cs_if_exist:cTF
3227   {
3228     @@ _ dotted _
3229     \int_use:N \l_@@_final_i_int -
3230     \int_use:N \l_@@_final_j_int
3231   }
3232   {
3233     \int_sub:Nn \l_@@_final_i_int { #3 }
3234     \int_sub:Nn \l_@@_final_j_int { #4 }
3235     \bool_set_true:N \l_@@_final_open_bool
3236     \bool_set_true:N \l_@@_stop_loop_bool
3237   }
3238   {
3239     \cs_if_exist:cTF
3240     {
3241       pgf @ sh @ ns @ \@@_env:
3242       - \int_use:N \l_@@_final_i_int
3243       - \int_use:N \l_@@_final_j_int
3244     }
3245     { \bool_set_true:N \l_@@_stop_loop_bool }
3246   }
3247 }

```

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.

```

3246 {
3247   \cs_set:cpn
3248   {
3249     @@ _ dotted _
3250     \int_use:N \l_@@_final_i_int -
3251     \int_use:N \l_@@_final_j_int
3252   }
3253   { }
3254 }
3255 }
3256 }
3257 }

```

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

```

3258 \bool_set_false:N \l_@@_stop_loop_bool
3259 \bool_do_until:Nn \l_@@_stop_loop_bool
3260 {
3261   \int_sub:Nn \l_@@_initial_i_int { #3 }
3262   \int_sub:Nn \l_@@_initial_j_int { #4 }
3263   \bool_set_false:N \l_@@_initial_open_bool
3264   \int_compare:nNnTF \l_@@_initial_i_int < \l_@@_row_min_int
3265   {
3266     \int_compare:nNnTF { #3 } = 1
3267     { \bool_set_true:N \l_@@_initial_open_bool }
3268     {
3269       \int_compare:nNnT \l_@@_initial_j_int = { \l_@@_col_min_int -1 }
3270     }
3271   }
3272 }

```

```

3270         { \bool_set_true:N \l_@@_initial_open_bool }
3271     }
3272 }
3273 {
3274     \int_compare:nNnTF \l_@@_initial_j_int < \l_@@_col_min_int
3275     {
3276         \int_compare:nNnT { #4 } = 1
3277         { \bool_set_true:N \l_@@_initial_open_bool }
3278     }
3279     {
3280         \int_compare:nNnT \l_@@_initial_j_int > \l_@@_col_max_int
3281         {
3282             \int_compare:nNnT { #4 } = { -1 }
3283             { \bool_set_true:N \l_@@_initial_open_bool }
3284         }
3285     }
3286 }
3287 \bool_if:NTF \l_@@_initial_open_bool
3288 {
3289     \int_add:Nn \l_@@_initial_i_int { #3 }
3290     \int_add:Nn \l_@@_initial_j_int { #4 }
3291     \bool_set_true:N \l_@@_stop_loop_bool
3292 }
3293 {
3294     \cs_if_exist:cTF
3295     {
3296         @@ _ dotted _
3297         \int_use:N \l_@@_initial_i_int -
3298         \int_use:N \l_@@_initial_j_int
3299     }
3300     {
3301         \int_add:Nn \l_@@_initial_i_int { #3 }
3302         \int_add:Nn \l_@@_initial_j_int { #4 }
3303         \bool_set_true:N \l_@@_initial_open_bool
3304         \bool_set_true:N \l_@@_stop_loop_bool
3305     }
3306     {
3307         \cs_if_exist:cTF
3308         {
3309             pgf @ sh @ ns @ \@@_env:
3310             - \int_use:N \l_@@_initial_i_int
3311             - \int_use:N \l_@@_initial_j_int
3312         }
3313         { \bool_set_true:N \l_@@_stop_loop_bool }
3314         {
3315             \cs_set:cpn
3316             {
3317                 @@ _ dotted _
3318                 \int_use:N \l_@@_initial_i_int -
3319                 \int_use:N \l_@@_initial_j_int
3320             }
3321             { }
3322         }
3323     }
3324 }
3325 }

```

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.

```

3326 \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
3327 {
3328     { \int_use:N \l_@@_initial_i_int }
3329     { \int_use:N \l_@@_initial_j_int }
3330     { \int_use:N \l_@@_final_i_int }

```

```

3331         { \int_use:N \l_@@_final_j_int }
3332     }
3333 }

```

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 rows and columns).

```

3334 \cs_new_protected:Npn \@@_adjust_to_submatrix:nn #1 #2
3335 {
3336     \int_set:Nn \l_@@_row_min_int 1
3337     \int_set:Nn \l_@@_col_min_int 1
3338     \int_set_eq:NN \l_@@_row_max_int \c@iRow
3339     \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`.

```

3340     \seq_map_inline:Nn \g_@@_submatrix_seq
3341     { \@@_adjust_to_submatrix:nnnnnn { #1 } { #2 } ##1 }
3342 }

```

#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.

```

3343 \cs_set_protected:Npn \@@_adjust_to_submatrix:nnnnnn #1 #2 #3 #4 #5 #6
3344 {
3345     \bool_if:nT
3346     {
3347         \int_compare_p:n { #3 <= #1 }
3348         && \int_compare_p:n { #1 <= #5 }
3349         && \int_compare_p:n { #4 <= #2 }
3350         && \int_compare_p:n { #2 <= #6 }
3351     }
3352     {
3353         \int_set:Nn \l_@@_row_min_int { \int_max:nn \l_@@_row_min_int { #3 } }
3354         \int_set:Nn \l_@@_col_min_int { \int_max:nn \l_@@_col_min_int { #4 } }
3355         \int_set:Nn \l_@@_row_max_int { \int_min:nn \l_@@_row_max_int { #5 } }
3356         \int_set:Nn \l_@@_col_max_int { \int_min:nn \l_@@_col_max_int { #6 } }
3357     }
3358 }

```

```

3359 \cs_new_protected:Npn \@@_set_initial_coords:
3360 {
3361     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3362     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3363 }
3364 \cs_new_protected:Npn \@@_set_final_coords:
3365 {
3366     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3367     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3368 }
3369 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
3370 {
3371     \pgfpointanchor
3372     {
3373         \@@_env:
3374         - \int_use:N \l_@@_initial_i_int
3375         - \int_use:N \l_@@_initial_j_int
3376     }
3377     { #1 }
3378     \@@_set_initial_coords:
3379 }
3380 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1

```

```

3381 {
3382   \pgfpointanchor
3383   {
3384     \@@_env:
3385     - \int_use:N \l_@@_final_i_int
3386     - \int_use:N \l_@@_final_j_int
3387   }
3388   { #1 }
3389   \@@_set_final_coords:
3390 }
3391 \cs_new_protected:Npn \@@_open_x_initial_dim:
3392 {
3393   \dim_set_eq:NN \l_@@_x_initial_dim \c_max_dim
3394   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
3395   {
3396     \cs_if_exist:cT
3397     { \pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_initial_j_int }
3398     {
3399       \pgfpointanchor
3400       { \@@_env: - ##1 - \int_use:N \l_@@_initial_j_int }
3401       { west }
3402       \dim_set:Nn \l_@@_x_initial_dim
3403       { \dim_min:nn \l_@@_x_initial_dim \pgf@x }
3404     }
3405   }

```

If, in fact, all the cells of the columns are empty (no PGF/Tikz nodes in those cells).

```

3406   \dim_compare:nNnT \l_@@_x_initial_dim = \c_max_dim
3407   {
3408     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
3409     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3410     \dim_add:Nn \l_@@_x_initial_dim \col@sep
3411   }
3412 }
3413 \cs_new_protected:Npn \@@_open_x_final_dim:
3414 {
3415   \dim_set:Nn \l_@@_x_final_dim { - \c_max_dim }
3416   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
3417   {
3418     \cs_if_exist:cT
3419     { \pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_final_j_int }
3420     {
3421       \pgfpointanchor
3422       { \@@_env: - ##1 - \int_use:N \l_@@_final_j_int }
3423       { east }
3424       \dim_set:Nn \l_@@_x_final_dim
3425       { \dim_max:nn \l_@@_x_final_dim \pgf@x }
3426     }
3427   }

```

If, in fact, all the cells of the columns are empty (no PGF/Tikz nodes in those cells).

```

3428   \dim_compare:nNnT \l_@@_x_final_dim = { - \c_max_dim }
3429   {
3430     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
3431     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3432     \dim_sub:Nn \l_@@_x_final_dim \col@sep
3433   }
3434 }

```

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.

```

3435 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3

```

```

3436 {
3437   \@@_adjust_to_submatrix:nn { #1 } { #2 }
3438   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
3439   {
3440     \@@_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.

```

3441     \group_begin:
3442     \int_compare:nNnTF { #1 } = 0
3443     { \color { nicematrix-first-row } }
3444     {

```

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.

```

3445         \int_compare:nNnT { #1 } = \l_@@_last_row_int
3446         { \color { nicematrix-last-row } }
3447     }
3448     \keys_set:nn { NiceMatrix / xdots } { #3 }
3449     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3450     \@@_actually_draw_Ldots:
3451   \group_end:
3452 }
3453 }

```

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`.

```

3454 \cs_new_protected:Npn \@@_actually_draw_Ldots:
3455 {
3456   \bool_if:NTF \l_@@_initial_open_bool
3457   {
3458     \@@_open_x_initial_dim:
3459     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
3460     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3461   }
3462   { \@@_set_initial_coords_from_anchor:n { base-east } }
3463   \bool_if:NTF \l_@@_final_open_bool
3464   {
3465     \@@_open_x_final_dim:
3466     \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
3467     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3468   }
3469   { \@@_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 (?).

```

3470   \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
3471   \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
3472   \@@_draw_line:
3473 }

```

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.

```

3474 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
3475 {
3476   \@@_adjust_to_submatrix:nn { #1 } { #2 }
3477   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
3478   {
3479     \@@_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.

```

3480     \group_begin:
3481     \int_compare:nNnTF { #1 } = 0
3482     { \color { nicematrix-first-row } }
3483     {

```

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.

```

3484         \int_compare:nNnT { #1 } = \l_@@_last_row_int
3485         { \color { nicematrix-last-row } }
3486     }
3487     \keys_set:nn { NiceMatrix / xdots } { #3 }
3488     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3489     \@@_actually_draw_Cdots:
3490     \group_end:
3491   }
3492 }

```

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`.

```

3493 \cs_new_protected:Npn \@@_actually_draw_Cdots:
3494 {
3495   \bool_if:NTF \l_@@_initial_open_bool
3496   { \@@_open_x_initial_dim: }
3497   { \@@_set_initial_coords_from_anchor:n { mid-east } }
3498   \bool_if:NTF \l_@@_final_open_bool
3499   { \@@_open_x_final_dim: }
3500   { \@@_set_final_coords_from_anchor:n { mid-west } }
3501   \bool_lazy_and:nnTF
3502   \l_@@_initial_open_bool
3503   \l_@@_final_open_bool
3504   {
3505     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
3506     \dim_set_eq:NN \l_tmpa_dim \pgf@y
3507     \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
3508     \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
3509     \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
3510   }
3511   {
3512     \bool_if:NT \l_@@_initial_open_bool
3513     { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
3514     \bool_if:NT \l_@@_final_open_bool
3515     { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
3516   }

```



```

3517 \@@_draw_line:
3518 }
3519 \cs_new_protected:Npn \@@_open_y_initial_dim:
3520 {
3521 \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
3522 \dim_set:Nn \l_@@_y_initial_dim
3523 { \pgf@y + ( \box_ht:N \strutbox + \extrarowheight ) * \arraystretch }
3524 \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
3525 {
3526 \cs_if_exist:cT
3527 { pgf @ sh @ ns @ \@@_env: - \int_use:N \l_@@_initial_i_int - ##1 }
3528 {
3529 \pgfpointanchor
3530 { \@@_env: - \int_use:N \l_@@_initial_i_int - ##1 }
3531 { north }
3532 \dim_set:Nn \l_@@_y_initial_dim
3533 { \dim_max:nn \l_@@_y_initial_dim \pgf@y }
3534 }
3535 }
3536 }
3537 \cs_new_protected:Npn \@@_open_y_final_dim:
3538 {
3539 \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
3540 \dim_set:Nn \l_@@_y_final_dim
3541 { \pgf@y - ( \box_dp:N \strutbox ) * \arraystretch }
3542 \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
3543 {
3544 \cs_if_exist:cT
3545 { pgf @ sh @ ns @ \@@_env: - \int_use:N \l_@@_final_i_int - ##1 }
3546 {
3547 \pgfpointanchor
3548 { \@@_env: - \int_use:N \l_@@_final_i_int - ##1 }
3549 { south }
3550 \dim_set:Nn \l_@@_y_final_dim
3551 { \dim_min:nn \l_@@_y_final_dim \pgf@y }
3552 }
3553 }
3554 }

```

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.

```

3555 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
3556 {
3557 \@@_adjust_to_submatrix:nn { #1 } { #2 }
3558 \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
3559 {
3560 \@@_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.

```

3561 \group_begin:
3562 \int_compare:nNnTF { #2 } = 0
3563 { \color { nicematrix-first-col } }
3564 {
3565 \int_compare:nNnT { #2 } = \l_@@_last_col_int
3566 { \color { nicematrix-last-col } }
3567 }
3568 \keys_set:nn { NiceMatrix / xdots } { #3 }
3569 \tl_if_empty:VF \l_@@_xdots_color_tl
3570 { \color { \l_@@_xdots_color_tl } }
3571 \@@_actually_draw_Vdots:
3572 \group_end:
3573 }
3574 }

```

The command `\l_@@_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`.

```
3575 \cs_new_protected:Npn \l_@@_actually_draw_Vdots:
3576 {
```

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

```
3577 \bool_set_false:N \l_tmpa_bool
```

First the case when the line is closed on both ends.

```
3578 \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
3579 {
3580   \@@_set_initial_coords_from_anchor:n { south-west }
3581   \@@_set_final_coords_from_anchor:n { north-west }
3582   \bool_set:Nn \l_tmpa_bool
3583   { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
3584 }
```

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

```
3585 \bool_if:NTF \l_@@_initial_open_bool
3586   \@@_open_y_initial_dim:
3587   { \@@_set_initial_coords_from_anchor:n { south } }
3588 \bool_if:NTF \l_@@_final_open_bool
3589   \@@_open_y_final_dim:
3590   { \@@_set_final_coords_from_anchor:n { north } }
3591 \bool_if:NTF \l_@@_initial_open_bool
3592 {
3593   \bool_if:NTF \l_@@_final_open_bool
3594   {
3595     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
3596     \dim_set_eq:NN \l_tmpa_dim \pgf@x
3597     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
3598     \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
3599     \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.

```
3600 \int_compare:nNnT \l_@@_last_col_int > { -2 }
3601 {
3602   \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
3603   {
3604     \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
3605     \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
3606     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
3607     \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
3608   }
3609 }
3610 }
3611 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
3612 }
3613 {
3614   \bool_if:NTF \l_@@_final_open_bool
3615   { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
3616 }
```

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.

```

3617         \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
3618         {
3619             \dim_set:Nn \l_@@_x_initial_dim
3620             {
3621                 \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
3622                 \l_@@_x_initial_dim \l_@@_x_final_dim
3623             }
3624             \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
3625         }
3626     }
3627 }
3628 \@@_draw_line:
3629 }
```

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.

```

3630 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
3631 {
3632     \@@_adjust_to_submatrix:nn { #1 } { #2 }
3633     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
3634     {
3635         \@@_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.

```

3636         \group_begin:
3637         \keys_set:nn { NiceMatrix / xdots } { #3 }
3638         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3639         \@@_actually_draw_Ddots:
3640         \group_end:
3641     }
3642 }
```

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.`

```

3643 \cs_new_protected:Npn \@@_actually_draw_Ddots:
3644 {
3645     \bool_if:NTF \l_@@_initial_open_bool
3646     {
3647         \@@_open_y_initial_dim:
3648         % \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
3649         % \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3650         \@@_open_x_initial_dim:
3651     }
3652     { \@@_set_initial_coords_from_anchor:n { south-east } }
3653     \bool_if:NTF \l_@@_final_open_bool
```

```

3654 {
3655   % \@@_open_y_final_dim:
3656   % \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
3657   \@@_open_x_final_dim:
3658   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3659 }
3660 { \@@_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.

```

3661   \bool_if:NT \l_@@_parallelize_diags_bool
3662   {
3663     \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).

```

3664     \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.

```

3665     {
3666       \dim_gset:Nn \g_@@_delta_x_one_dim
3667       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
3668       \dim_gset:Nn \g_@@_delta_y_one_dim
3669       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
3670     }

```

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`.

```

3671     {
3672       \dim_set:Nn \l_@@_y_final_dim
3673       {
3674         \l_@@_y_initial_dim +
3675         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3676         \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
3677       }
3678     }
3679   }
3680   \@@_draw_line:
3681 }

```

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.

```

3682 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
3683 {
3684   \@@_adjust_to_submatrix:nn { #1 } { #2 }
3685   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
3686   {
3687     \@@_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.

```

3688     \group_begin:
3689     \keys_set:nn { NiceMatrix / xdots } { #3 }
3690     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3691     \@@_actually_draw_Iddots:
3692   \group_end:
3693 }
3694 }

```

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.

```

3695 \cs_new_protected:Npn \@@_actually_draw_Iddots:
3696 {
3697   \bool_if:NTF \l_@@_initial_open_bool
3698   {
3699     \@@_open_y_initial_dim:
3700     \@@_open_x_initial_dim:
3701   }
3702   { \@@_set_initial_coords_from_anchor:n { south-west } }
3703   \bool_if:NTF \l_@@_final_open_bool
3704   {
3705     \@@_open_y_final_dim:
3706     \@@_open_x_final_dim:
3707   }
3708   { \@@_set_final_coords_from_anchor:n { north-east } }
3709   \bool_if:NT \l_@@_parallelize_diags_bool
3710   {
3711     \int_gincr:N \g_@@_iddots_int
3712     \int_compare:nNnTF \g_@@_iddots_int = 1
3713     {
3714       \dim_gset:Nn \g_@@_delta_x_two_dim
3715       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
3716       \dim_gset:Nn \g_@@_delta_y_two_dim
3717       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
3718     }
3719     {
3720       \dim_set:Nn \l_@@_y_final_dim
3721       {
3722         \l_@@_y_initial_dim +
3723         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3724         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
3725       }
3726     }
3727   }
3728   \@@_draw_line:
3729 }

```

## The actual instructions for drawing the dotted lines 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

```

3730 \cs_new_protected:Npn \@@_draw_line:
3731 {
3732   \pgfrememberpicturepositiononpagetrue
3733   \pgf@relevantforpicturesizefalse
3734   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
3735     \@@_draw_standard_dotted_line:
3736     \@@_draw_non_standard_dotted_line:
3737 }

```

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.

```

3738 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
3739 {
3740   \begin { scope }
3741   \exp_args:No \@@_draw_non_standard_dotted_line:n
3742     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
3743 }

```

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.

```

3744 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
3745 {
3746   \@@_draw_non_standard_dotted_line:nVV
3747     { #1 }
3748     \l_@@_xdots_up_tl
3749     \l_@@_xdots_down_tl
3750 }
3751 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:nnn #1 #2 #3
3752 {
3753   \draw
3754     [
3755       #1 ,
3756       shorten-> = \l_@@_xdots_shorten_dim ,
3757       shorten-< = \l_@@_xdots_shorten_dim ,
3758     ]
3759     ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )

```

Be careful: We can't put `\c_math_toggle_token` instead of `$` in the following lines because we are in the contents of Tikz nodes (and they will be *rescanned* if the Tikz library `babel` is loaded).

```

3760   -- node [ sloped , above ] { $ \scriptstyle #2 $ }
3761   node [ sloped , below ] { $ \scriptstyle #3 $ }
3762   ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
3763   \end { scope }
3764 }
3765 \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).

```

3766 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
3767 {
3768   \bool_lazy_and:nnF
3769     { \tl_if_empty_p:N \l_@@_xdots_up_tl }
3770     { \tl_if_empty_p:N \l_@@_xdots_down_tl }
3771   {
3772     \pgfscope
3773     \pgftransformshift
3774       {
3775         \pgfpointlineattime { 0.5 }
3776         { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
3777         { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
3778       }

```

```

3779 \pgftransformrotate
3780 {
3781     \fp_eval:n
3782     {
3783         atand
3784         (
3785             \l_@@_y_final_dim - \l_@@_y_initial_dim ,
3786             \l_@@_x_final_dim - \l_@@_x_initial_dim
3787         )
3788     }
3789 }
3790 \pgfnode
3791 { rectangle }
3792 { south }
3793 {
3794     \c_math_toggle_token
3795     \scriptstyle \l_@@_xdots_up_tl
3796     \c_math_toggle_token
3797 }
3798 { }
3799 { \pgfusepath { } }
3800 \pgfnode
3801 { rectangle }
3802 { north }
3803 {
3804     \c_math_toggle_token
3805     \scriptstyle \l_@@_xdots_down_tl
3806     \c_math_toggle_token
3807 }
3808 { }
3809 { \pgfusepath { } }
3810 \endpgfscope
3811 }
3812 \group_begin:

```

The dimension `\l_@@_l_dim` is the length  $\ell$  of the line to draw. We use the floating point reals of the L3 programming layer to compute this length.

```

3813 \dim_zero_new:N \l_@@_l_dim
3814 \dim_set:Nn \l_@@_l_dim
3815 {
3816     \fp_to_dim:n
3817     {
3818         sqrt
3819         (
3820             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
3821             +
3822             ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
3823         )
3824     }
3825 }

```

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.

```

3826 \bool_lazy_or:nnF
3827 { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
3828 { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
3829 \@@_draw_standard_dotted_line_i:
3830 \group_end:
3831 }
3832 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
3833 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
3834 {

```

The number of dots will be  $\backslash l\_tmpa\_int + 1$ .

```

3835   \bool_if:NTF \l_@@_initial_open_bool
3836   {
3837     \bool_if:NTF \l_@@_final_open_bool
3838     {
3839       \int_set:Nn \l_tmpa_int
3840       { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
3841     }
3842     {
3843       \int_set:Nn \l_tmpa_int
3844       {
3845         \dim_ratio:nn
3846         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
3847         \l_@@_inter_dots_dim
3848       }
3849     }
3850   }
3851   {
3852     \bool_if:NTF \l_@@_final_open_bool
3853     {
3854       \int_set:Nn \l_tmpa_int
3855       {
3856         \dim_ratio:nn
3857         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
3858         \l_@@_inter_dots_dim
3859       }
3860     }
3861     {
3862       \int_set:Nn \l_tmpa_int
3863       {
3864         \dim_ratio:nn
3865         { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
3866         \l_@@_inter_dots_dim
3867       }
3868     }
3869   }

```

The dimensions  $\backslash l\_tmpa\_dim$  and  $\backslash l\_tmpb\_dim$  are the coordinates of the vector between two dots in the dotted line.

```

3870   \dim_set:Nn \l_tmpa_dim
3871   {
3872     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3873     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
3874   }
3875   \dim_set:Nn \l_tmpb_dim
3876   {
3877     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
3878     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
3879   }

```

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  $\backslash l\_tmpb\_int$ .

```

3880   \int_set:Nn \l_tmpb_int
3881   {
3882     \bool_if:NTF \l_@@_initial_open_bool
3883     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
3884     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
3885   }

```

In the loop over the dots, the dimensions  $\backslash l\_@@\_x\_initial\_dim$  and  $\backslash l\_@@\_y\_initial\_dim$  will be used for the coordinates of the dots. But, before the loop, we must move until the first dot.



```

3886 \dim_gadd:Nn \l_@@_x_initial_dim
3887 {
3888   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3889   \dim_ratio:nn
3890   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3891   { 2 \l_@@_l_dim }
3892   * \l_tmpb_int
3893 }
3894 \dim_gadd:Nn \l_@@_y_initial_dim
3895 {
3896   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
3897   \dim_ratio:nn
3898   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3899   { 2 \l_@@_l_dim }
3900   * \l_tmpb_int
3901 }
3902 \pgf@relevantforpicturesizefalse
3903 \int_step_inline:nnn 0 \l_tmpa_int
3904 {
3905   \pgfpathcircle
3906   { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
3907   { \l_@@_radius_dim }
3908   \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
3909   \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
3910 }
3911 \pgfusepathqfill
3912 }

```

## 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 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.

```

3913 \AtBeginDocument
3914 {
3915   \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
3916   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3917   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
3918   {
3919     \int_compare:nNnTF \c@jCol = 0
3920     { \@@_error:nn { in~first~col } \Ldots }
3921     {
3922       \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3923       { \@@_error:nn { in~last~col } \Ldots }
3924       {
3925         \@@_instruction_of_type:nnn \c_false_bool { \Ldots }
3926         { #1 , down = #2 , up = #3 }
3927       }
3928     }
3929     \bool_if:NF \l_@@_nullify_dots_bool
3930     { \phantom { \ensuremath { \@@_old_ldots } } }
3931     \bool_gset_true:N \g_@@_empty_cell_bool
3932   }

```

```

3933 \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
3934 {
3935   \int_compare:nNnTF \c@jCol = 0
3936   { \@@_error:nn { in~first~col } \Cdots }
3937   {
3938     \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3939     { \@@_error:nn { in~last~col } \Cdots }
3940     {
3941       \@@_instruction_of_type:nnn \c_false_bool { Cdots }
3942       { #1 , down = #2 , up = #3 }
3943     }
3944   }
3945   \bool_if:NF \l_@@_nullify_dots_bool
3946   { \phantom { \ensuremath { \@@_old_cdots } } }
3947   \bool_gset_true:N \g_@@_empty_cell_bool
3948 }

3949 \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
3950 {
3951   \int_compare:nNnTF \c@iRow = 0
3952   { \@@_error:nn { in~first~row } \Vdots }
3953   {
3954     \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
3955     { \@@_error:nn { in~last~row } \Vdots }
3956     {
3957       \@@_instruction_of_type:nnn \c_false_bool { Vdots }
3958       { #1 , down = #2 , up = #3 }
3959     }
3960   }
3961   \bool_if:NF \l_@@_nullify_dots_bool
3962   { \phantom { \ensuremath { \@@_old_vdots } } }
3963   \bool_gset_true:N \g_@@_empty_cell_bool
3964 }

3965 \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
3966 {
3967   \int_case:nnF \c@iRow
3968   {
3969     0 { \@@_error:nn { in~first~row } \Ddots }
3970     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
3971   }
3972   {
3973     \int_case:nnF \c@jCol
3974     {
3975       0 { \@@_error:nn { in~first~col } \Ddots }
3976       \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
3977     }
3978     {
3979       \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3980       \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Ddots }
3981       { #1 , down = #2 , up = #3 }
3982     }
3983   }
3984 }
3985 \bool_if:NF \l_@@_nullify_dots_bool
3986 { \phantom { \ensuremath { \@@_old_ddots } } }
3987 \bool_gset_true:N \g_@@_empty_cell_bool
3988 }

3989 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
3990 {

```

```

3991 \int_case:nnF \c@iRow
3992 {
3993     0 { \@@_error:nn { in~first~row } \Iddots }
3994     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
3995 }
3996 {
3997     \int_case:nnF \c@jCol
3998     {
3999         0 { \@@_error:nn { in~first~col } \Iddots }
4000         \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
4001     }
4002     {
4003         \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
4004         \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Iddots }
4005         { #1 , down = #2 , up = #3 }
4006     }
4007 }
4008 \bool_if:NF \l_@@_nullify_dots_bool
4009 { \phantom { \ensuremath { \@@_old_iddots } } }
4010 \bool_gset_true:N \g_@@_empty_cell_bool
4011 }
4012 }

```

End of the `\AtBeginDocument`.

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

```

4013 \keys_define:nn { NiceMatrix / Ddots }
4014 {
4015     draw-first .bool_set:N = \l_@@_draw_first_bool ,
4016     draw-first .default:n = true ,
4017     draw-first .value_forbidden:n = true
4018 }

```

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

```

4019 \cs_new_protected:Npn \@@_Hspace:
4020 {
4021     \bool_gset_true:N \g_@@_empty_cell_bool
4022     \hspace
4023 }

```

In the environments of `nicematrix`, the command `\multicolumn` is redefined. We will patch the environment `{tabular}` to go back to the previous value of `\multicolumn`.

```

4024 \cs_set_eq:NN \@@_old_multicolumn \multicolumn

```

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`.

```

4025 \cs_new:Npn \@@_Hdotsfor:
4026 {
4027     \bool_lazy_and:nnTF
4028     { \int_compare_p:nNn \c@jCol = 0 }
4029     { \int_compare_p:nNn \l_@@_first_col_int = 0 }
4030     {
4031         \bool_if:NTF \g_@@_after_col_zero_bool
4032         {
4033             \multicolumn { 1 } { c } { }
4034             \@@_Hdotsfor_i
4035         }
4036         { \@@_fatal:n { Hdotsfor~in~col~0 } }
4037     }
4038 }

```

```

4039     \multicolumn { 1 } { c } { }
4040     \@@_Hdotsfor_i
4041   }
4042 }

```

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:`).

```

4043 \AtBeginDocument
4044 {
4045   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
4046   \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.

```

4047   \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
4048   {
4049     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
4050     {
4051       \@@_Hdotsfor:nnnn
4052       { \int_use:N \c@iRow }
4053       { \int_use:N \c@jCol }
4054       { #2 }
4055       {
4056         #1 , #3 ,
4057         down = \exp_not:n { #4 } ,
4058         up = \exp_not:n { #5 }
4059       }
4060     }
4061     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }
4062   }
4063 }

```

Enf of `\AtBeginDocument`.

```

4064 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
4065 {
4066   \bool_set_false:N \l_@@_initial_open_bool
4067   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

4068   \int_set:Nn \l_@@_initial_i_int { #1 }
4069   \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

4070   \int_compare:nNnTF { #2 } = 1
4071   {
4072     \int_set:Nn \l_@@_initial_j_int 1
4073     \bool_set_true:N \l_@@_initial_open_bool
4074   }
4075   {
4076     \cs_if_exist:cTF
4077     {
4078       pgf @ sh @ ns @ \@@_env:
4079       - \int_use:N \l_@@_initial_i_int
4080       - \int_eval:n { #2 - 1 }
4081     }
4082     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
4083     {
4084       \int_set:Nn \l_@@_initial_j_int { #2 }
4085       \bool_set_true:N \l_@@_initial_open_bool
4086     }
4087   }
4088   \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
4089   {

```

```

4090     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
4091     \bool_set_true:N \l_@@_final_open_bool
4092   }
4093   {
4094     \cs_if_exist:cTF
4095     {
4096       pgf @ sh @ ns @ \@@_env:
4097       - \int_use:N \l_@@_final_i_int
4098       - \int_eval:n { #2 + #3 }
4099     }
4100     { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
4101     {
4102       \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
4103       \bool_set_true:N \l_@@_final_open_bool
4104     }
4105   }
4106   \group_begin:
4107   \int_compare:nNnTF { #1 } = 0
4108   { \color { nicematrix-first-row } }
4109   {
4110     \int_compare:nNnT { #1 } = \g_@@_row_total_int
4111     { \color { nicematrix-last-row } }
4112   }
4113   \keys_set:nn { NiceMatrix / xdots } { #4 }
4114   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
4115   \@@_actually_draw_Ldots:
4116   \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).

```

4117     \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
4118     { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
4119   }

4120   \AtBeginDocument
4121   {
4122     \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
4123     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
4124     \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
4125     {
4126       \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
4127       {
4128         \@@_Vdotsfor:nnnn
4129         { \int_use:N \c@iRow }
4130         { \int_use:N \c@jCol }
4131         { #2 }
4132         {
4133           #1 , #3 ,
4134           down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
4135         }
4136       }
4137     }
4138   }

```

Enf of `\AtBeginDocument`.

```

4139   \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
4140   {
4141     \bool_set_false:N \l_@@_initial_open_bool
4142     \bool_set_false:N \l_@@_final_open_bool

```

For the column, it's easy.

```
4143 \int_set:Nn \l_@@_initial_j_int { #2 }
4144 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int
```

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

```
4145 \int_compare:nNnTF #1 = 1
4146 {
4147   \int_set:Nn \l_@@_initial_i_int 1
4148   \bool_set_true:N \l_@@_initial_open_bool
4149 }
4150 {
4151   \cs_if_exist:cTF
4152   {
4153     pgf @ sh @ ns @ \@@_env:
4154     - \int_eval:n { #1 - 1 }
4155     - \int_use:N \l_@@_initial_j_int
4156   }
4157   { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
4158   {
4159     \int_set:Nn \l_@@_initial_i_int { #1 }
4160     \bool_set_true:N \l_@@_initial_open_bool
4161   }
4162 }
4163 \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
4164 {
4165   \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
4166   \bool_set_true:N \l_@@_final_open_bool
4167 }
4168 {
4169   \cs_if_exist:cTF
4170   {
4171     pgf @ sh @ ns @ \@@_env:
4172     - \int_eval:n { #1 + #3 }
4173     - \int_use:N \l_@@_final_j_int
4174   }
4175   { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
4176   {
4177     \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
4178     \bool_set_true:N \l_@@_final_open_bool
4179   }
4180 }
4181 \group_begin:
4182 \int_compare:nNnTF { #2 } = 0
4183 { \color { nicematrix-first-col } }
4184 {
4185   \int_compare:nNnT { #2 } = \g_@@_col_total_int
4186   { \color { nicematrix-last-col } }
4187 }
4188 \keys_set:nn { NiceMatrix / xdots } { #4 }
4189 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
4190 \@@_actually_draw_Vdots:
4191 \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).

```
4192 \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
4193 { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
4194 }
```

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

```
4195 \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>65</sup>

```
4196 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
4197 { \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).

```
4198 \AtBeginDocument
4199 {
4200   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
4201   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
4202   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
4203   {
4204     \group_begin:
4205     \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
4206     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
4207     \use:e
4208     {
4209       \@@_line_i:nn
4210       { \@@_double_int_eval:n #2 \q_stop }
4211       { \@@_double_int_eval:n #3 \q_stop }
4212     }
4213     \group_end:
4214   }
4215 }

4216 \cs_new_protected:Npn \@@_line_i:nn #1 #2
4217 {
4218   \bool_set_false:N \l_@@_initial_open_bool
4219   \bool_set_false:N \l_@@_final_open_bool
4220   \bool_if:nTF
4221   {
4222     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
4223     ||
4224     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
4225   }
4226   {
4227     \@@_error:nnn { unknown~cell~for~line~in~CodeAfter } { #1 } { #2 }
4228   }
4229   { \@@_draw_line_ii:nn { #1 } { #2 } }
4230 }

4231 \AtBeginDocument
4232 {
4233   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
4234   {
```

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:.`

---

<sup>65</sup>Indeed, we want that the user may use the command `\line` in `\CodeAfter` with LaTeX counters in the arguments — with the command `\value`.

```

4235     \c_@@_pgfortikzpicture_tl
4236     \@@_draw_line_iii:nn { #1 } { #2 }
4237     \c_@@_endpgfortikzpicture_tl
4238   }
4239 }

```

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

```

4240 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
4241 {
4242   \pgfrememberpicturepositiononpagetrue
4243   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
4244   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
4245   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
4246   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
4247   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
4248   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
4249   \@@_draw_line:
4250 }

```

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`, `\@@_rectanglecolor` and `\@@_rowlistcolors` 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 `\@@_cartesian_color:nn` and `\@@_rectanglecolor:nn`.

`#1` is the color and `#2` is an instruction using that color. Despite its name, the command `\@@_add_to_colors_seq:nn` 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).

```

4251 \cs_new_protected:Npn \@@_add_to_colors_seq:nn #1 #2
4252 {

```

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.

```

4253   \int_zero:N \l_tmpa_int
4254   \seq_map_indexed_inline:Nn \g_@@_colors_seq
4255     { \tl_if_eq:nnT { #1 } { ##2 } { \int_set:Nn \l_tmpa_int { ##1 } } }
4256   \int_compare:nNnTF \l_tmpa_int = \c_zero_int

```

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

```

4257     {
4258       \seq_gput_right:Nn \g_@@_colors_seq { #1 }
4259       \tl_gset:cx { g_@@_color _ \seq_count:N \g_@@_colors_seq _ tl } { #2 }
4260     }

```



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

```

4261     { \tl_gput_right:cx { g_@@_color _ \int_use:N \l_tmpa_int _tl } { #2 } }
4262   }

4263 \cs_generate_variant:Nn \@@_add_to_colors_seq:nn { x n }
4264 \cs_generate_variant:Nn \@@_add_to_colors_seq:nn { x x }

```

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`).

```

4265 \cs_new_protected:Npn \@@_actually_color:
4266 {
4267   \pgfpicture
4268   \pgf@relevantforpicturesizefalse
4269   \seq_map_indexed_inline:Nn \g_@@_colors_seq
4270   {
4271     \color ##2
4272     \use:c { g_@@_color _ ##1 _tl }
4273     \tl_gclear:c { g_@@_color _ ##1 _tl }
4274     \pgfusepath { fill }
4275   }
4276   \endpgfpicture
4277 }

4278 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
4279 {
4280   \tl_set:Nn \l_tmpa_tl { #1 }
4281   \tl_set:Nn \l_tmpb_tl { #2 }
4282 }

4283 \cs_new_protected:Npn \@@_cartesian_color:nn #1 #2
4284 {
4285   \tl_set:Nn \l_@@_rows_tl { #1 }
4286   \tl_set:Nn \l_@@_cols_tl { #2 }
4287   \@@_cartesian_path:
4288 }

```

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

```

4289 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
4290 {
4291   \tl_if_blank:nF { #2 }
4292   {
4293     \@@_add_to_colors_seq:xn
4294     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
4295     { \@@_cartesian_color:nn { #3 } { - } }
4296   }
4297 }

```

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

```

4298 \NewDocumentCommand \@@_columncolor { 0 { } m m }
4299 {
4300   \tl_if_blank:nF { #2 }
4301   {
4302     \@@_add_to_colors_seq:xn
4303     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
4304     { \@@_cartesian_color:nn { - } { #3 } }
4305   }
4306 }

```

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

```

4307 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
4308 {

```

```

4309 \tl_if_blank:nF { #2 }
4310 {
4311   \@@_add_to_colors_seq:xn
4312   { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
4313   { \@@_rectanglecolor:nnn { #3 } { #4 } { 0 pt } }
4314 }
4315 }

```

The last argument is the radius of the corners of the rectangle.

```

4316 \NewDocumentCommand \@@_roundedrectanglecolor { 0 { } m m m m }
4317 {
4318   \tl_if_blank:nF { #2 }
4319   {
4320     \@@_add_to_colors_seq:xn
4321     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
4322     { \@@_rectanglecolor:nnn { #3 } { #4 } { #5 } }
4323   }
4324 }

```

The last argument is the radius of the corners of the rectangle.

```

4325 \cs_new_protected:Npn \@@_rectanglecolor:nnn #1 #2 #3
4326 {
4327   \@@_cut_on_hyphen:w #1 \q_stop
4328   \tl_clear_new:N \l_tmpc_tl
4329   \tl_clear_new:N \l_tmpd_tl
4330   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
4331   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
4332   \@@_cut_on_hyphen:w #2 \q_stop
4333   \tl_set:Nx \l_@@_rows_tl { \l_tmpc_tl - \l_tmpa_tl }
4334   \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`.

```

4335 \@@_cartesian_path:n { #3 }
4336 }

```

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

```

4337 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
4338 {
4339   \clist_map_inline:nn { #3 }
4340   { \@@_rectanglecolor [ #1 ] { #2 } { ##1 } { ##1 } }
4341 }

```

```

4342 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
4343 {
4344   \int_step_inline:nn { \int_use:N \c@iRow }
4345   {
4346     \int_step_inline:nn { \int_use:N \c@jCol }
4347     {
4348       \int_if_even:nTF { ####1 + ##1 }
4349       { \@@_cellcolor [ #1 ] { #2 } }
4350       { \@@_cellcolor [ #1 ] { #3 } }
4351       { ##1 - ####1 }
4352     }
4353   }
4354 }

```

```

4355 \keys_define:nn { NiceMatrix / arraycolor }
4356 { 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.

```

4357 \NewDocumentCommand \@@_arraycolor { 0 { } m 0 { } }
4358 {
4359   \keys_set:nn { NiceMatrix / arraycolor } { #3 }
4360   \@@_rectanglecolor [ #1 ] { #2 }
4361   { 1 - 1 }
4362   { \int_use:N \c@iRow - \int_use:N \c@jCol }
4363 }

4364 \keys_define:nn { NiceMatrix / rowcolors }
4365 {
4366   respect-blocks .bool_set:N = \l_@@_respect_blocks_bool ,
4367   respect-blocks .default:n = true ,
4368   cols .tl_set:N = \l_@@_cols_tl ,
4369   restart .bool_set:N = \l_@@_rowcolors_restart_bool ,
4370   restart .default:n = true ,
4371   unknown .code:n = \@@_error:n { Unknown-key-for-rowcolors }
4372 }

```

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 list of colors ; **#4** is for the optional list of pairs key-value.

```

4373 \NewDocumentCommand \@@_rowlistcolors { 0 { } m m 0 { } }
4374 {

```

The group is for the options. `\l_@@_colors_seq` will be the list of colors.

```

4375   \group_begin:
4376   \seq_clear_new:N \l_@@_colors_seq
4377   \seq_set_split:Nnn \l_@@_colors_seq { , } { #3 }
4378   \tl_clear_new:N \l_@@_cols_tl
4379   \tl_set:Nn \l_@@_cols_tl { - }
4380   \keys_set:nn { NiceMatrix / rowcolors } { #4 }

```

The counter `\l_@@_color_int` will be the rank of the current color in the list of colors (modulo the length of the list).

```

4381   \int_zero_new:N \l_@@_color_int
4382   \int_set:Nn \l_@@_color_int 1
4383   \bool_if:NT \l_@@_respect_blocks_bool
4384   {

```

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`).

```

4385     \seq_set_eq:NN \l_tmpb_seq \g_@@_pos_of_blocks_seq
4386     \seq_set_filter:Nnn \l_tmpa_seq \l_tmpb_seq
4387     { \@@_not_in_exterior_p:nnnn ##1 }
4388   }
4389   \pgfpicture
4390   \pgf@relevantforpicturesizefalse

```

**#2** is the list of intervals of rows.

```

4391   \clist_map_inline:nn { #2 }
4392   {
4393     \tl_set:Nn \l_tmpa_tl { ##1 }
4394     \tl_if_in:NnTF \l_tmpa_tl { - }
4395     { \@@_cut_on_hyphen:w ##1 \q_stop }
4396     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }

```

Now, `l_tmpa_tl` and `l_tmppb_tl` are the first row and the last row of the interval of rows that we have to treat. The counter `\l_tmpa_int` will be the index of the loop over the rows.

```

4397     \int_set:Nn \l_tmpa_int \l_tmpa_tl
4398     \bool_if:NTF \l_@@_rowcolors_restart_bool
4399       { \int_set:Nn \l_@@_color_int 1 }
4400       { \int_set:Nn \l_@@_color_int \l_tmpa_tl }
4401     \int_zero_new:N \l_tmppc_int
4402     \int_set:Nn \l_tmppc_int \l_tmppb_tl
4403     \int_do_until:nNnn \l_tmpa_int > \l_tmppc_int
4404     {

```

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

```

4405         \int_set_eq:NN \l_tmppb_int \l_tmpa_int

```

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

```

4406         \bool_if:NT \l_@@_respect_blocks_bool
4407         {
4408             \seq_set_filter:NNn \l_tmppb_seq \l_tmpa_seq
4409             { \@@_intersect_our_row_p:nnnn ###1 }
4410             \seq_map_inline:Nn \l_tmppb_seq { \@@_rowcolors_i:nnnn ###1 }

```

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

```

4411         }
4412         \tl_set:Nx \l_@@_rows_tl
4413         { \int_use:N \l_tmpa_int - \int_use:N \l_tmppb_int }

```

`\l_tmppc_tl` will be the color that we will use.

```

4414         \tl_clear_new:N \l_@@_color_tl
4415         \tl_set:Nx \l_@@_color_tl
4416         {
4417             \@@_color_index:n
4418             {
4419                 \int_mod:nn
4420                 { \l_@@_color_int - 1 }
4421                 { \seq_count:N \l_@@_colors_seq }
4422                 + 1
4423             }
4424         }
4425         \tl_if_empty:NF \l_@@_color_tl
4426         {
4427             \@@_add_to_colors_seq:xx
4428             { \tl_if_blank:nF { #1 } { [ #1 ] } { \l_@@_color_tl } }
4429             { \@@_cartesian_color:nn { \l_@@_rows_tl } { \l_@@_cols_tl } }
4430         }
4431         \int_incr:N \l_@@_color_int
4432         \int_set:Nn \l_tmpa_int { \l_tmppb_int + 1 }
4433     }
4434 }
4435 \endpgfpicture
4436 \group_end:
4437 }

```

The command `\@@_color_index:n` peeks in `\l_@@_colors_seq` the color at the index #1. However, if that color is the symbol `=`, the previous one is poken. This macro is recursive.

```

4438 \cs_new:Npn \@@_color_index:n #1
4439 {
4440     \str_if_eq:eeTF { \seq_item:Nn \l_@@_colors_seq { #1 } } { = }
4441     { \@@_color_index:n { #1 - 1 } }
4442     { \seq_item:Nn \l_@@_colors_seq { #1 } }
4443 }

```

The command `\rowcolors` (available in the `\CodeBefore`) is a specialisation of the most general command `\rowlistcolors`.

```

4444 \NewDocumentCommand \@@_rowcolors { 0 { } m m m 0 { } }
4445 { \@@_rowlistcolors [ #1 ] { #2 } { { #3 } , { #4 } } [ #5 ] }

```

```

4446 \cs_new_protected:Npn \@@_rowcolors_i:nnnn #1 #2 #3 #4
4447 {
4448   \int_compare:nNt { #3 } > \l_tmpb_int
4449   { \int_set:Nn \l_tmpb_int { #3 } }
4450 }

4451 \prg_new_conditional:Nnn \@@_not_in_exterior:nnnn p
4452 {
4453   \bool_lazy_or:nnTF
4454   { \int_compare_p:nNn { #4 } = \c_zero_int }
4455   { \int_compare_p:nNn { #2 } = { \@@_succ:n { \c@jCol } } }
4456   \prg_return_false:
4457   \prg_return_true:
4458 }

```

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

```

4459 \prg_new_conditional:Nnn \@@_intersect_our_row:nnnn p
4460 {
4461   \bool_if:nTF
4462   {
4463     \int_compare_p:n { #1 <= \l_tmpa_int }
4464     &&
4465     \int_compare_p:n { \l_tmpa_int <= #3 }
4466   }
4467   \prg_return_true:
4468   \prg_return_false:
4469 }

```

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).

```

4470 \cs_new_protected:Npn \@@_cartesian_path:n #1
4471 {
4472   \bool_lazy_and:nnT
4473   { ! \seq_if_empty_p:N \l_@@_corners_cells_seq }
4474   { \dim_compare_p:nNn { #1 } = \c_zero_dim }
4475   {
4476     \@@_expand_clist:NN \l_@@_cols_tl \c@jCol
4477     \@@_expand_clist:NN \l_@@_rows_tl \c@iRow
4478   }

```

We begin the loop over the columns.

```

4479 \clist_map_inline:Nn \l_@@_cols_tl
4480 {
4481   \tl_set:Nn \l_tmpa_tl { ##1 }
4482   \tl_if_in:NnTF \l_tmpa_tl { - }
4483   { \@@_cut_on_hyphen:w ##1 \q_stop }
4484   { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
4485   \bool_lazy_or:nnT
4486   { \tl_if_blank_p:V \l_tmpa_tl }
4487   { \str_if_eq_p:Vn \l_tmpa_tl { * } }
4488   { \tl_set:Nn \l_tmpa_tl { 1 } }
4489   \bool_lazy_or:nnT
4490   { \tl_if_blank_p:V \l_tmpb_tl }
4491   { \str_if_eq_p:Vn \l_tmpb_tl { * } }
4492   { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
4493   \int_compare:nNt \l_tmpb_tl > \c@jCol
4494   { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }

```

\l\_tmpc\_tl will contain the number of column.

```
4495 \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.

```
4496 \@@_qpoint:n { col - \l_tmpa_tl }
4497 \int_compare:nNnTF \l_@@_first_col_int = \l_tmpa_tl
4498 { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
4499 { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
4500 \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
4501 \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }
```

We begin the loop over the rows.

```
4502 \clist_map_inline:Nn \l_@@_rows_tl
4503 {
4504   \tl_set:Nn \l_tmpa_tl { #####1 }
4505   \tl_if_in:NnTF \l_tmpa_tl { - }
4506   { \@@_cut_on_hyphen:w #####1 \q_stop }
4507   { \@@_cut_on_hyphen:w #####1 - #####1 \q_stop }
4508   \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
4509   \tl_if_empty:NT \l_tmpb_tl
4510   { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
4511   \int_compare:nNnT \l_tmpb_tl > \c@iRow
4512   { \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.

```
4513 \seq_if_in:NxF \l_@@_corners_cells_seq
4514 { \l_tmpa_tl - \l_tmpc_tl }
4515 {
4516   \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
4517   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
4518   \@@_qpoint:n { row - \l_tmpa_tl }
4519   \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
4520   \pgfsetcornersarced { \pgfpoint { #1 } { #1 } }
4521   \pgfpathrectanglecorners
4522   { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
4523   { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4524 }
4525 }
4526 }
4527 }
```

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).

```
4528 \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.

```
4529 \cs_new_protected:Npn \@@_expand_clist:NN #1 #2
4530 {
4531   \clist_set_eq:NN \l_tmpa_clist #1
4532   \clist_clear:N #1
4533   \clist_map_inline:Nn \l_tmpa_clist
4534   {
4535     \tl_set:Nn \l_tmpa_tl { ##1 }
4536     \tl_if_in:NnTF \l_tmpa_tl { - }
4537     { \@@_cut_on_hyphen:w ##1 \q_stop }
4538     { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
4539     \bool_lazy_or:nnT
4540     { \tl_if_blank_p:V \l_tmpa_tl }
4541     { \str_if_eq_p:Vn \l_tmpa_tl { * } }
```

```

4542     { \tl_set:Nn \l_tmpa_tl { 1 } }
4543   \bool_lazy_or:nnT
4544     { \tl_if_blank_p:V \l_tmpb_tl }
4545     { \str_if_eq_p:Vn \l_tmpb_tl { * } }
4546     { \tl_set:Nx \l_tmpb_tl { \int_use:N #2 } }
4547   \int_compare:nNnT \l_tmpb_tl > #2
4548     { \tl_set:Nx \l_tmpb_tl { \int_use:N #2 } }
4549   \int_step_inline:nnn \l_tmpa_tl \l_tmpb_tl
4550     { \clist_put_right:Nn #1 { ####1 } }
4551 }
4552 }

```

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

```

4553 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
4554 {
4555   \peek_remove_spaces:n
4556   {
4557     \tl_gput_right:Nx \g_nicematrix_code_before_tl
4558     {

```

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`).

```

4559       \cellcolor [ #1 ] { \exp_not:n { #2 } }
4560       { \int_use:N \c@iRow - \int_use:N \c@jCol }
4561     }
4562   }
4563 }

```

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

```

4564 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
4565 {
4566   \peek_remove_spaces:n
4567   {
4568     \tl_gput_right:Nx \g_nicematrix_code_before_tl
4569     {
4570       \exp_not:N \rectanglecolor [ #1 ] { \exp_not:n { #2 } }
4571       { \int_use:N \c@iRow - \int_use:N \c@jCol }
4572       { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
4573     }
4574   }
4575 }

```

```

4576 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
4577 {

```

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).

```

4578   \int_compare:nNnT \c@jCol > \g_@@_col_total_int
4579   {

```

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).

```

4580     \tl_gput_left:Nx \g_nicematrix_code_before_tl
4581     {
4582       \exp_not:N \columncolor [ #1 ]
4583       { \exp_not:n { #2 } } { \int_use:N \c@jCol }
4584     }
4585   }
4586 }

```

## 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`.

```
4587 \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`.

```
4588 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
4589 {
4590   \int_compare:nNnTF \l_@@_first_col_int = 0
4591   { \@@_OnlyMainNiceMatrix_i:n { #1 } }
4592   {
4593     \int_compare:nNnTF \c@jCol = 0
4594     {
4595       \int_compare:nNnF \c@iRow = { -1 }
4596       { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
4597     }
4598     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
4599   }
4600 }
```

This definition may seem complicated but 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.

```
4601 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
4602 {
4603   \int_compare:nNnF \c@iRow = 0
4604   { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
4605 }
```

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 `|`. `#3` and `#4` are numbers of rows that define the delimitation of the horizontal rule that we have to draw. If `#4` is empty, that means that the rule extends until the last row.

```
4606 \cs_new_protected:Npn \@@_vline:nnnn #1 #2 #3 #4
4607 {
```

The following test is for the case where the user does not use all the columns specified in the preamble of the environment (for instance, a preamble of `|c|c|c|` but only two columns used).

```
4608   \int_compare:nNnT { #1 } < { \c@jCol + 2 }
4609   {
4610     \pgfpicture
4611     \@@_vline_i:nnnn { #1 } { #2 } { #3 } { #4 }
4612     \endpgfpicture
4613   }
4614 }
4615 \cs_new_protected:Npn \@@_vline_i:nnnn #1 #2 #3 #4
4616 {
```



`\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`.

```

4617 \tl_set:Nx \l_tmpb_tl { #1 }
4618 \tl_clear_new:N \l_tmpc_tl
4619 \int_step_variable:nnNn
4620 { #3 }
4621 { \tl_if_blank:nTF { #4 } { \int_use:N \c@iRow } { #4 } }
4622 \l_tmpa_tl
4623 {

```

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.

```

4624 \bool_gset_true:N \g_tmpa_bool
4625 \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4626 { \@@_test_vline_in_block:nnnn #1 }
4627 \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
4628 { \@@_test_vline_in_block:nnnn #1 }
4629 \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
4630 { \@@_test_vline_in_stroken_block:nnnn #1 }
4631 \clist_if_empty:NF \l_@@_corners_clist \@@_test_in_corner_v:
4632 \bool_if:NTF \g_tmpa_bool
4633 {
4634 \tl_if_empty:NT \l_tmpc_tl

```

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

```

4635 { \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl }
4636 }
4637 {
4638 \tl_if_empty:NF \l_tmpc_tl
4639 {
4640 \@@_vline_ii:nnnn
4641 { #1 }
4642 { #2 }
4643 \l_tmpc_tl
4644 { \int_eval:n { \l_tmpa_tl - 1 } }
4645 \tl_clear:N \l_tmpc_tl
4646 }
4647 }
4648 }
4649 \tl_if_empty:NF \l_tmpc_tl
4650 {
4651 \@@_vline_ii:nnnn
4652 { #1 }
4653 { #2 }
4654 \l_tmpc_tl
4655 { \tl_if_blank:nTF { #4 } { \int_use:N \c@iRow } { #4 } }
4656 \tl_clear:N \l_tmpc_tl
4657 }
4658 }

```

```

4659 \cs_new_protected:Npn \@@_test_in_corner_v:
4660 {
4661 \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
4662 {
4663 \seq_if_in:NxT
4664 \l_@@_corners_cells_seq
4665 { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
4666 { \bool_set_false:N \g_tmpa_bool }
4667 }
4668 {
4669 \seq_if_in:NxT

```

```

4670     \l_@@_corners_cells_seq
4671     { \l_tmpa_tl - \l_tmpb_tl }
4672     {
4673         \int_compare:nNnTF \l_tmpb_tl = 1
4674         { \bool_set_false:N \g_tmpa_bool }
4675         {
4676             \seq_if_in:NxT
4677             \l_@@_corners_cells_seq
4678             { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
4679             { \bool_set_false:N \g_tmpa_bool }
4680         }
4681     }
4682 }
4683 }

```

#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.

```

4684 \cs_new_protected:Npn \@@_vline_ii:nnnn #1 #2 #3 #4
4685 {
4686     \pgfrememberpicturepositiononpagetrue
4687     \pgf@relevantforpicturesizefalse
4688     \@@_qpoint:n { row - #3 }
4689     \dim_set_eq:NN \l_tmpa_dim \pgf@y
4690     \@@_qpoint:n { col - #1 }
4691     \dim_set_eq:NN \l_tmpb_dim \pgf@x
4692     \@@_qpoint:n { row - \@@_succ:n { #4 } }
4693     \dim_set_eq:NN \l_tmpc_dim \pgf@y
4694     \bool_lazy_all:nT
4695     {
4696         { \int_compare_p:nNn { #2 } > 1 }
4697         { \cs_if_exist_p:N \CT@drsc@ } % condition added in version 5.18a
4698         { ! \tl_if_blank_p:V \CT@drsc@ }
4699     }
4700     {
4701         \group_begin:
4702         \CT@drsc@
4703         \dim_add:Nn \l_tmpa_dim { 0.5 \arrayrulewidth }
4704         \dim_sub:Nn \l_tmpc_dim { 0.5 \arrayrulewidth }
4705         \dim_set:Nn \l_tmpd_dim
4706         { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
4707         \pgfpathrectanglecorners
4708         { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4709         { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
4710         \pgfusepath { fill }
4711         \group_end:
4712     }
4713     \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4714     \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4715     \prg_replicate:nn { #2 - 1 }
4716     {
4717         \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
4718         \dim_sub:Nn \l_tmpb_dim \doublerulesep
4719         \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4720         \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4721     }
4722     \CT@arc@
4723     \pgfsetlinewidth { 1.1 \arrayrulewidth }
4724     \pgfsetrectcap
4725     \pgfusepathqstroke
4726 }

```

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).

```
4727 \cs_new_protected:Npn \@@_vline_i_complete:nn #1 #2
4728 { \@@_vline_ii:nnnn { #1 } { #2 } 1 { \int_use:N \c@iRow } }
```

The command `\@@_draw_vlines:` 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).

```
4729 \cs_new_protected:Npn \@@_draw_vlines:
4730 {
4731   \int_step_inline:nnn
4732   {
4733     \bool_if:nTF { \l_@@_NiceArray_bool && ! \l_@@_except_borders_bool }
4734     1 2
4735   }
4736   {
4737     \bool_if:nTF { \l_@@_NiceArray_bool && ! \l_@@_except_borders_bool }
4738     { \@@_succ:n \c@jCol }
4739     \c@jCol
4740   }
4741   {
4742     \tl_if_eq:NnF \l_@@_vlines_clist { all }
4743     { \clist_if_in:NnT \l_@@_vlines_clist { ##1 } }
4744     { \@@_vline:nnnn { ##1 } 1 1 { } }
4745   }
4746 }
```

## 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`. `#3` and `#4` are numbers of columns that define the delimitation of the horizontal rule that we have to draw. If `#4` is empty, that means that the rule extends until the last column.

```
4747 \cs_new_protected:Npn \@@_hline:nnnn #1 #2 #3 #4
4748 {
4749   \pgfpicture
4750   \@@_hline_i:nnnn { #1 } { #2 } { #3 } { #4 }
4751   \endpgfpicture
4752 }
4753 \cs_new_protected:Npn \@@_hline_i:nnnn #1 #2 #3 #4
4754 {
```

`\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`.

```
4755   \tl_set:Nn \l_tmpa_tl { #1 }
4756   \tl_clear_new:N \l_tmppc_tl
4757   \int_step_variable:nnNn
4758   { #3 }
4759   { \tl_if_blank:nTF { #4 } { \int_use:N \c@jCol } { #4 } }
4760   \l_tmpb_tl
4761   {
```

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.

```
4762   \bool_gset_true:N \g_tmpa_bool
4763   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4764   { \@@_test_hline_in_block:nnnn ##1 }
4765   \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
4766   { \@@_test_hline_in_block:nnnn ##1 }
4767   \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
```

```

4768     { \@@_test_hline_in_stroken_block:nnnn ##1 }
4769     \clist_if_empty:NF \l_@@_corners_clist \@@_test_in_corner_h:
4770     \bool_if:NTF \g_tmpa_bool
4771     {
4772         \tl_if_empty:NT \l_tmpc_tl
We keep in memory that we have a rule to draw.
4773         { \tl_set_eq:NN \l_tmpc_tl \l_tmpb_tl }
4774     }
4775     {
4776         \tl_if_empty:NF \l_tmpc_tl
4777         {
4778             \@@_hline_ii:nnnn
4779             { #1 }
4780             { #2 }
4781             \l_tmpc_tl
4782             { \int_eval:n { \l_tmpb_tl - 1 } }
4783             \tl_clear:N \l_tmpc_tl
4784         }
4785     }
4786 }
4787 \tl_if_empty:NF \l_tmpc_tl
4788 {
4789     \@@_hline_ii:nnnn
4790     { #1 }
4791     { #2 }
4792     \l_tmpc_tl
4793     { \tl_if_blank:nTF { #4 } { \int_use:N \c@jCol } { #4 } }
4794     \tl_clear:N \l_tmpc_tl
4795 }
4796 }

```

```

4797 \cs_new_protected:Npn \@@_test_in_corner_h:
4798 {
4799     \int_compare:nNnTF \l_tmpa_tl = { \@@_succ:n \c@iRow }
4800     {
4801         \seq_if_in:NxT
4802         \l_@@_corners_cells_seq
4803         { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
4804         { \bool_set_false:N \g_tmpa_bool }
4805     }
4806     {
4807         \seq_if_in:NxT
4808         \l_@@_corners_cells_seq
4809         { \l_tmpa_tl - \l_tmpb_tl }
4810         {
4811             \int_compare:nNnTF \l_tmpa_tl = 1
4812             { \bool_set_false:N \g_tmpa_bool }
4813             {
4814                 \seq_if_in:NxT
4815                 \l_@@_corners_cells_seq
4816                 { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
4817                 { \bool_set_false:N \g_tmpa_bool }
4818             }
4819         }
4820     }
4821 }

```

#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.

```

4822 \cs_new_protected:Npn \@@_hline_ii:nnnn #1 #2 #3 #4
4823 {

```

```

4824 \pgfrememberpicturepositiononpagetrue
4825 \pgf@relevantforpicturesizefalse
4826 \@@_qpoint:n { col - #3 }
4827 \dim_set_eq:NN \l_tmpa_dim \pgf@x
4828 \@@_qpoint:n { row - #1 }
4829 \dim_set_eq:NN \l_tmpb_dim \pgf@y
4830 \@@_qpoint:n { col - \@@_succ:n { #4 } }
4831 \dim_set_eq:NN \l_tmpc_dim \pgf@x
4832 \bool_lazy_all:nT
4833 {
4834   { \int_compare_p:nNn { #2 } > 1 }
4835   { \cs_if_exist_p:N \CT@drsc@ } % condition added in version 6.0
4836   { ! \tl_if_blank_p:V \CT@drsc@ }
4837 }
4838 {
4839   \group_begin:
4840   \CT@drsc@
4841   \dim_set:Nn \l_tmpd_dim
4842     { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
4843   \pgfpathrectanglecorners
4844     { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4845     { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
4846   \pgfusepathqfill
4847   \group_end:
4848 }
4849 \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4850 \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4851 \prg_replicate:nn { #2 - 1 }
4852 {
4853   \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
4854   \dim_sub:Nn \l_tmpb_dim \doublerulesep
4855   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4856   \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4857 }
4858 \CT@arc@
4859 \pgfsetlinewidth { 1.1 \arrayrulewidth }
4860 \pgfsetrectcap
4861 \pgfusepathqstroke
4862 }

4863 \cs_new_protected:Npn \@@_hline_i_complete:nn #1 #2
4864 { \@@_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).

```

4865 \cs_new_protected:Npn \@@_draw_hlines:
4866 {
4867   \int_step_inline:nnn
4868   {
4869     \bool_if:nTF { \l_@@_NiceArray_bool && ! \l_@@_except_borders_bool }
4870     1 2
4871   }
4872   {
4873     \bool_if:nTF { \l_@@_NiceArray_bool && ! \l_@@_except_borders_bool }
4874     { \@@_succ:n \c@iRow }
4875     \c@iRow
4876   }
4877   {
4878     \tl_if_eq:NnF \l_@@_hlines_clist { all }
4879     { \clist_if_in:NnT \l_@@_hlines_clist { ##1 } }
4880     { \@@_hline:nnnn { ##1 } 1 1 { } }

```

```

4881     }
4882 }

```

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

```

4883 \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.

```

4884 \cs_set:Npn \@@_Hline_i:n #1
4885 {
4886   \peek_meaning_ignore_spaces:NTF \Hline
4887   { \@@_Hline_ii:nn { #1 + 1 } }
4888   { \@@_Hline_iii:n { #1 } }
4889 }
4890 \cs_set:Npn \@@_Hline_ii:nn #1 #2 { \@@_Hline_i:n { #1 } }
4891 \cs_set:Npn \@@_Hline_iii:n #1
4892 {
4893   \skip_vertical:n
4894   {
4895     \arrayrulewidth * ( #1 )
4896     + \doublerulesep * ( \int_max:nn 0 { #1 - 1 } )
4897   }
4898   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4899   { \@@_hline:nnnn { \@@_succ:n { \c@iRow } } { #1 } 1 { } }
4900   \ifnum 0 = ` { \fi }
4901 }

```

## 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`.

```

4902 \cs_new_protected:Npn \@@_test_hline_in_block:nnnn #1 #2 #3 #4
4903 {
4904   \bool_lazy_all:nT
4905   {
4906     { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
4907     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4908     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4909     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4910   }
4911   { \bool_gset_false:N \g_tmpa_bool }
4912 }

```

The same for vertical rules.

```

4913 \cs_new_protected:Npn \@@_test_vline_in_block:nnnn #1 #2 #3 #4
4914 {
4915   \bool_lazy_all:nT
4916   {
4917     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4918     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4919     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
4920     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4921   }
4922   { \bool_gset_false:N \g_tmpa_bool }
4923 }
4924 \cs_new_protected:Npn \@@_test_hline_in_stroken_block:nnnn #1 #2 #3 #4
4925 {
4926   \bool_lazy_all:nT
4927   {
4928     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }

```

```

4929     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 2 } }
4930     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4931     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4932   }
4933   { \bool_gset_false:N \g_tmpa_bool }
4934 }
4935 \cs_new_protected:Npn \@@_test_vline_in_stroken_block:nnnn #1 #2 #3 #4
4936 {
4937   \bool_lazy_all:nT
4938   {
4939     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4940     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4941     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4942     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 2 } }
4943   }
4944   { \bool_gset_false:N \g_tmpa_bool }
4945 }

```

## 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.

```

4946 \cs_new_protected:Npn \@@_compute_corners:
4947 {

```

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.

```

4948   \seq_clear_new:N \l_@@_corners_cells_seq
4949   \clist_map_inline:Nn \l_@@_corners_clist
4950   {
4951     \str_case:nnF { ##1 }
4952     {
4953       { NW }
4954       { \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol }
4955       { NE }
4956       { \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1 }
4957       { SW }
4958       { \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol }
4959       { SE }
4960       { \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1 }
4961     }
4962     { \@@_error:nn { bad~corner } { ##1 } }
4963   }

```

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.

```

4964   \seq_if_empty:NF \l_@@_corners_cells_seq
4965   {

```

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.

```

4966     \tl_gput_right:Nx \g_@@_aux_tl
4967     {
4968       \seq_set_from_clist:Nn \exp_not:N \l_@@_corners_cells_seq
4969       { \seq_use:Nnnn \l_@@_corners_cells_seq , , , }
4970     }
4971   }
4972 }

```

“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.

```
4973 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
4974 {
```

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.

```
4975   \bool_set_false:N \l_tmpa_bool
4976   \int_zero_new:N \l_@@_last_empty_row_int
4977   \int_set:Nn \l_@@_last_empty_row_int { #1 }
4978   \int_step_inline:nnnn { #1 } { #3 } { #5 }
4979   {
4980     \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
4981     \bool_lazy_or:nnTF
4982     {
4983       \cs_if_exist_p:c
4984       { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
4985     }
4986     \l_tmpb_bool
4987     { \bool_set_true:N \l_tmpa_bool }
4988     {
4989       \bool_if:NF \l_tmpa_bool
4990       { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
4991     }
4992   }
```

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

```
4993   \bool_set_false:N \l_tmpa_bool
4994   \int_zero_new:N \l_@@_last_empty_column_int
4995   \int_set:Nn \l_@@_last_empty_column_int { #2 }
4996   \int_step_inline:nnnn { #2 } { #4 } { #6 }
4997   {
4998     \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
4999     \bool_lazy_or:nnTF
5000     \l_tmpb_bool
5001     {
5002       \cs_if_exist_p:c
5003       { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
5004     }
5005     { \bool_set_true:N \l_tmpa_bool }
5006     {
5007       \bool_if:NF \l_tmpa_bool
5008       { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
5009     }
5010   }
```

Now, we loop over the rows.

```
5011   \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
5012   {
```



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

```

5013     \bool_set_false:N \l_tmpa_bool
5014     \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
5015     {
5016         \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
5017         \bool_lazy_or:nnTF
5018             \l_tmpb_bool
5019             {
5020                 \cs_if_exist_p:c
5021                     { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
5022             }
5023         { \bool_set_true:N \l_tmpa_bool }
5024         {
5025             \bool_if:NF \l_tmpa_bool
5026             {
5027                 \int_set:Nn \l_@@_last_empty_column_int { #####1 }
5028                 \seq_put_right:Nn
5029                     \l_@@_corners_cells_seq
5030                     { ##1 - #####1 }
5031             }
5032         }
5033     }
5034 }
5035 }
```

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`).

```

5036 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
5037 {
5038     \int_set:Nn \l_tmpa_int { #1 }
5039     \int_set:Nn \l_tmpb_int { #2 }
5040     \bool_set_false:N \l_tmpb_bool
5041     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
5042         { \@@_test_if_cell_in_block:nnnnnnn \l_tmpa_int \l_tmpb_int ##1 }
5043 }
5044 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnnn #1 #2 #3 #4 #5 #6
5045 {
5046     \int_compare:nNnT { #3 } < { \@@_succ:n { #1 } }
5047     {
5048         \int_compare:nNnT { #1 } < { \@@_succ:n { #5 } }
5049         {
5050             \int_compare:nNnT { #4 } < { \@@_succ:n { #2 } }
5051             {
5052                 \int_compare:nNnT { #2 } < { \@@_succ:n { #6 } }
5053                 { \bool_set_true:N \l_tmpb_bool }
5054             }
5055         }
5056     }
5057 }
```

## 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`.

```

5058 \cs_new:Npn \@@_hdottedline:
5059 {
```

```

5060 \noalign { \skip_vertical:N 2\l_@@_radius_dim }
5061 \@@_hdottedline_i:
5062 }

```

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

```

5063 \cs_new_protected:Npn \@@_hdottedline_i:
5064 {

```

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).

```

5065 \tl_gput_right:Nx \g_@@_internal_code_after_tl
5066 { \@@_hdottedline:n { \int_use:N \c@iRow } }
5067 }

```

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.

```

5068 \AtBeginDocument
5069 {

```

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}`).

```

5070 \cs_new_protected:Npx \@@_hdottedline:n #1
5071 {
5072 \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
5073 \bool_set_true:N \exp_not:N \l_@@_final_open_bool
5074 \c_@@_pgfortikzpicture_tl
5075 \@@_hdottedline_i:n { #1 }
5076 \c_@@_endpgfortikzpicture_tl
5077 }
5078 }

```

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

```

5079 \cs_new_protected:Npn \@@_hdottedline_i:n #1
5080 {
5081 \pgfrememberpicturepositiononpagetrue
5082 \@@_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).

```

5083 \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
5084 \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
5085 \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

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

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

```
\begin{bNiceMatrix}
```

```
1 & 2 & 3 & 4 \\\
```

```
\hline
```

```
1 & 2 & 3 & 4 \\\
```

```
\hdottedline
```

```
1 & 2 & 3 & 4
```

```
\end{bNiceMatrix}
```

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

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

```
\begin{bNiceMatrix}[margin]
```

```
1 & 2 & 3 & 4 \\\
```

```
\hline
```

```
1 & 2 & 3 & 4 \\\
```

```
\hdottedline
```

```
1 & 2 & 3 & 4
```

```
\end{bNiceMatrix}
```

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

```

5086 \@@_qpoint:n { col - 1 }
5087 \dim_set:Nn \l_@@_x_initial_dim
5088 {
5089 \pgf@x +

```

We do a reduction by `\arraycolsep` for the environments with delimiters (and not for the other).

```

5090 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
5091 - \l_@@_left_margin_dim
5092 }
5093 \@@_qpoint:n { col - \@@_succ:n \c@jCol }
5094 \dim_set:Nn \l_@@_x_final_dim
5095 {
5096 \pgf@x -
5097 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
5098 + \l_@@_right_margin_dim
5099 }

```

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.

```

5100 \tl_if_eq:NnF \g_@@_left_delim_tl (
5101 { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
5102 \tl_if_eq:NnF \g_@@_right_delim_tl )
5103 { \dim_gsub:Nn \l_@@_x_final_dim { 0.5 \l_@@_inter_dots_dim } }

```

Up to 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`.

```

5104 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
5105 \@@_draw_line:
5106 }

```

## Vertical dotted lines

```

5107 \cs_new_protected:Npn \@@_vdottedline:n #1
5108 {
5109 \bool_set_true:N \l_@@_initial_open_bool
5110 \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”.

```

5111 \bool_if:NTF \c_@@_tikz_loaded_bool
5112 {
5113 \tikzpicture
5114 \@@_vdottedline_i:n { #1 }
5115 \endtikzpicture
5116 }
5117 {
5118 \pgfpicture
5119 \@@_vdottedline_i:n { #1 }
5120 \endpgfpicture
5121 }
5122 }

```

```

5123 \cs_new_protected:Npn \@@_vdottedline_i:n #1
5124 {

```

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.

```

5125 \CT@arc@
5126 \pgfrememberpicturepositiononpagetrue
5127 \@@_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).

```
5128 \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
5129 \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
5130 \@@_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.

```
5131 \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
5132 \@@_qpoint:n { row - \@@_succ:n \c@iRow }
5133 \dim_set:Nn \l_@@_y_final_dim { \pgf@y + 0.5 \l_@@_inter_dots_dim }
```

Up to 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`.

```
5134 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
5135 \@@_draw_line:
5136 }
```

## 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.

```
5137 \bool_new:N \l_@@_block_auto_columns_width_bool
```

Up to now, there is only one option available for the environment {NiceMatrixBlock}.

```
5138 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
5139 {
5140   auto-columns-width .code:n =
5141   {
5142     \bool_set_true:N \l_@@_block_auto_columns_width_bool
5143     \dim_gzero_new:N \g_@@_max_cell_width_dim
5144     \bool_set_true:N \l_@@_auto_columns_width_bool
5145   }
5146 }

5147 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
5148 {
5149   \int_gincr:N \g_@@_NiceMatrixBlock_int
5150   \dim_zero:N \l_@@_columns_width_dim
5151   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
5152   \bool_if:NT \l_@@_block_auto_columns_width_bool
5153   {
5154     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
5155     {
5156       \exp_args:NNc \dim_set:Nn \l_@@_columns_width_dim
5157       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
5158     }
5159   }
5160 }
```

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`).

```
5161 {
5162   \bool_if:NT \l_@@_block_auto_columns_width_bool
5163   {
5164     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
5165     \iow_shipout:Nx \@mainaux
```

```

5166     {
5167         \cs_gset:cpn
5168         { @@ _ 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.

```

5169         { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
5170     }
5171     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
5172 }
5173 }

```

## The extra nodes

First, two variants of the functions `\dim_min:nn` and `\dim_max:nn`.

```

5174 \cs_generate_variant:Nn \dim_min:nn { v n }
5175 \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 and that construction uses the standard medium nodes).

```

5176 \cs_new_protected:Npn \@@_create_extra_nodes:
5177 {
5178     \bool_if:nTF \l_@@_medium_nodes_bool
5179     {
5180         \bool_if:nTF \l_@@_large_nodes_bool
5181         \@@_create_medium_and_large_nodes:
5182         \@@_create_medium_nodes:
5183     }
5184     { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
5185 }

```

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`.

```

5186 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
5187 {
5188     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
5189     {
5190         \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
5191         \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
5192         \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
5193         \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
5194     }
5195     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
5196     {

```

```

5197 \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
5198 \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
5199 \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
5200 \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
5201 }

```

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

```

5202 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
5203 {
5204   \int_step_variable:nnNn
5205   \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.

```

5206   {
5207     \cs_if_exist:cT
5208     { 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.

```

5209   {
5210     \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
5211     \dim_set:cn { l_@@_row_\@@_i: _min_dim }
5212     { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
5213     \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
5214     {
5215       \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
5216       { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
5217     }

```

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.

```

5218     \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
5219     \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
5220     { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
5221     \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
5222     {
5223       \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
5224       { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
5225     }
5226   }
5227 }
5228 }

```

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

```

5229 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
5230 {
5231   \dim_compare:nNnT
5232   { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
5233   {
5234     \@@_qpoint:n { row - \@@_i: - base }
5235     \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
5236     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
5237   }
5238 }
5239 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
5240 {
5241   \dim_compare:nNnT
5242   { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
5243   {
5244     \@@_qpoint:n { col - \@@_j: }
5245     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@x
5246     \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@x
5247   }

```

```

5248     }
5249 }

```

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

```

5250 \cs_new_protected:Npn \@@_create_medium_nodes:
5251 {
5252   \pgfpicture
5253     \pgfrememberpicturepositiononpagetrue
5254     \pgf@relevantforpicturesizefalse
5255     \@@_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”.

```

5256     \tl_set:Nn \l_@@_suffix_tl { -medium }
5257     \@@_create_nodes:
5258   \endpgfpicture
5259 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones<sup>66</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:`.

```

5260 \cs_new_protected:Npn \@@_create_large_nodes:
5261 {
5262   \pgfpicture
5263     \pgfrememberpicturepositiononpagetrue
5264     \pgf@relevantforpicturesizefalse
5265     \@@_computations_for_medium_nodes:
5266     \@@_computations_for_large_nodes:
5267     \tl_set:Nn \l_@@_suffix_tl { -large }
5268     \@@_create_nodes:
5269   \endpgfpicture
5270 }
5271 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
5272 {
5273   \pgfpicture
5274     \pgfrememberpicturepositiononpagetrue
5275     \pgf@relevantforpicturesizefalse
5276     \@@_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”.

```

5277     \tl_set:Nn \l_@@_suffix_tl { -medium }
5278     \@@_create_nodes:
5279     \@@_computations_for_large_nodes:
5280     \tl_set:Nn \l_@@_suffix_tl { -large }
5281     \@@_create_nodes:
5282   \endpgfpicture
5283 }

```

For “large nodes”, the exterior rows and columns don’t interfere. That’s why the loop over the columns will start at 1 and stop at `\c@jCol` (and not `\g_@@_col_total_int`). Idem for the rows.

```

5284 \cs_new_protected:Npn \@@_computations_for_large_nodes:
5285 {
5286   \int_set:Nn \l_@@_first_row_int 1
5287   \int_set:Nn \l_@@_first_col_int 1

```

---

<sup>66</sup>If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

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`.

```

5288 \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
5289 {
5290   \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
5291   {
5292     (
5293       \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
5294       \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
5295     )
5296     / 2
5297   }
5298   \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
5299   { l_@@_row_\@@_i: _ min_dim }
5300 }
5301 \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
5302 {
5303   \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
5304   {
5305     (
5306       \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
5307       \dim_use:c
5308       { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
5309     )
5310     / 2
5311   }
5312   \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
5313   { l_@@_column _ \@@_j: _ max _ dim }
5314 }

```

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

```

5315 \dim_sub:cn
5316 { l_@@_column _ 1 _ min _ dim }
5317 \l_@@_left_margin_dim
5318 \dim_add:cn
5319 { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
5320 \l_@@_right_margin_dim
5321 }

```

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).

```

5322 \cs_new_protected:Npn \@@_create_nodes:
5323 {
5324   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
5325   {
5326     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
5327     {

```

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

```

5328 \@@_pgf_rect_node:nnnnn
5329 { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
5330 { \dim_use:c { l_@@_column _ \@@_j: _ min_dim } }
5331 { \dim_use:c { l_@@_row _ \@@_i: _ min_dim } }
5332 { \dim_use:c { l_@@_column _ \@@_j: _ max_dim } }
5333 { \dim_use:c { l_@@_row _ \@@_i: _ max_dim } }
5334 \str_if_empty:NF \l_@@_name_str
5335 {
5336   \pgfnodealias
5337   { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }

```



```

5338         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
5339     }
5340 }
5341 }

```

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  $n$ .

```

5342 \seq_mapthread_function:NNN
5343   \g_@@_multicolumn_cells_seq
5344   \g_@@_multicolumn_sizes_seq
5345   \@@_node_for_multicolumn:nn
5346 }

```

```

5347 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
5348 {
5349   \cs_set_nopar:Npn \@@_i: { #1 }
5350   \cs_set_nopar:Npn \@@_j: { #2 }
5351 }

```

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”).

```

5352 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
5353 {
5354   \@@_extract_coords_values: #1 \q_stop
5355   \@@_pgf_rect_node:nnnnn
5356     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
5357     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }
5358     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } }
5359     { \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
5360     { \dim_use:c { l_@@_row _ \@@_i: _ max _ dim } }
5361   \str_if_empty:NF \l_@@_name_str
5362   {
5363     \pgfnodealias
5364       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
5365       { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
5366   }
5367 }

```

## 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.

```

5368 \keys_define:nn { NiceMatrix / Block / FirstPass }
5369 {
5370   l .code:n = \str_set:Nn \l_@@_hpos_block_str l ,
5371   l .value_forbidden:n = true ,
5372   r .code:n = \str_set:Nn \l_@@_hpos_block_str r ,
5373   r .value_forbidden:n = true ,
5374   c .code:n = \str_set:Nn \l_@@_hpos_block_str c ,
5375   c .value_forbidden:n = true ,
5376   L .code:n = \str_set:Nn \l_@@_hpos_block_str l ,
5377   L .value_forbidden:n = true ,
5378   R .code:n = \str_set:Nn \l_@@_hpos_block_str r ,
5379   R .value_forbidden:n = true ,
5380   C .code:n = \str_set:Nn \l_@@_hpos_block_str c ,

```

```

5381 C .value_forbidden:n = true ,
5382 t .code:n = \str_set:Nn \l_@@_vpos_of_block_tl t ,
5383 t .value_forbidden:n = true ,
5384 b .code:n = \str_set:Nn \l_@@_vpos_of_block_tl b ,
5385 b .value_forbidden:n = true ,
5386 color .tl_set:N = \l_@@_color_tl ,
5387 color .value_required:n = true
5388 }

```

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 `>`. It's mandatory to use an expandable command.

```

5389 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
5390 {

```

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).

```

5391 \peek_remove_spaces:n
5392 {
5393   \tl_if_blank:nTF { #2 }
5394   { \@@_Block_i 1-1 \q_stop }
5395   { \@@_Block_i #2 \q_stop }
5396   { #1 } { #3 } { #4 }
5397 }
5398 }

```

With the following construction, we extract the values of  $i$  and  $j$  in the first mandatory argument of the command.

```

5399 \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.

```

5400 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
5401 {

```

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).

```

5402 \bool_lazy_or:nnTF
5403 { \tl_if_blank_p:n { #1 } }
5404 { \str_if_eq_p:nn { #1 } { * } }
5405 { \int_set:Nn \l_tmpa_int { 100 } }
5406 { \int_set:Nn \l_tmpa_int { #1 } }
5407 \bool_lazy_or:nnTF
5408 { \tl_if_blank_p:n { #2 } }
5409 { \str_if_eq_p:nn { #2 } { * } }
5410 { \int_set:Nn \l_tmpb_int { 100 } }
5411 { \int_set:Nn \l_tmpb_int { #2 } }

```

If the block is mono-column.

```

5412 \int_compare:nNnTF \l_tmpb_int = 1
5413 {
5414   \str_if_empty:NTF \l_@@_hpos_cell_str
5415   { \str_set:Nn \l_@@_hpos_block_str c }
5416   { \str_set_eq:NN \l_@@_hpos_block_str \l_@@_hpos_cell_str }
5417 }
5418 { \str_set:Nn \l_@@_hpos_block_str c }

```

The value of `\l_@@_hpos_block_str` may be modified by the keys of the command `\Block` that we will analyze now.

```

5419 \keys_set_known:nn { NiceMatrix / Block / FirstPass } { #3 }
5420 \tl_set:Nx \l_tmpa_tl
5421 {
5422   { \int_use:N \c@iRow }
5423   { \int_use:N \c@jCol }
5424   { \int_eval:n { \c@iRow + \l_tmpa_int - 1 } }
5425   { \int_eval:n { \c@jCol + \l_tmpb_int - 1 } }
5426 }

```

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).

```

5427 \bool_if:nTF
5428 {
5429   (
5430     \int_compare_p:nNn { \l_tmpa_int } = 1
5431     ||
5432     \int_compare_p:nNn { \l_tmpb_int } = 1
5433   )
5434   && ! \tl_if_empty_p:n { #5 }

```

For the blocks mono-column, we will compose right now in a box in order to compute its width and take that width into account for the width of the column. However, if the column is a X column, we should not do that since the width is determined by another way. This should be the same for the p, m and b columns and we should modify that point. However, for the X column, it’s imperative. Otherwise, the process for the determination of the widths of the columns will be wrong.

```

5435   && ! \l_@@_X_column_bool
5436 }
5437 { \exp_args:Nxx \@@_Block_iv:nnnnn }
5438 { \exp_args:Nxx \@@_Block_v:nnnnn }
5439 { \l_tmpa_int } { \l_tmpb_int } { #3 } { #4 } { #5 }
5440 }

```

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).

```

5441 \cs_new_protected:Npn \@@_Block_iv:nnnnn #1 #2 #3 #4 #5
5442 {
5443   \int_gincr:N \g_@@_block_box_int
5444   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
5445   {
5446     \tl_gput_right:Nx \g_@@_internal_code_after_tl
5447     {
5448       \@@_actually_diagbox:nnnnnn
5449       { \int_use:N \c@iRow }
5450       { \int_use:N \c@jCol }
5451       { \int_eval:n { \c@iRow + #1 - 1 } }
5452       { \int_eval:n { \c@jCol + #2 - 1 } }
5453       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
5454     }
5455   }
5456   \box_gclear_new:c
5457   { g_@@_block_box_int _ \int_use:N \g_@@_block_box_int _ box }

```

```

5458 \hbox_gset:cn
5459 { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
5460 {

```

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:` (in order to use `\color_ensure_current:` safely, you should load `l3backend` before the `\documentclass` with `\RequirePackage{expl3}`).

```

5461 \tl_if_empty:NTF \l_@@_color_tl
5462 { \int_compare:nNnT { #2 } = 1 \set@color }
5463 { \color { \l_@@_color_tl } }

```

If the block is mono-row, we use `\g_@@_row_style_tl` even if it has yet been used in the beginning of the cell where the command `\Block` has been issued because we want to be able to take into account a potential instruction of color of the font in `\g_@@_row_style_tl`.

```

5464 \int_compare:nNnT { #1 } = 1 \g_@@_row_style_tl
5465 \group_begin:
5466 \cs_set:Npn \arraystretch { 1 }
5467 \dim_set_eq:NN \extrarowheight \c_zero_dim
5468 #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).

```

5469 \bool_if:NT \g_@@_rotate_bool { \str_set:Nn \l_@@_hpos_block_str c }
5470 \bool_if:NTF \l_@@_NiceTabular_bool
5471 {
5472   \bool_lazy_and:nnTF
5473   { \int_compare_p:nNn { #2 } = 1 }
5474   { \dim_compare_p:n { \l_@@_col_width_dim >= \c_zero_dim } }

```

When the block is mono-column in a column with a fixed width (eg `p{3cm}`).

```

5475 {
5476   \begin { minipage } [ \l_@@_vpos_of_block_tl ]
5477   { \l_@@_col_width_dim }
5478   \str_case:Nn \l_@@_hpos_block_str
5479   {
5480     c \centering
5481     r \raggedleft
5482     l \raggedright
5483   }
5484   #5
5485   \end { minipage }
5486 }
5487 {
5488   \use:x
5489   {
5490     \exp_not:N \begin { tabular } [ \l_@@_vpos_of_block_tl ]
5491     { @ { } \l_@@_hpos_block_str @ { } }
5492   }
5493   #5
5494   \end { tabular }
5495 }
5496 }
5497 {
5498   \c_math_toggle_token
5499   \use:x
5500   {
5501     \exp_not:N \begin { array } [ \l_@@_vpos_of_block_tl ]
5502     { @ { } \l_@@_hpos_block_str @ { } }
5503   }
5504   #5

```

```

5505         \end { array }
5506         \c_math_toggle_token
5507     }
5508     \group_end:
5509 }
5510 \bool_if:NT \g_@@_rotate_bool
5511 {
5512     \box_grotate:cn
5513     { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
5514     { 90 }
5515     \bool_gset_false:N \g_@@_rotate_bool
5516 }

```

If we are in a mono-column block, we take into account the width of that block for the width of the column.

```

5517 \int_compare:nNnT { #2 } = 1
5518 {
5519     \dim_gset:Nn \g_@@_blocks_wd_dim
5520     {
5521         \dim_max:nn
5522         \g_@@_blocks_wd_dim
5523         {
5524             \box_wd:c
5525             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
5526         }
5527     }
5528 }

```

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.

```

5529 \int_compare:nNnT { #1 } = 1
5530 {
5531     \dim_gset:Nn \g_@@_blocks_ht_dim
5532     {
5533         \dim_max:nn
5534         \g_@@_blocks_ht_dim
5535         {
5536             \box_ht:c
5537             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
5538         }
5539     }
5540     \dim_gset:Nn \g_@@_blocks_dp_dim
5541     {
5542         \dim_max:nn
5543         \g_@@_blocks_dp_dim
5544         {
5545             \box_dp:c
5546             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
5547         }
5548     }
5549 }
5550 \seq_gput_right:Nx \g_@@_blocks_seq
5551 {
5552     \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_block_str`. 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_block_str`, which is fixed by the type of current column.

```

5553 { \exp_not:n { #3 } , \l_@@_hpos_block_str }
5554 {
5555     \box_use_drop:c
5556     { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
5557 }

```

```

5558 }
5559 }

```

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.

```

5560 \cs_new_protected:Npn \@@_Block_v:nnnnn #1 #2 #3 #4 #5
5561 {
5562   \seq_gput_right:Nx \g_@@_blocks_seq
5563   {
5564     \l_tmpa_tl
5565     { \exp_not:n { #3 } }
5566     \exp_not:n
5567     {
5568       {
5569         \bool_if:NTF \l_@@_NiceTabular_bool
5570         {
5571           \group_begin:
5572           \cs_set:Npn \arraystretch { 1 }
5573           \dim_set_eq:NN \extrarowheight \c_zero_dim
5574           #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).

```

5575       \bool_if:NT \g_@@_rotate_bool
5576       { \str_set:Nn \l_@@_hpos_block_str c }
5577       \use:x
5578       {
5579         \exp_not:N \begin { tabular } [ \l_@@_vpos_of_block_tl ]
5580         { @ { } \l_@@_hpos_block_str @ { } }
5581       }
5582       #5
5583       \end { tabular }
5584       \group_end:
5585     }
5586     {
5587       \group_begin:
5588       \cs_set:Npn \arraystretch { 1 }
5589       \dim_set_eq:NN \extrarowheight \c_zero_dim
5590       #4
5591       \bool_if:NT \g_@@_rotate_bool
5592       { \str_set:Nn \l_@@_hpos_block_str c }
5593       \c_math_toggle_token
5594       \use:x
5595       {
5596         \exp_not:N \begin { array } [ \l_@@_vpos_of_block_tl ]
5597         { @ { } \l_@@_hpos_block_str @ { } }
5598       }
5599       #5
5600       \end { array }
5601       \c_math_toggle_token
5602       \group_end:
5603     }
5604   }
5605 }
5606 }
5607 }

```

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).

```

5608 \keys_define:nn { NiceMatrix / Block / SecondPass }
5609 {
5610     tikz .code:n =
5611         \bool_if:NTF \c_@@_tikz_loaded_bool
5612         { \seq_put_right:Nn \l_@@_tikz_seq { { #1 } } }
5613         { \@@_error:n { tikz-key-without-tikz } } ,
5614     tikz .value_required:n = true ,
5615     fill .tl_set:N = \l_@@_fill_tl ,
5616     fill .value_required:n = true ,
5617     draw .tl_set:N = \l_@@_draw_tl ,
5618     draw .default:n = default ,
5619     rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
5620     rounded-corners .default:n = 4 pt ,
5621     color .code:n = \color { #1 } \tl_set:Nn \l_@@_draw_tl { #1 } ,
5622     color .value_required:n = true ,
5623     borders .clist_set:N = \l_@@_borders_clist ,
5624     borders .value_required:n = true ,
5625     hvlines .bool_set:N = \l_@@_hvlines_block_bool ,
5626     hvlines .default:n = true ,
5627     line-width .dim_set:N = \l_@@_line_width_dim ,
5628     line-width .value_required:n = true ,
5629     l .code:n = \str_set:Nn \l_@@_hpos_block_str l ,
5630     l .value_forbidden:n = true ,
5631     r .code:n = \str_set:Nn \l_@@_hpos_block_str r ,
5632     r .value_forbidden:n = true ,
5633     c .code:n = \str_set:Nn \l_@@_hpos_block_str c ,
5634     c .value_forbidden:n = true ,
5635     L .code:n = \str_set:Nn \l_@@_hpos_block_str l
5636         \bool_set_true:N \l_@@_hpos_of_block_cap_bool ,
5637     L .value_forbidden:n = true ,
5638     R .code:n = \str_set:Nn \l_@@_hpos_block_str r
5639         \bool_set_true:N \l_@@_hpos_of_block_cap_bool ,
5640     R .value_forbidden:n = true ,
5641     C .code:n = \str_set:Nn \l_@@_hpos_block_str c
5642         \bool_set_true:N \l_@@_hpos_of_block_cap_bool ,
5643     C .value_forbidden:n = true ,
5644     t .code:n = \str_set:Nn \l_@@_vpos_of_block_tl t ,
5645     t .value_forbidden:n = true ,
5646     b .code:n = \str_set:Nn \l_@@_vpos_of_block_tl b ,
5647     b .value_forbidden:n = true ,
5648     unknown .code:n = \@@_error:n { Unknown-key-for-Block }
5649 }

```

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.

```

5650 \cs_new_protected:Npn \@@_draw_blocks:
5651 {
5652     \cs_set_eq:NN \ialign \@@_old_ialign:
5653     \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iv:nnnnnn #1 }
5654 }
5655 \cs_new_protected:Npn \@@_Block_iv:nnnnnn #1 #2 #3 #4 #5 #6
5656 {

```

The integer `\l_@@_last_row_int` will be the last row of the block and `\l_@@_last_col_int` its last column.

```

5657 \int_zero_new:N \l_@@_last_row_int
5658 \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.

```

5659 \int_compare:nNnTF { #3 } > { 99 }
5660 { \int_set_eq:NN \l_@@_last_row_int \c@iRow }
5661 { \int_set:Nn \l_@@_last_row_int { #3 } }
5662 \int_compare:nNnTF { #4 } > { 99 }
5663 { \int_set_eq:NN \l_@@_last_col_int \c@jCol }
5664 { \int_set:Nn \l_@@_last_col_int { #4 } }
5665 \int_compare:nNnTF \l_@@_last_col_int > \g_@@_col_total_int
5666 {
5667   \int_compare:nTF
5668   { \l_@@_last_col_int <= \g_@@_static_num_of_col_int }
5669   {
5670     \msg_error:nnnn { nicematrix } { Block-too~large~2 } { #1 } { #2 }
5671     \@@_msg_redirect_name:nn { Block-too~large~2 } { none }
5672     \group_begin:
5673     \globaldefs = 1
5674     \@@_msg_redirect_name:nn { columns-not~used } { none }
5675     \group_end:
5676   }
5677   { \msg_error:nnnn { nicematrix } { Block-too~large~1 } { #1 } { #2 } }
5678 }
5679 {
5680   \int_compare:nNnTF \l_@@_last_row_int > \g_@@_row_total_int
5681   { \msg_error:nnnn { nicematrix } { Block-too~large~1 } { #1 } { #2 } }
5682   { \@@_Block_v:nnnnnn { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } }
5683 }
5684 }
5685 \cs_new_protected:Npn \@@_Block_v:nnnnnn #1 #2 #3 #4 #5 #6
5686 {

```

The group is for the keys.

```

5687 \group_begin:
5688 \keys_set:nn { NiceMatrix / Block / SecondPass } { #5 }
5689 \bool_if:NTF \l_@@_hvlines_block_bool
5690 {
5691   \tl_gput_right:Nx \g_nicematrix_code_after_tl
5692   {
5693     \@@_hvlines_block:nnn
5694     { \exp_not:n { #5 } }
5695     { #1 - #2 }
5696     { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5697   }
5698 }
5699 {

```

The sequence of the positions of the blocks (excepted the blocks with the key `hvlines`) will be used when drawing the rules (in fact, there is also the `\multicolumn` and the `\diagbox` in that sequence).

```

5700 \seq_gput_left:Nn \g_@@_pos_of_blocks_seq
5701 { { #1 } { #2 } { #3 } { #4 } }
5702 }
5703 \tl_if_empty:NF \l_@@_draw_tl
5704 {
5705   \tl_gput_right:Nx \g_nicematrix_code_after_tl
5706   {
5707     \@@_stroke_block:nnn
5708     { \exp_not:n { #5 } }
5709     { #1 - #2 }

```



```

5710         { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5711     }
5712     \seq_gput_right:Nn \g_@@_pos_of_stroken_blocks_seq
5713     { { #1 } { #2 } { #3 } { #4 } }
5714 }
5715 \clist_if_empty:NF \l_@@_borders_clist
5716 {
5717     \tl_gput_right:Nx \g_nicematrix_code_after_tl
5718     {
5719         \@@_stroke_borders_block:nnn
5720         { \exp_not:n { #5 } }
5721         { #1 - #2 }
5722         { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5723     }
5724 }
5725 \tl_if_empty:NF \l_@@_fill_tl
5726 {

```

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`.

```

5727     \exp_last_unbraced:NV \@@_extract_brackets \l_@@_fill_tl \q_stop
5728     \tl_gput_right:Nx \g_nicematrix_code_before_tl
5729     {
5730         \exp_not:N \roundedrectanglecolor
5731         [ \l_tmpa_tl ]
5732         { \exp_not:V \l_tmpb_tl }
5733         { #1 - #2 }
5734         { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5735         { \dim_use:N \l_@@_rounded_corners_dim }
5736     }
5737 }
5738 \seq_if_empty:NF \l_@@_tikz_seq
5739 {
5740     \tl_gput_right:Nx \g_nicematrix_code_before_tl
5741     {
5742         \@@_block_tikz:nnnnn
5743         { #1 }
5744         { #2 }
5745         { \int_use:N \l_@@_last_row_int }
5746         { \int_use:N \l_@@_last_col_int }
5747         { \seq_use:Nn \l_@@_tikz_seq { , } }
5748     }
5749 }
5750 \cs_set_protected_nopar:Npn \diagbox ##1 ##2
5751 {
5752     \tl_gput_right:Nx \g_@@_internal_code_after_tl
5753     {
5754         \@@_actually_diagbox:nnnnnn
5755         { #1 }
5756         { #2 }
5757         { \int_use:N \l_@@_last_row_int }
5758         { \int_use:N \l_@@_last_col_int }
5759         { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
5760     }
5761 }
5762 \hbox_set:Nn \l_@@_cell_box { \set@color #6 }
5763 \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`.

```
\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.

```
5764 \pgfpicture
5765 \pgfrememberpicturepositiononpagetrue
5766 \pgf@relevantforpicturesizefalse
5767 \@@_qpoint:n { row - #1 }
5768 \dim_set_eq:NN \l_tmpa_dim \pgf@y
5769 \@@_qpoint:n { col - #2 }
5770 \dim_set_eq:NN \l_tmpb_dim \pgf@x
5771 \@@_qpoint:n { row - \@@_succ:n { \l_@@_last_row_int } }
5772 \dim_set_eq:NN \l_tmppc_dim \pgf@y
5773 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
5774 \dim_set_eq:NN \l_tmppd_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.

```
5775 \begin { pgfscope }
5776 \@@_pgf_rect_node:nnnnn
5777 { \@@_env: - #1 - #2 - block }
5778 \l_tmpb_dim \l_tmpa_dim \l_tmppd_dim \l_tmppc_dim
5779 \end { pgfscope }
```

Now, we create the “short node” which, in general, will be used to put the label (that is to say the content of the node). However, if one the keys `L`, `C` or `R` is used (that information is provided by the boolean `\l_@@_hpos_of_block_cap_bool`), we don't need to create that node since the normal node is used to put the label.

```
5780 \bool_if:NF \l_@@_hpos_of_block_cap_bool
5781 {
5782 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
```

The short node is constructed by taking into account the *contents* of the columns involved in at least one cell of the block. That's why we have to do a loop over the rows of the array.

```
5783 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5784 {
```

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.

```
5785 \cs_if_exist:cT
5786 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
5787 {
5788 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
5789 {
5790 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
5791 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
```

```

5792     }
5793   }
5794 }

```

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.

```

5795   \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
5796   {
5797     \@@_qpoint:n { col - #2 }
5798     \dim_set_eq:NN \l_tmpb_dim \pgf@x
5799   }
5800   \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
5801   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5802   {
5803     \cs_if_exist:cT
5804     { \pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
5805     {
5806       \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
5807       {
5808         \pgfpointanchor
5809         { \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
5810         { east }
5811         \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
5812       }
5813     }
5814   }
5815   \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
5816   {
5817     \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
5818     \dim_set_eq:NN \l_tmpd_dim \pgf@x
5819   }
5820   \@@_pgf_rect_node:nnnnn
5821   { \@@_env: - #1 - #2 - block - short }
5822   \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
5823 }

```

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

```

5824   \bool_if:NT \l_@@_medium_nodes_bool
5825   {
5826     \@@_pgf_rect_node:nnn
5827     { \@@_env: - #1 - #2 - block - medium }
5828     { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
5829     {
5830       \pgfpointanchor
5831       { \@@_env:
5832         - \int_use:N \l_@@_last_row_int
5833         - \int_use:N \l_@@_last_col_int - medium
5834       }
5835       { south-east }
5836     }
5837   }

```

Now, we will put the label of the block beginning with the case of a `\Block` of one row.

```

5838   \int_compare:nNnTF { #1 } = { #3 }
5839   {

```

We take into account the case of a block of one row in the “first row” or the “last row”.

```

5840     \int_compare:nNnTF { #1 } = 0
5841     { \l_@@_code_for_first_row_tl }
5842     {
5843       \int_compare:nNnT { #1 } = \l_@@_last_row_int
5844       \l_@@_code_for_last_row_tl
5845     }

```

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`.

```
5846 \pgfextracty \l_tmpa_dim { \l_@@_qpoint:n { row - #1 - base } }
```

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

```
5847 \pgfpointanchor
5848 {
5849   \l_@@_env: - #1 - #2 - block
5850   \bool_if:NF \l_@@_hpos_of_block_cap_bool { - short }
5851 }
5852 {
5853   \str_case:Vn \l_@@_hpos_block_str
5854   {
5855     c { center }
5856     l { west }
5857     r { east }
5858   }
5859 }
```

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

```
5860 \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
5861 \pgfset { inner~sep = \c_zero_dim }
5862 \pgfnode
5863 { rectangle }
5864 {
5865   \str_case:Vn \l_@@_hpos_block_str
5866   {
5867     c { base }
5868     l { base~west }
5869     r { base~east }
5870   }
5871 }
5872 { \box_use_drop:N \l_@@_cell_box } { } { }
5873 }
```

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`).

```
5874 {
```

If we are in the first column, we must put the block as if it was with the key `r`.

```
5875 \int_compare:nNnT { #2 } = 0
5876 { \str_set:Nn \l_@@_hpos_block_str r }
5877 \bool_if:nT \g_@@_last_col_found_bool
5878 {
5879   \int_compare:nNnT { #2 } = \g_@@_col_total_int
5880   { \str_set:Nn \l_@@_hpos_block_str l }
5881 }
5882 \pgftransformshift
5883 {
5884   \pgfpointanchor
5885   {
5886     \l_@@_env: - #1 - #2 - block
5887     \bool_if:NF \l_@@_hpos_of_block_cap_bool { - short }
5888   }
5889   {
5890     \str_case:Vn \l_@@_hpos_block_str
5891     {
5892       c { center }
5893       l { west }
5894       r { east }
5895     }
5896   }
5897 }
```

```

5898     \pgfset { inner~sep = \c_zero_dim }
5899     \pgfnode
5900     { rectangle }
5901     {
5902         \str_case:Vn \l_@@_hpos_block_str
5903         {
5904             c { center }
5905             l { west }
5906             r { east }
5907         }
5908     }
5909     { \box_use_drop:N \l_@@_cell_box } { } { }
5910 }
5911 \endpgfpicture
5912 \group_end:
5913 }

5914 \NewDocumentCommand \@@_extract_brackets { 0 { } }
5915 {
5916     \tl_set:Nn \l_tmpa_tl { #1 }
5917     \@@_store_in_tmpb_tl
5918 }
5919 \cs_new_protected:Npn \@@_store_in_tmpb_tl #1 \q_stop
5920 { \tl_set:Nn \l_tmpb_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).

```

5921 \cs_new_protected:Npn \@@_stroke_block:nnn #1 #2 #3
5922 {
5923     \group_begin:
5924     \tl_clear:N \l_@@_draw_tl
5925     \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
5926     \keys_set_known:nn { NiceMatrix / BlockStroke } { #1 }
5927     \pgfpicture
5928     \pgfrememberpicturepositiononpagetrue
5929     \pgf@relevantforpicturesizefalse
5930     \tl_if_empty:NF \l_@@_draw_tl
5931     {

```

If the user has used the key `color` of the command `\Block` without value, the color fixed by `\arrayrulecolor` is used.

```

5932     \str_if_eq:VnTF \l_@@_draw_tl { default }
5933     { \CT@arc@ }
5934     { \exp_args:NV \pgfsetstrokecolor \l_@@_draw_tl }
5935 }
5936 \pgfsetcornersarced
5937 {
5938     \pgfpoint
5939     { \dim_use:N \l_@@_rounded_corners_dim }
5940     { \dim_use:N \l_@@_rounded_corners_dim }
5941 }
5942 \@@_cut_on_hyphen:w #2 \q_stop
5943 \bool_lazy_and:nnT
5944 { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
5945 { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
5946 {
5947     \@@_qpoint:n { row - \l_tmpa_tl }
5948     \dim_set:Nn \l_tmpb_dim { \pgf@y }
5949     \@@_qpoint:n { col - \l_tmpb_tl }
5950     \dim_set:Nn \l_tmpc_dim { \pgf@x }
5951     \@@_cut_on_hyphen:w #3 \q_stop

```

```

5952 \int_compare:nNnT \l_tmpa_tl > \c@iRow
5953 { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
5954 \int_compare:nNnT \l_tmpb_tl > \c@jCol
5955 { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
5956 \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
5957 \dim_set:Nn \l_tmpa_dim { \pgf@y }
5958 \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
5959 \dim_set:Nn \l_tmpd_dim { \pgf@x }
5960 \pgfpathrectanglecorners
5961 { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
5962 { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
5963 \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }

```

We can't use `\pgfusepathqstroke` because of the key `rounded-corners`.

```

5964 \pgfusepath { stroke }
5965 }
5966 \endpgfpicture
5967 \group_end:
5968 }

```

Here is the set of keys for the command `\@@_stroke_block:nnn`.

```

5969 \keys_define:nn { NiceMatrix / BlockStroke }
5970 {
5971   color .tl_set:N = \l_@@_draw_tl ,
5972   draw .tl_set:N = \l_@@_draw_tl ,
5973   draw .default:n = default ,
5974   line-width .dim_set:N = \l_@@_line_width_dim ,
5975   rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
5976   rounded-corners .default:n = 4 pt
5977 }

```

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).

```

5978 \cs_new_protected:Npn \@@_hvlines_block:nnn #1 #2 #3
5979 {
5980   \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
5981   \keys_set_known:nn { NiceMatrix / BlockBorders } { #1 }
5982   \@@_cut_on_hyphen:w #2 \q_stop
5983   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
5984   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
5985   \@@_cut_on_hyphen:w #3 \q_stop
5986   \tl_set:Nx \l_tmpa_tl { \int_eval:n { \l_tmpa_tl + 1 } }
5987   \tl_set:Nx \l_tmpb_tl { \int_eval:n { \l_tmpb_tl + 1 } }
5988   \int_step_inline:nnn \l_tmpd_tl \l_tmpb_tl
5989   {
5990     \use:x
5991     { \@@_vline:nnnn { ##1 } 1 { \l_tmpc_tl } { \@@_pred:n \l_tmpa_tl } }
5992   }
5993   \int_step_inline:nnn \l_tmpc_tl \l_tmpa_tl
5994   {
5995     \use:x
5996     { \@@_hline:nnnn { ##1 } 1 { \l_tmpd_tl } { \@@_pred:n \l_tmpb_tl } }
5997   }
5998 }

```

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).

```

5999 \cs_new_protected:Npn \@@_stroke_borders_block:nnn #1 #2 #3
6000 {
6001   \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
6002   \keys_set_known:nn { NiceMatrix / BlockBorders } { #1 }

```

```

6003 \dim_compare:nNnTF \l_@@_rounded_corners_dim > \c_zero_dim
6004 { \@@_error:n { borders~forbidden } }
6005 {
6006   \clist_map_inline:Nn \l_@@_borders_clist
6007   {
6008     \clist_if_in:nNF { top , bottom , left , right } { ##1 }
6009     { \@@_error:nn { bad-border } { ##1 } }
6010   }
6011   \@@_cut_on_hyphen:w #2 \q_stop
6012   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
6013   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
6014   \@@_cut_on_hyphen:w #3 \q_stop
6015   \tl_set:Nx \l_tmpa_tl { \int_eval:n { \l_tmpa_tl + 1 } }
6016   \tl_set:Nx \l_tmpb_tl { \int_eval:n { \l_tmpb_tl + 1 } }
6017   \pgfpicture
6018   \pgfrememberpicturepositiononpagetrue
6019   \pgf@relevantforpicturesizefalse
6020   \CT@arc@
6021   \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }
6022   \clist_if_in:NnT \l_@@_borders_clist { right }
6023   { \@@_stroke_vertical:n \l_tmpb_tl }
6024   \clist_if_in:NnT \l_@@_borders_clist { left }
6025   { \@@_stroke_vertical:n \l_tmpd_tl }
6026   \clist_if_in:NnT \l_@@_borders_clist { bottom }
6027   { \@@_stroke_horizontal:n \l_tmpa_tl }
6028   \clist_if_in:NnT \l_@@_borders_clist { top }
6029   { \@@_stroke_horizontal:n \l_tmpc_tl }
6030   \endpgfpicture
6031 }
6032 }

```

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).

```

6033 \cs_new_protected:Npn \@@_stroke_vertical:n #1
6034 {
6035   \@@_qpoint:n \l_tmpc_tl
6036   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \l_@@_line_width_dim }
6037   \@@_qpoint:n \l_tmpa_tl
6038   \dim_set:Nn \l_tmpc_dim { \pgf@y + 0.5 \l_@@_line_width_dim }
6039   \@@_qpoint:n { #1 }
6040   \pgfpathmoveto { \pgfpoint \pgf@x \l_tmpb_dim }
6041   \pgfpathlineto { \pgfpoint \pgf@x \l_tmpc_dim }
6042   \pgfusepathqstroke
6043 }

```

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).

```

6044 \cs_new_protected:Npn \@@_stroke_horizontal:n #1
6045 {
6046   \@@_qpoint:n \l_tmpd_tl
6047   \clist_if_in:NnTF \l_@@_borders_clist { left }
6048   { \dim_set:Nn \l_tmpa_dim { \pgf@x - 0.5 \l_@@_line_width_dim } }
6049   { \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \l_@@_line_width_dim } }
6050   \@@_qpoint:n \l_tmpb_tl
6051   \dim_set:Nn \l_tmpb_dim { \pgf@x + 0.5 \l_@@_line_width_dim }
6052   \@@_qpoint:n { #1 }
6053   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \pgf@y }
6054   \pgfpathlineto { \pgfpoint \l_tmpb_dim \pgf@y }
6055   \pgfusepathqstroke
6056 }

```

Here is the set of keys for the command \@@\_stroke\_borders\_block:nnn.

```

6057 \keys_define:nn { NiceMatrix / BlockBorders }

```

```

6058 {
6059   borders .clist_set:N = \l_@@_borders_clist ,
6060   rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
6061   rounded-corners .default:n = 4 pt ,
6062   line-width .dim_set:N = \l_@@_line_width_dim
6063 }

```

The following command will be used if the key `tikz` has been used for the command `\Block`. The arguments `#1` and `#2` are the coordinates of the first cell and `#3` and `#4` the coordinates of the last cell of the block. `#5` is a comma-separated list of the Tikz keys used with the path.

```

6064 \cs_new_protected:Npn \@@_block_tikz:nnnnn #1 #2 #3 #4 #5
6065 {
6066   \begin { tikzpicture }
6067   \clist_map_inline:nn { #5 }
6068   {
6069     \path [ ##1 ]
6070       ( #1 -| #2 ) rectangle ( \@@_succ:n { #3 } -| \@@_succ:n { #4 } ) ;
6071   }
6072   \end { tikzpicture }
6073 }

```

## How to draw the dotted lines transparently

```

6074 \cs_set_protected:Npn \@@_renew_matrix:
6075 {
6076   \RenewDocumentEnvironment { pmatrix } { } {
6077     { \pNiceMatrix }
6078     { \endpNiceMatrix }
6079   }
6080   \RenewDocumentEnvironment { vmatrix } { } {
6081     { \vNiceMatrix }
6082     { \endvNiceMatrix }
6083   }
6084   \RenewDocumentEnvironment { Vmatrix } { } {
6085     { \VNiceMatrix }
6086     { \endVNiceMatrix }
6087   }
6088   \RenewDocumentEnvironment { bmatrix } { } {
6089     { \bNiceMatrix }
6090     { \endbNiceMatrix }
6091   }
6092   \RenewDocumentEnvironment { Bmatrix } { } {
6093     { \BNiceMatrix }
6094     { \endBNiceMatrix }
6095   }
6096 }

```

## Automatic arrays

```

6092 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
6093 {
6094   \int_set:Nn \l_@@_nb_rows_int { #1 }
6095   \int_set:Nn \l_@@_nb_cols_int { #2 }
6096 }

```

We will extract the potential keys `l`, `r` and `c` and pass the other keys to the environment `{NiceArrayWithDelims}`.

```

6097 \keys_define:nn { NiceMatrix / Auto }
6098 {
6099   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
6100   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
6101   c .code:n = \tl_set:Nn \l_@@_type_of_col_tl c
6102 }
6103 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
6104 {

```



```

6105 \int_zero_new:N \l_@@_nb_rows_int
6106 \int_zero_new:N \l_@@_nb_cols_int
6107 \@@_set_size:n #4 \q_stop

```

The group is for the protection of `\l_@@_type_of_col_tl`.

```

6108 \group_begin:
6109 \tl_set:Nn \l_@@_type_of_col_tl c
6110 \keys_set_known:nnN { NiceMatrix / Auto } { #3, #5, #7 } \l_tmpa_tl
6111 \use:x
6112 {
6113   \exp_not:N \begin { NiceArrayWithDelims } { #1 } { #2 }
6114   { * { \int_use:N \l_@@_nb_cols_int } { \l_@@_type_of_col_tl } }
6115   [ \exp_not:N \l_tmpa_tl ]
6116 }
6117 \int_compare:nNnT \l_@@_first_row_int = 0
6118 {
6119   \int_compare:nNnT \l_@@_first_col_int = 0 { & }
6120   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
6121   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
6122 }
6123 \prg_replicate:nn \l_@@_nb_rows_int
6124 {
6125   \int_compare:nNnT \l_@@_first_col_int = 0 { & }

```

We 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`).

```

6126   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
6127   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
6128 }
6129 \int_compare:nNnT \l_@@_last_row_int > { -2 }
6130 {
6131   \int_compare:nNnT \l_@@_first_col_int = 0 { & }
6132   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
6133   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
6134 }
6135 \end { NiceArrayWithDelims }
6136 \group_end:
6137 }
6138 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
6139 {
6140   \cs_set_protected:cpn { #1 AutoNiceMatrix }
6141   {
6142     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
6143     \AutoNiceMatrixWithDelims { #2 } { #3 }
6144   }
6145 }
6146 \@@_define_com:nnn p ( )
6147 \@@_define_com:nnn b [ ]
6148 \@@_define_com:nnn v | |
6149 \@@_define_com:nnn V \ | \ |
6150 \@@_define_com:nnn B \{ \}

```

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

```

6151 \NewDocumentCommand \AutoNiceMatrix { 0 { } m 0 { } m ! 0 { } }
6152 {
6153   \group_begin:
6154   \bool_set_true:N \l_@@_NiceArray_bool
6155   \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
6156   \group_end:
6157 }

```

The redefinition of the command `\dotfill`

```

6158 \cs_set_eq:NN \@@_old_dotfill \dotfill
6159 \cs_new_protected:Npn \@@_dotfill:
6160 {

```

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}`).

```

6161 \@@_old_dotfill
6162 \bool_if:NT \l_@@_NiceTabular_bool
6163 { \group_insert_after:N \@@_dotfill_ii: }
6164 { \group_insert_after:N \@@_dotfill_i: }
6165 }
6166 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
6167 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }

```

Now, if the box is not empty (unfortunately, we can’t actually test whether the box is empty and that’s why we only consider it’s width), we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in the cell of the array, and it will extend, since it is no longer in `\l_@@_cell_box`.

```

6168 \cs_new_protected:Npn \@@_dotfill_iii:
6169 { \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`. However, there are also redefinitions of `\diagbox` in other circumstances.

```

6170 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
6171 {
6172   \tl_gput_right:Nx \g_@@_internal_code_after_tl
6173   {
6174     \@@_actually_diagbox:nnnnnn
6175     { \int_use:N \c@iRow }
6176     { \int_use:N \c@jCol }
6177     { \int_use:N \c@iRow }
6178     { \int_use:N \c@jCol }
6179     { \exp_not:n { #1 } }
6180     { \exp_not:n { #2 } }
6181   }

```

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`.

```

6182   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
6183   {
6184     { \int_use:N \c@iRow }
6185     { \int_use:N \c@jCol }
6186     { \int_use:N \c@iRow }
6187     { \int_use:N \c@jCol }
6188   }
6189 }

```

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.

```

6190 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
6191 {
6192   \pgfpicture
6193   \pgf@relevantforpicturesizefalse
6194   \pgfrememberpicturepositiononpagetrue
6195   \@@_qpoint:n { row - #1 }
6196   \dim_set_eq:NN \l_tmpa_dim \pgf@y
6197   \@@_qpoint:n { col - #2 }
6198   \dim_set_eq:NN \l_tmpb_dim \pgf@x
6199   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
6200   \@@_qpoint:n { row - \@@_succ:n { #3 } }

```

```

6201 \dim_set_eq:NN \l_tmpc_dim \pgf@y
6202 \@@_qpoint:n { col - \@@_succ:n { #4 } }
6203 \dim_set_eq:NN \l_tmpd_dim \pgf@x
6204 \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
6205 {

```

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.

```

6206 \CT@arc@
6207 \pgfsetroundcap
6208 \pgfusepathqstroke
6209 }
6210 \pgfset { inner~sep = 1 pt }
6211 \pgfscope
6212 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
6213 \pgfnode { rectangle } { south~west }
6214 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
6215 \endpgfscope
6216 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
6217 \pgfnode { rectangle } { north~east }
6218 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
6219 \endpgfpicture
6220 }

```

## 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.

```

6221 \keys_define:nn { NiceMatrix }
6222 {
6223   CodeAfter / rules .inherit:n = NiceMatrix / rules ,
6224   CodeAfter / sub-matrix .inherit:n = NiceMatrix / sub-matrix
6225 }
6226 \keys_define:nn { NiceMatrix / CodeAfter }
6227 {
6228   sub-matrix .code:n = \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
6229   sub-matrix .value_required:n = true ,
6230   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
6231   delimiters / color .value_required:n = true ,
6232   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
6233   rules .value_required:n = true ,
6234   unknown .code:n = \@@_error:n { Unknown-key-for-CodeAfter }
6235 }

```

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. 115.

In the environments of `nicematrix`, `\CodeAfter` will be linked to `\@@_CodeAfter:`. That macro must *not* be protected since it begins with `\omit`.

```

6236 \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`.

```

6237 \cs_new_protected:Npn \@@_CodeAfter_i:n #1 \end
6238 {
6239   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
6240   \@@_CodeAfter_ii:n
6241 }

```

We catch the argument of the command `\end` (in `#1`).

```
6242 \cs_new_protected:Npn \@@_CodeAfter_ii:n #1
6243 {
```

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.

```
6244 \str_if_eq:eeTF \currentenv { #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`.

```
6245 {
6246 \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
6247 \@@_CodeAfter_i:n
6248 }
6249 }
```

## 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 columnn. The third argument is a boolean equal to `\c_true_bool` (resp. `\c_false_true`) when the delimiter must be put on the left (resp. right) side.

```
6250 \cs_new_protected:Npn \@@_delimiter:nnn #1 #2 #3
6251 {
6252 \pgfpicture
6253 \pgfrememberpicturepositiononpagetrue
6254 \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.

```
6255 \@@_qpoint:n { row - 1 }
6256 \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
6257 \@@_qpoint:n { row - \@@_succ:n \c@iRow }
6258 \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).

```
6259 \bool_if:nTF { #3 }
6260 { \dim_set_eq:NN \l_tmpa_dim \c_max_dim }
6261 { \dim_set:Nn \l_tmpa_dim { - \c_max_dim } }
6262 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
6263 {
6264 \cs_if_exist:cT
6265 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
6266 {
6267 \pgfpicture
6268 { \@@_env: - ##1 - #2 }
6269 { \bool_if:nTF { #3 } { west } { east } }
6270 \dim_set:Nn \l_tmpa_dim
6271 { \bool_if:nTF { #3 } \dim_min:nn \dim_max:nn \l_tmpa_dim \pgf@x }
6272 }
6273 }
```

Now we can put the delimiter with a node of PGF.

```

6274 \pgfset { inner~sep = \c_zero_dim }
6275 \dim_zero:N \nulldelimiterspace
6276 \pgftransformshift
6277 {
6278   \pgfpoint
6279   { \l_tmpa_dim }
6280   { ( \l_@@_y_initial_dim + \l_@@_y_final_dim + \arrayrulewidth ) / 2 }
6281 }
6282 \pgfnode
6283 { rectangle }
6284 { \bool_if:nTF { #3 } { east } { west } }
6285 {

```

Here is the content of the PGF node, that is to say the delimiter, constructed with its right size.

```

6286   \nullfont
6287   \c_math_toggle_token
6288   \tl_if_empty:NF \l_@@_delimiters_color_tl
6289   { \color { \l_@@_delimiters_color_tl } }
6290   \bool_if:nTF { #3 } { \left #1 } { \left . }
6291   \vcenter
6292   {
6293     \nullfont
6294     \hrule \@height
6295     \dim_eval:n { \l_@@_y_initial_dim - \l_@@_y_final_dim }
6296     \@depth \c_zero_dim
6297     \@width \c_zero_dim
6298   }
6299   \bool_if:nTF { #3 } { \right . } { \right #1 }
6300   \c_math_toggle_token
6301 }
6302 { }
6303 { }
6304 \endpgfpicture
6305 }

```

## The command \SubMatrix

```

6306 \keys_define:nn { NiceMatrix / sub-matrix }
6307 {
6308   extra-height .dim_set:N = \l_@@_submatrix_extra_height_dim ,
6309   extra-height .value_required:n = true ,
6310   left-xshift .dim_set:N = \l_@@_submatrix_left_xshift_dim ,
6311   left-xshift .value_required:n = true ,
6312   right-xshift .dim_set:N = \l_@@_submatrix_right_xshift_dim ,
6313   right-xshift .value_required:n = true ,
6314   xshift .meta:n = { left-xshift = #1, right-xshift = #1 } ,
6315   xshift .value_required:n = true ,
6316   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
6317   delimiters / color .value_required:n = true ,
6318   slim .bool_set:N = \l_@@_submatrix_slim_bool ,
6319   slim .default:n = true ,
6320   hlines .clist_set:N = \l_@@_submatrix_hlines_clist ,
6321   hlines .default:n = all ,
6322   vlines .clist_set:N = \l_@@_submatrix_vlines_clist ,
6323   vlines .default:n = all ,
6324   hvlines .meta:n = { hlines, vlines } ,
6325   hvlines .value_forbidden:n = true ,
6326 }
6327 \keys_define:nn { NiceMatrix }
6328 {
6329   SubMatrix .inherit:n = NiceMatrix / sub-matrix ,
6330   CodeAfter / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,

```

```

6331 NiceMatrix / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
6332 NiceArray / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
6333 pNiceArray / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
6334 NiceMatrixOptions / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
6335 }

```

The following keys set is for the command `\SubMatrix` itself (not the tuning of `\SubMatrix` that can be done elsewhere).

```

6336 \keys_define:nn { NiceMatrix / SubMatrix }
6337 {
6338   hlines .clist_set:N = \l_@@_submatrix_hlines_clist ,
6339   hlines .default:n = all ,
6340   vlines .clist_set:N = \l_@@_submatrix_vlines_clist ,
6341   vlines .default:n = all ,
6342   hvlines .meta:n = { hlines, vlines } ,
6343   hvlines .value_forbidden:n = true ,
6344   name .code:n =
6345     \tl_if_empty:nTF { #1 }
6346     { \@@_error:n { Invalid-name-format } }
6347     {
6348       \regex_match:nnTF { \A[A-Za-z][A-Za-z0-9]*\Z } { #1 }
6349       {
6350         \seq_if_in:NnTF \g_@@_submatrix_names_seq { #1 }
6351         { \@@_error:nn { Duplicate-name-for-SubMatrix } { #1 } }
6352         {
6353           \str_set:Nn \l_@@_submatrix_name_str { #1 }
6354           \seq_gput_right:Nn \g_@@_submatrix_names_seq { #1 }
6355         }
6356       }
6357       { \@@_error:n { Invalid-name-format } }
6358     } ,
6359   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
6360   rules .value_required:n = true ,
6361   code .tl_set:N = \l_@@_code_tl ,
6362   code .value_required:n = true ,
6363   name .value_required:n = true ,
6364   unknown .code:n = \@@_error:n { Unknown-key-for-SubMatrix }
6365 }

6366 \NewDocumentCommand \@@_SubMatrix_in_code_before { m m m m ! O { } }
6367 {
6368   \peek_remove_spaces:n
6369   {
6370     \@@_cut_on_hyphen:w #3 \q_stop
6371     \tl_clear_new:N \l_tmpc_tl
6372     \tl_clear_new:N \l_tmpd_tl
6373     \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
6374     \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
6375     \@@_cut_on_hyphen:w #2 \q_stop
6376     \seq_gput_right:Nx \g_@@_submatrix_seq
6377     { { \l_tmpa_tl } { \l_tmpb_tl } { \l_tmpc_tl } { \l_tmpd_tl } }
6378     \tl_gput_right:Nn \g_@@_internal_code_after_tl
6379     { \SubMatrix { #1 } { #2 } { #3 } { #4 } [ #5 ] }
6380   }
6381 }

```

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;
- #6 is the potential subscript;
- #7 is the potential superscript.

For explanations about the construction with rescanning of the preamble, see the documentation for the user command `\Cdots`.

```

6382 \AtBeginDocument
6383 {
6384   \tl_set:Nn \l_@@_argspec_tl { m m m m 0 { } E { _ ^ } { { } { } } }
6385   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
6386   \exp_args:NNV \NewDocumentCommand \@@_SubMatrix \l_@@_argspec_tl
6387   {
6388     \peek_remove_spaces:n
6389     { \@@_sub_matrix:nnnnnnn
6390       { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } { #7 } }
6391   }
6392 }

6393 \cs_new_protected:Npn \@@_sub_matrix:nnnnnnn #1 #2 #3 #4 #5 #6 #7
6394 {
6395   \group_begin:

```

The four following token lists correspond to the position of the `\SubMatrix`.

```

6396 \tl_clear_new:N \l_@@_first_i_tl
6397 \tl_clear_new:N \l_@@_first_j_tl
6398 \tl_clear_new:N \l_@@_last_i_tl
6399 \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`.

```

6400 \@@_cut_on_hyphen:w #2 \q_stop
6401 \tl_set_eq:NN \l_@@_first_i_tl \l_tmpa_tl
6402 \tl_set_eq:NN \l_@@_first_j_tl \l_tmpb_tl
6403 \@@_cut_on_hyphen:w #3 \q_stop
6404 \tl_set_eq:NN \l_@@_last_i_tl \l_tmpa_tl
6405 \tl_set_eq:NN \l_@@_last_j_tl \l_tmpb_tl
6406 \bool_lazy_or:nnTF
6407 { \int_compare_p:nNn \l_@@_last_i_tl > \g_@@_row_total_int }
6408 { \int_compare_p:nNn \l_@@_last_j_tl > \g_@@_col_total_int }
6409 { \@@_error:n { SubMatrix~too~large } }
6410 {
6411   \str_clear_new:N \l_@@_submatrix_name_str
6412   \keys_set:nn { NiceMatrix / SubMatrix } { #5 }
6413   \pgfpicture
6414   \pgfrememberpicturepositiononpagetrue
6415   \pgf@relevantforpicturesizefalse
6416   \pgfset { inner-sep = \c_zero_dim }
6417   \dim_set_eq:NN \l_@@_x_initial_dim \c_max_dim
6418   \dim_set:Nn \l_@@_x_final_dim { - \c_max_dim }

```

The last value of `\int_step_inline:nnn` is provided by currying.

```

6419 \bool_if:NTF \l_@@_submatrix_slim_bool
6420 { \int_step_inline:nnn \l_@@_first_i_tl \l_@@_last_i_tl }
6421 { \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int }
6422 {
6423   \cs_if_exist:cT
6424   { pgf @ sh @ ns @ \@@_env: - ##1 - \l_@@_first_j_tl }
6425   {
6426     \pgfpointanchor { \@@_env: - ##1 - \l_@@_first_j_tl } { west }
6427     \dim_set:Nn \l_@@_x_initial_dim
6428     { \dim_min:nn \l_@@_x_initial_dim \pgf@x }
6429   }

```

```

6430         \cs_if_exist:cT
6431         { pgf @ sh @ ns @ \@@_env: - ##1 - \l_@@_last_j_tl }
6432         {
6433             \pgfpointanchor { \@@_env: - ##1 - \l_@@_last_j_tl } { east }
6434             \dim_set:Nn \l_@@_x_final_dim
6435             { \dim_max:nn \l_@@_x_final_dim \pgf@x }
6436         }
6437     }
6438     \dim_compare:nNnTF \l_@@_x_initial_dim = \c_max_dim
6439     { \@@_error:nn { impossible-delimiter } { left } }
6440     {
6441         \dim_compare:nNnTF \l_@@_x_final_dim = { - \c_max_dim }
6442         { \@@_error:nn { impossible~delimiter } { right } }
6443         { \@@_sub_matrix_i:nnnn { #1 } { #4 } { #6 } { #7 } }
6444     }
6445     \endpgfpicture
6446 }
6447 \group_end:
6448 }

```

#1 is the left delimiter, #2 is the right one, #3 is the subscript and #4 is the superscript.

```

6449 \cs_new_protected:Npn \@@_sub_matrix_i:nnnn #1 #2 #3 #4
6450 {
6451     \@@_qpoint:n { row - \l_@@_first_i_tl - base }
6452     \dim_set:Nn \l_@@_y_initial_dim
6453     { \pgf@y + ( \box_ht:N \strutbox + \extrarowheight ) * \arraystretch }
6454     \@@_qpoint:n { row - \l_@@_last_i_tl - base }
6455     \dim_set:Nn \l_@@_y_final_dim
6456     { \pgf@y - ( \box_dp:N \strutbox ) * \arraystretch }
6457     \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
6458     {
6459         \cs_if_exist:cT
6460         { pgf @ sh @ ns @ \@@_env: - \l_@@_first_i_tl - ##1 }
6461         {
6462             \pgfpointanchor { \@@_env: - \l_@@_first_i_tl - ##1 } { north }
6463             \dim_set:Nn \l_@@_y_initial_dim
6464             { \dim_max:nn \l_@@_y_initial_dim \pgf@y }
6465         }
6466         \cs_if_exist:cT
6467         { pgf @ sh @ ns @ \@@_env: - \l_@@_last_i_tl - ##1 }
6468         {
6469             \pgfpointanchor { \@@_env: - \l_@@_last_i_tl - ##1 } { south }
6470             \dim_set:Nn \l_@@_y_final_dim
6471             { \dim_min:nn \l_@@_y_final_dim \pgf@y }
6472         }
6473     }
6474     \dim_set:Nn \l_tmpa_dim
6475     {
6476         \l_@@_y_initial_dim - \l_@@_y_final_dim +
6477         \l_@@_submatrix_extra_height_dim - \arrayrulewidth
6478     }
6479     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim

```

We will draw the rules in the `\SubMatrix`.

```

6480 \group_begin:
6481 \pgfsetlinewidth { 1.1 \arrayrulewidth }
6482 \tl_if_empty:NF \l_@@_rules_color_tl
6483 { \exp_after:wN \@@_set_CT@arc@: \l_@@_rules_color_tl \q_stop }
6484 \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`.



```

6485 \seq_map_inline:Nn \g_@@_cols_vlism_seq
6486 {
6487   \int_compare:nNnT \l_@@_first_j_tl < { ##1 }
6488   {
6489     \int_compare:nNnT
6490       { ##1 } < { \int_eval:n { \l_@@_last_j_tl + 1 } }
6491     {

```

First, we extract the value of the abscissa of the rule we have to draw.

```

6492       \@@_qpoint:n { col - ##1 }
6493       \pgfpathmoveto { \pgfpoint \pgf@x \l_@@_y_initial_dim }
6494       \pgfpathlineto { \pgfpoint \pgf@x \l_@@_y_final_dim }
6495       \pgfusepathqstroke
6496     }
6497   }
6498 }

```

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.

```

6499 \tl_if_eq:NnTF \l_@@_submatrix_vlines_clist { all }
6500 { \int_step_inline:nn { \l_@@_last_j_tl - \l_@@_first_j_tl } }
6501 { \clist_map_inline:Nn \l_@@_submatrix_vlines_clist }
6502 {
6503   \bool_lazy_and:nnTF
6504     { \int_compare_p:nNn { ##1 } > 0 }
6505     {
6506       \int_compare_p:nNn
6507         { ##1 } < { \l_@@_last_j_tl - \l_@@_first_j_tl + 1 } }
6508     {
6509       \@@_qpoint:n { col - \int_eval:n { ##1 + \l_@@_first_j_tl } }
6510       \pgfpathmoveto { \pgfpoint \pgf@x \l_@@_y_initial_dim }
6511       \pgfpathlineto { \pgfpoint \pgf@x \l_@@_y_final_dim }
6512       \pgfusepathqstroke
6513     }
6514     { \@@_error:nnn { Wrong-line-in-SubMatrix } { vertical } { ##1 } }
6515 }

```

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.

```

6516 \tl_if_eq:NnTF \l_@@_submatrix_hlines_clist { all }
6517 { \int_step_inline:nn { \l_@@_last_i_tl - \l_@@_first_i_tl } }
6518 { \clist_map_inline:Nn \l_@@_submatrix_hlines_clist }
6519 {
6520   \bool_lazy_and:nnTF
6521     { \int_compare_p:nNn { ##1 } > 0 }
6522     {
6523       \int_compare_p:nNn
6524         { ##1 } < { \l_@@_last_i_tl - \l_@@_first_i_tl + 1 } }
6525     {
6526       \@@_qpoint:n { row - \int_eval:n { ##1 + \l_@@_first_i_tl } }

```

We use a group to protect `\l_tmpa_dim` and `\l_tmpb_dim`.

```

6527 \group_begin:

```

We compute in `\l_tmpa_dim` the  $x$ -value of the left end of the rule.

```

6528 \dim_set:Nn \l_tmpa_dim
6529 { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
6530 \str_case:nn { #1 }
6531 {
6532   ( { \dim_sub:Nn \l_tmpa_dim { 0.9 mm } }
6533   [ { \dim_sub:Nn \l_tmpa_dim { 0.2 mm } }
6534   \{ { \dim_sub:Nn \l_tmpa_dim { 0.9 mm } }
6535   }
6536   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \pgf@y }

```

We compute in `\l_tmpb_dim` the  $x$ -value of the right end of the rule.

```

6537         \dim_set:Nn \l_tmpb_dim
6538         { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
6539         \str_case:nn { #2 }
6540         {
6541             ) { \dim_add:Nn \l_tmpb_dim { 0.9 mm } }
6542             ] { \dim_add:Nn \l_tmpb_dim { 0.2 mm } }
6543             \} { \dim_add:Nn \l_tmpb_dim { 0.9 mm } }
6544         }
6545         \pgfpathlineto { \pgfpoint \l_tmpb_dim \pgf@y }
6546         \pgfusepathqstroke
6547         \group_end:
6548     }
6549     { \@@_error:nnn { Wrong~line~in~SubMatrix } { horizontal } { ##1 } }
6550 }

```

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).

```

6551     \str_if_empty:NF \l_@@_submatrix_name_str
6552     {
6553         \@@_pgf_rect_node:nnnnn \l_@@_submatrix_name_str
6554         \l_@@_x_initial_dim \l_@@_y_initial_dim
6555         \l_@@_x_final_dim \l_@@_y_final_dim
6556     }
6557     \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`.

```

6558     \begin { pgfscope }
6559     \pgftransformshift
6560     {
6561         \pgfpoint
6562         { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
6563         { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
6564     }
6565     \str_if_empty:NTF \l_@@_submatrix_name_str
6566     { \@@_node_left:nn #1 { } }
6567     { \@@_node_left:nn #1 { \@@_env: - \l_@@_submatrix_name_str - left } }
6568     \end { pgfscope }

```

Now, we deal with the right delimiter.

```

6569     \pgftransformshift
6570     {
6571         \pgfpoint
6572         { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
6573         { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
6574     }
6575     \str_if_empty:NTF \l_@@_submatrix_name_str
6576     { \@@_node_right:nnnn #2 { } { #3 } { #4 } }
6577     {
6578         \@@_node_right:nnnn #2
6579         { \@@_env: - \l_@@_submatrix_name_str - right } { #3 } { #4 }
6580     }
6581     \cs_set_eq:NN \pgfpointanchor \@@_pgfpointanchor:n
6582     \flag_clear_new:n { nicematrix }
6583     \l_@@_code_tl
6584 }

```

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`.

```
6585 \cs_set_eq:NN \@@_old_pgfpntanchor \pgfpntanchor
```

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 `\@@_pgfpntanchor_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.

```
6586 \cs_new_protected:Npn \@@_pgfpntanchor:n #1
6587 {
6588   \use:e
6589   { \exp_not:N \@@_old_pgfpntanchor { \@@_pgfpntanchor_i:nn #1 } }
6590 }
```

In fact, the argument of `\pgfpointanchor` 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 -.

```
6591 \cs_new:Npn \@@_pgfpntanchor_i:nn #1 #2
6592 { #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.

```
6593 \tl_const:Nn \c_@@_integers_alist_tl
6594 {
6595   { 1 } { } { 2 } { } { 3 } { } { 4 } { } { 5 } { }
6596   { 6 } { } { 7 } { } { 8 } { } { 9 } { } { 10 } { }
6597   { 11 } { } { 12 } { } { 13 } { } { 14 } { } { 15 } { }
6598   { 16 } { } { 17 } { } { 18 } { } { 19 } { } { 20 } { }
6599 }

6600 \cs_new:Npn \@@_pgfpntanchor_ii:w #1-#2\q_stop
6601 {
```

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 `\pgfpointanchor` and, the, the  $j$  arrives (alone) in the following `\pgfpointanchor`. 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`.

```
6602   \tl_if_empty:nTF { #2 }
6603   {
6604     \str_case:nVTF { #1 } \c_@@_integers_alist_tl
6605     {
6606       \flag_raise:n { nicematrix }
6607       \int_if_even:nTF { \flag_height:n { nicematrix } }
6608       { \int_eval:n { #1 + \l_@@_first_i_tl - 1 } }
6609       { \int_eval:n { #1 + \l_@@_first_j_tl - 1 } }
6610     }
6611     { #1 }
6612   }
```

If there is an hyphen, we have to see whether we have a node of the form  $i-j$ , row- $i$  or col- $j$ .

```
6613   { \@@_pgfpntanchor_iii:w { #1 } #2 }
6614 }
```

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`).

```
6615 \cs_new:Npn \@@_pgfpntanchor_iii:w #1 #2 -
```

```

6616 {
6617   \str_case:nnF { #1 }
6618   {
6619     { row } { row - \int_eval:n { #2 + \l_@@_first_i_tl - 1 } }
6620     { col } { col - \int_eval:n { #2 + \l_@@_first_j_tl - 1 } }
6621   }

```

Now the case of a node of the form  $i-j$ .

```

6622 {
6623   \int_eval:n { #1 + \l_@@_first_i_tl - 1 }
6624   - \int_eval:n { #2 + \l_@@_first_j_tl - 1 }
6625 }
6626 }

```

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`).

```

6627 \cs_new_protected:Npn \@@_node_left:nn #1 #2
6628 {
6629   \pgfnode
6630   { rectangle }
6631   { east }
6632   {
6633     \nullfont
6634     \c_math_toggle_token
6635     \tl_if_empty:NF \l_@@_delimiters_color_tl
6636     { \color { \l_@@_delimiters_color_tl } }
6637     \left #1
6638     \vcenter
6639     {
6640       \nullfont
6641       \hrule \@height \l_tmpa_dim
6642       \@depth \c_zero_dim
6643       \@width \c_zero_dim
6644     }
6645     \right .
6646     \c_math_toggle_token
6647   }
6648   { #2 }
6649   { }
6650 }

```

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`). The argument `#3` is the subscript and `#4` is the superscript.

```

6651 \cs_new_protected:Npn \@@_node_right:nnnn #1 #2 #3 #4
6652 {
6653   \pgfnode
6654   { rectangle }
6655   { west }
6656   {
6657     \nullfont
6658     \c_math_toggle_token
6659     \tl_if_empty:NF \l_@@_delimiters_color_tl
6660     { \color { \l_@@_delimiters_color_tl } }
6661     \left .
6662     \vcenter
6663     {
6664       \nullfont
6665       \hrule \@height \l_tmpa_dim
6666       \@depth \c_zero_dim
6667       \@width \c_zero_dim
6668     }

```

```

6669     \right #1
6670     \tl_if_empty:nF { #3 } { _ { \smash { #3 } } }
6671     ^ { \smash { #4 } }
6672     \c_math_toggle_token
6673   }
6674   { #2 }
6675   { }
6676 }

```

## 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.

```

6677 \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.

```

6678 \bool_new:N \c_@@_footnote_bool
6679 \@@_msg_new:nnn { Unknown-key-for-package }
6680 {
6681   The-key~'\l_keys_key_str'~is-unknown. \\
6682   If-you-go-on,~it~will~be~ignored. \\
6683   For-a-list-of~the~available~keys,~type-H~<return>.
6684 }
6685 {
6686   The-available-keys~are~(in~alphabetic~order):~
6687   footnote,~
6688   footnotehyper,~
6689   renew-dots,~and
6690   renew-matrix.
6691 }

```

Maybe we will completely delete the key 'transparent' in a future version.

```

6692 \@@_msg_new:nn { Key-transparent }
6693 {
6694   The-key~'transparent'~is-now-obsolete~(because~it's~name~
6695   is~not~clear).~You~should~use~the~conjunction~of~'renew-dots'~
6696   and~'renew-matrix'.~However,~you~can~go~on.
6697 }
6698 \keys_define:nn { NiceMatrix / Package }
6699 {
6700   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
6701   renew-dots .value_forbidden:n = true ,
6702   renew-matrix .code:n = \@@_renew_matrix: ,
6703   renew-matrix .value_forbidden:n = true ,
6704   transparent .code:n =
6705     {
6706       \@@_renew_matrix:
6707       \bool_set_true:N \l_@@_renew_dots_bool
6708       \@@_error:n { Key-transparent }
6709     } ,
6710   transparent .value_forbidden:n = true,
6711   footnote .bool_set:N = \c_@@_footnote_bool ,
6712   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
6713   unknown .code:n = \@@_error:n { Unknown-key-for-package }

```

```

6714 }
6715 \ProcessKeysOptions { NiceMatrix / Package }

6716 \@@_msg_new:nn { footnote~with~footnotehyper~package }
6717 {
6718   You~can't~use~the~option~'footnote'~because~the~package~
6719   footnotehyper~has~already~been~loaded.~
6720   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
6721   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
6722   of~the~package~footnotehyper.\\
6723   If~you~go~on,~the~package~footnote~won't~be~loaded.
6724 }

6725 \@@_msg_new:nn { footnotehyper~with~footnote~package }
6726 {
6727   You~can't~use~the~option~'footnotehyper'~because~the~package~
6728   footnote~has~already~been~loaded.~
6729   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
6730   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
6731   of~the~package~footnote.\\
6732   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
6733 }

6734 \bool_if:NT \c_@@_footnote_bool
6735 {

```

The class beamer has its own system to extract footnotes and that's why we have nothing to do if beamer is used.

```

6736 \@ifclassloaded { beamer }
6737 { \bool_set_false:N \c_@@_footnote_bool }
6738 {
6739   \@ifpackageloaded { footnotehyper }
6740   { \@@_error:n { footnote~with~footnotehyper~package } }
6741   { \usepackage { footnote } }
6742 }
6743 }

6744 \bool_if:NT \c_@@_footnotehyper_bool
6745 {

```

The class beamer has its own system to extract footnotes and that's why we have nothing to do if beamer is used.

```

6746 \@ifclassloaded { beamer }
6747 { \bool_set_false:N \c_@@_footnote_bool }
6748 {
6749   \@ifpackageloaded { footnote }
6750   { \@@_error:n { footnotehyper~with~footnote~package } }
6751   { \usepackage { footnotehyper } }
6752 }
6753 \bool_set_true:N \c_@@_footnote_bool
6754 }

```

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`.

```

6755 \@@_msg_new:nn { key except-corners }
6756 {
6757   The~key~'except-corners'~has~been~deleted~for~the~command~\token_to_str:N

```

```

6758 \arraycolor\ in-the-\token_to_str:N \CodeBefore.~You~should~instead~use~
6759 the~key~'corners'~in~your~\@@_full_name_env:.\
6760 If~you~go~on,~this~key~will~be~ignored.
6761 }

6762 \seq_new:N \c_@@_types_of_matrix_seq
6763 \seq_set_from_clist:Nn \c_@@_types_of_matrix_seq
6764 {
6765   NiceMatrix ,
6766   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
6767 }
6768 \seq_set_map_x:NNn \c_@@_types_of_matrix_seq \c_@@_types_of_matrix_seq
6769 { \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`.

```

6770 \cs_new_protected:Npn \@@_error_too_much_cols:
6771 {
6772   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
6773   {
6774     \int_compare:nNnTF \l_@@_last_col_int = { -2 }
6775     { \@@_fatal:n { too~much~cols~for~matrix } }
6776     {
6777       \bool_if:NF \l_@@_last_col_without_value_bool
6778       { \@@_fatal:n { too~much~cols~for~matrix~with~last~col } }
6779     }
6780   }
6781   { \@@_fatal:n { too~much~cols~for~array } }
6782 }

```

The following command must *not* be protected since it's used in an error message.

```

6783 \cs_new:Npn \@@_message_hdotsfor:
6784 {
6785   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
6786   { ~Maybe~your~use~of~\token_to_str:N \Hdotsfor\ is~incorrect.}
6787 }

6788 \@@_msg_new:nn { too~much~cols~for~matrix~with~last~col }
6789 {
6790   You~try~to~use~more~columns~than~allowed~by~your~
6791   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~
6792   columns~is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus~the~
6793   exterior~columns).~This~error~is~fatal.
6794 }

6795 \@@_msg_new:nn { too~much~cols~for~matrix }
6796 {
6797   You~try~to~use~more~columns~than~allowed~by~your~
6798   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall~that~the~maximal~
6799   number~of~columns~for~a~matrix~is~fixed~by~the~LaTeX~counter~
6800   'MaxMatrixCols'.~Its~actual~value~is~\int_use:N \c@MaxMatrixCols.~
6801   This~error~is~fatal.
6802 }

```

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`.

```

6803 \@@_msg_new:nn { too~much~cols~for~array }
6804 {
6805   You~try~to~use~more~columns~than~allowed~by~your~
6806   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~columns~is~
6807   \int_use:N \g_@@_static_num_of_col_int\
6808   ~(plus~the~potential~exterior~ones).~

```

```

6809     This-error-is-fatal.
6810 }
6811 \@@_msg_new:nn { last-col-not-used }
6812 {
6813     The-key~'last-col'~is~in~force~but~you~have~not~used~that~last~column~
6814     in~your~\@@_full_name_env:~.~However,~you~can~go~on.
6815 }
6816 \@@_msg_new:nn { columns-not-used }
6817 {
6818     The-preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
6819     \g_@@_static_num_of_col_int\ columns~but~you~use~only~\int_use:N \c@jCol.\
6820     You~can~go~on~but~the~columns~you~did~not~used~won't~be~created.
6821 }
6822 \@@_msg_new:nn { in-first-col }
6823 {
6824     You~can't~use~the~command~\#1 in~the~first~column~(number~0)~of~the~array.\
6825     If~you~go~on,~this~command~will~be~ignored.
6826 }
6827 \@@_msg_new:nn { in-last-col }
6828 {
6829     You~can't~use~the~command~\#1 in~the~last~column~(exterior)~of~the~array.\
6830     If~you~go~on,~this~command~will~be~ignored.
6831 }
6832 \@@_msg_new:nn { in-first-row }
6833 {
6834     You~can't~use~the~command~\#1 in~the~first~row~(number~0)~of~the~array.\
6835     If~you~go~on,~this~command~will~be~ignored.
6836 }
6837 \@@_msg_new:nn { in-last-row }
6838 {
6839     You~can't~use~the~command~\#1 in~the~last~row~(exterior)~of~the~array.\
6840     If~you~go~on,~this~command~will~be~ignored.
6841 }
6842 \@@_msg_new:nn { double-closing-delimiter }
6843 {
6844     You~can't~put~a~second~closing~delimiter~"#1"~just~after~a~first~closing~
6845     delimiter.~This~delimiter~will~be~ignored.
6846 }
6847 \@@_msg_new:nn { delimiter-after-opening }
6848 {
6849     You~can't~put~a~second~delimiter~"#1"~just~after~a~first~opening~
6850     delimiter.~This~delimiter~will~be~ignored.
6851 }
6852 \@@_msg_new:nn { bad-option-for-line-style }
6853 {
6854     Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line-style'~
6855     is~'standard'.~If~you~go~on,~this~key~will~be~ignored.
6856 }
6857 \@@_msg_new:nn { Unknown-key-for-xdots }
6858 {
6859     As~for~now,~there~is~only~three~keys~available~here:~'color',~'line-style'~
6860     and~'shorten'~(and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
6861     this~key~will~be~ignored.
6862 }
6863 \@@_msg_new:nn { Unknown-key-for-RowStyle }
6864 {
6865     As~for~now,~there~is~only~four~keys~available~here:~'cell-space-top-limit',~
6866     'cell-space-bottom-limit'~'cell-space-limits'~and~color~(and~you~try~to~use~
6867     '\l_keys_key_str').~If~you~go~on,~this~key~will~be~ignored.
6868 }

```



```

6869 \@@_msg_new:nn { Unknown-key-for-rowcolors }
6870 {
6871     As-for-now,~there-is-only~two-keys-available-here:~'cols'~and~'respect-blocks'~
6872     (and-you-try-to-use~'\l_keys_key_str').~If-you-go-on,~
6873     this-key-will-be-ignored.
6874 }
6875 \@@_msg_new:nn { ampersand-in-light-syntax }
6876 {
6877     You-can't-use-an-ampersand~(\token_to_str:N &)~to-separate-columns-because~
6878     ~you-have-used-the-key~'light-syntax'.~This-error-is-fatal.
6879 }
6880 \@@_msg_new:nn { SubMatrix-too-large }
6881 {
6882     Your-command~\token_to_str:N \SubMatrix\
6883     can't-be-drawn-because-your-matrix-is-too-small.\\
6884     If-you-go-on,~this-command-will-be-ignored.
6885 }
6886 \@@_msg_new:nn { double-backslash-in-light-syntax }
6887 {
6888     You-can't-use~\token_to_str:N \\~to-separate-rows-because-you-have-used~
6889     the-key~'light-syntax'.~You-must-use-the-character~'\l_@@_end_of_row_tl'~
6890     (set-by-the-key~'end-of-row').~This-error-is-fatal.
6891 }
6892 \@@_msg_new:nn { standard-cline-in-document }
6893 {
6894     The-key~'standard-cline'~is-available-only-in-the-preamble.\\
6895     If-you-go-on~this-command-will-be-ignored.
6896 }
6897 \@@_msg_new:nn { bad-value-for-baseline }
6898 {
6899     The-value-given-to~'baseline'~(\int_use:N \l_tmpa_int)~is-not~
6900     valid.~The-value-must-be-between~\int_use:N \l_@@_first_row_int\ and~
6901     \int_use:N \g_@@_row_total_int\ or-equal-to~'t',~'c'~or~'b'.\\
6902     If-you-go-on,~a-value-of~1~will-be-used.
6903 }
6904 \@@_msg_new:nn { Invalid-name-format }
6905 {
6906     You-can't-give-the-name~'\l_keys_value_tl'~to-a~\token_to_str:N
6907     \SubMatrix.\\
6908     A-name-must-be-accepted-by-the-regular-expression~[A-Za-z][A-Za-z0-9]*.\\
6909     If-you-go-on,~this-key-will-be-ignored.
6910 }
6911 \@@_msg_new:nn { Wrong-line-in-SubMatrix }
6912 {
6913     You-try-to-draw-a~#1~line-of-number~'#2'~in-a~
6914     \token_to_str:N \SubMatrix\ of-your~\@@_full_name_env:\ but-that~
6915     number-is-not-valid.~If-you-go-on,~it-will-be-ignored.
6916 }
6917 \@@_msg_new:nn { impossible-delimiter }
6918 {
6919     It's-impossible-to-draw-the~#1~delimiter~of-your~
6920     \token_to_str:N \SubMatrix\ because-all-the-cells-are-empty~
6921     in-that-column.
6922     \bool_if:NT \l_@@_submatrix_slim_bool
6923     { ~Maybe-you-should-try-without-the-key~'slim'. } \\
6924     If-you-go-on,~this~\token_to_str:N \SubMatrix\ will-be-ignored.
6925 }
6926 \@@_msg_new:nn { width-without-X-columns }
6927 {
6928     You-have-used-the-key~'width'~but-you-have-put-no~'X'~column. \\

```

```

6929     If~you~go~on,~that~key~will~be~ignored.
6930 }

6931 \@@_msg_new:nn { empty~environment }
6932 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }

6933 \@@_msg_new:nn { Delimiter~with~small }
6934 {
6935     You~can't~put~a~delimiter~in~the~preamble~of~your~\@@_full_name_env:\
6936     because~the~key~'small'~is~in~force.\\
6937     This~error~is~fatal.
6938 }

6939 \@@_msg_new:nn { unknown~cell~for~line~in~CodeAfter }
6940 {
6941     Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
6942     can't~be~executed~because~a~cell~doesn't~exist.\\
6943     If~you~go~on~this~command~will~be~ignored.
6944 }

6945 \@@_msg_new:nnn { Duplicate~name~for~SubMatrix }
6946 {
6947     The~name~'#1'~is~already~used~for~a~\token_to_str:N \SubMatrix\
6948     in~this~\@@_full_name_env:.\
6949     If~you~go~on,~this~key~will~be~ignored.\\
6950     For~a~list~of~the~names~already~used,~type~H~<return>.
6951 }
6952 {
6953     The~names~already~defined~in~this~\@@_full_name_env:\ are:~
6954     \seq_use:Nnnn \g_@@_submatrix_names_seq { ~and~ } { ,~ } { ~and~ }.
6955 }

6956 \@@_msg_new:nn { r~or~l~with~preamble }
6957 {
6958     You~can't~use~the~key~'\l_keys_key_str'~in~your~\@@_full_name_env:~
6959     You~must~specify~the~alignment~of~your~columns~with~the~preamble~of~
6960     your~\@@_full_name_env:.\
6961     If~you~go~on,~this~key~will~be~ignored.
6962 }

6963 \@@_msg_new:nn { Hdotsfor~in~col~0 }
6964 {
6965     You~can't~use~\token_to_str:N \Hdotsfor\ in~an~exterior~column~of~
6966     the~array.~This~error~is~fatal.
6967 }

6968 \@@_msg_new:nn { bad~corner }
6969 {
6970     #1~is~an~incorrect~specification~for~a~corner~(in~the~keys~
6971     'corners'~and~'except~corners').~The~available~
6972     values~are:~NW,~SW,~NE~and~SE.\\
6973     If~you~go~on,~this~specification~of~corner~will~be~ignored.
6974 }

6975 \@@_msg_new:nn { bad~border }
6976 {
6977     #1~is~an~incorrect~specification~for~a~border~(in~the~key~
6978     'borders'~of~the~command~\token_to_str:N \Block).~The~available~
6979     values~are:~left,~right,~top~and~bottom.\\
6980     If~you~go~on,~this~specification~of~border~will~be~ignored.
6981 }

6982 \@@_msg_new:nn { tikz~key~without~tikz }
6983 {
6984     You~can't~use~the~key~'tikz'~for~the~command~'\token_to_str:N
6985     \Block'~because~you~have~not~loaded~Tikz.~
6986     If~you~go~on,~this~key~will~be~ignored.
6987 }

```

```

6988 \@@_msg_new:nn { last-col~non~empty~for~NiceArray }
6989 {
6990   In~the~\@@_full_name_env:,~you~must~use~the~key~
6991   'last-col'~without~value.\\
6992   However,~you~can~go~on~for~this~time~
6993   (the~value~'\l_keys_value_tl'~will~be~ignored).
6994 }
6995 \@@_msg_new:nn { last-col~non~empty~for~NiceMatrixOptions }
6996 {
6997   In~\NiceMatrixoptions,~you~must~use~the~key~
6998   'last-col'~without~value.\\
6999   However,~you~can~go~on~for~this~time~
7000   (the~value~'\l_keys_value_tl'~will~be~ignored).
7001 }
7002 \@@_msg_new:nn { Block~too~large~1 }
7003 {
7004   You~try~to~draw~a~block~in~the~cell~#1~#2~of~your~matrix~but~the~matrix~is~
7005   too~small~for~that~block. \\
7006 }
7007 \@@_msg_new:nn { Block~too~large~2 }
7008 {
7009   The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
7010   \g_@@_static_num_of_col_int\
7011   columns~but~you~use~only~\int_use:N \c@jCol\ and~that's~why~a~block~
7012   specified~in~the~cell~#1~#2~can't~be~drawn.~You~should~add~some~ampersands~
7013   (&)~at~the~end~of~the~first~row~of~your~
7014   \@@_full_name_env:.\\
7015   If~you~go~on,~this~block~and~maybe~others~will~be~ignored.
7016 }
7017 \@@_msg_new:nn { unknown~column~type }
7018 {
7019   The~column~type~'#1'~in~your~\@@_full_name_env:\
7020   is~unknown. \\
7021   This~error~is~fatal.
7022 }
7023 \@@_msg_new:nn { tabularnote~forbidden }
7024 {
7025   You~can't~use~the~command~\token_to_str:N\tabularnote\
7026   ~in~a~\@@_full_name_env:.~This~command~is~available~only~in~
7027   \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
7028   If~you~go~on,~this~command~will~be~ignored.
7029 }
7030 \@@_msg_new:nn { borders~forbidden }
7031 {
7032   You~can't~use~the~key~'borders'~of~the~command~\token_to_str:N \Block\
7033   because~the~option~'rounded-corners'~
7034   is~in~force~with~a~non-zero~value.\\
7035   If~you~go~on,~this~key~will~be~ignored.
7036 }
7037 \@@_msg_new:nn { bottomrule~without~booktabs }
7038 {
7039   You~can't~use~the~key~'tabular/bottomrule'~because~you~haven't~
7040   loaded~'booktabs'.\\
7041   If~you~go~on,~this~key~will~be~ignored.
7042 }
7043 \@@_msg_new:nn { enumitem~not~loaded }
7044 {
7045   You~can't~use~the~command~\token_to_str:N\tabularnote\
7046   ~because~you~haven't~loaded~'enumitem'.\\
7047   If~you~go~on,~this~command~will~be~ignored.
7048 }

```

```

7049 \@@_msg_new:nn { Wrong-last-row }
7050 {
7051   You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
7052   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
7053   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
7054   last-row.~You~can~avoid~this~problem~by~using~'last-row'~
7055   without~value~(more~compilations~might~be~necessary).
7056 }
7057 \@@_msg_new:nn { Yet-in-env }
7058 { Environments~of~nicematrix~can't~be~nested.\\ This~error~is~fatal. }
7059 \@@_msg_new:nn { Outside-math-mode }
7060 {
7061   The~\@@_full_name_env:\ can~be~used~only~in~math-mode~
7062   (and~not~in~\token_to_str:N \vcenter).\\
7063   This~error~is~fatal.
7064 }
7065 \@@_msg_new:nn { One-letter-allowed }
7066 {
7067   The~value~of~key~'\l_keys_key_str'~must~be~of~length~1.\\
7068   If~you~go~on,~it~will~be~ignored.
7069 }
7070 \@@_msg_new:nnn { Unknown~key~for~Block }
7071 {
7072   The~key~'\l_keys_key_str'~is~unknown~for~the~command~\token_to_str:N
7073   \Block.\\ If~you~go~on,~it~will~be~ignored. \\
7074   For~a~list~of~the~available~keys,~type~H~<return>.
7075 }
7076 {
7077   The~available~keys~are~(in~alphabetic~order):~b,~borders,~c,~draw,~fill,~
7078   hvlines,~l,~line-width,~rounded-corners,~r,~t~and~tikz.
7079 }
7080 \@@_msg_new:nnn { Unknown~key~for~CodeAfter }
7081 {
7082   The~key~'\l_keys_key_str'~is~unknown.\\
7083   If~you~go~on,~it~will~be~ignored. \\
7084   For~a~list~of~the~available~keys~in~\token_to_str:N
7085   \CodeAfter,~type~H~<return>.
7086 }
7087 {
7088   The~available~keys~are~(in~alphabetic~order):~
7089   delimiters/color,~
7090   rules~(with~the~subkeys~'color'~and~'width'),~
7091   sub-matrix~(several~subkeys)~
7092   and~xdots~(several~subkeys).~
7093   The~latter~is~for~the~command~\token_to_str:N \line.
7094 }
7095 \@@_msg_new:nnn { Unknown~key~for~SubMatrix }
7096 {
7097   The~key~'\l_keys_key_str'~is~unknown.\\
7098   If~you~go~on,~this~key~will~be~ignored. \\
7099   For~a~list~of~the~available~keys~in~\token_to_str:N
7100   \SubMatrix,~type~H~<return>.
7101 }
7102 {
7103   The~available~keys~are~(in~alphabetic~order):~
7104   'delimiters/color',~
7105   'extra-height',~
7106   'hlines',~
7107   'hvlines',~
7108   'left-xshift',~
7109   'name',~

```

```

7110     'right-xshift',~
7111     'rules'~(with~the~subkeys~'color'~and~'width'),~
7112     'slim',~
7113     'vlines'~and~'xshift'~(which~sets~both~'left-xshift'~
7114     and~'right-xshift').\\
7115 }
7116 \\@@_msg_new:nnn { Unknown~key~for~notes }
7117 {
7118     The~key~'\l_keys_key_str'~is~unknown.\\
7119     If~you~go~on,~it~will~be~ignored. \\
7120     For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
7121 }
7122 {
7123     The~available~keys~are~(in~alphabetic~order):~
7124     bottomrule,~
7125     code-after,~
7126     code-before,~
7127     enumitem-keys,~
7128     enumitem-keys-para,~
7129     para,~
7130     label-in-list,~
7131     label-in-tabular~and~
7132     style.
7133 }
7134 \\@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
7135 {
7136     The~key~'\l_keys_key_str'~is~unknown~for~the~command~
7137     \token_to_str:N \NiceMatrixOptions. \\
7138     If~you~go~on,~it~will~be~ignored. \\
7139     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
7140 }
7141 {
7142     The~available~keys~are~(in~alphabetic~order):~
7143     allow-duplicate-names,~
7144     cell-space-bottom-limit,~
7145     cell-space-limits,~
7146     cell-space-top-limit,~
7147     code-for-first-col,~
7148     code-for-first-row,~
7149     code-for-last-col,~
7150     code-for-last-row,~
7151     corners,~
7152     create-extra-nodes,~
7153     create-medium-nodes,~
7154     create-large-nodes,~
7155     delimiters~(several~subkeys),~
7156     end-of-row,~
7157     first-col,~
7158     first-row,~
7159     hlines,~
7160     hvlines,~
7161     last-col,~
7162     last-row,~
7163     left-margin,~
7164     letter-for-dotted-lines,~
7165     light-syntax,~
7166     notes~(several~subkeys),~
7167     nullify-dots,~
7168     renew-dots,~
7169     renew-matrix,~
7170     right-margin,~
7171     rules~(with~the~subkeys~'color'~and~'width'),~
7172     small,~

```

```

7173     sub-matrix~(several~subkeys),
7174     vlines,~
7175     xdots~(several~subkeys).
7176 }

7177 \@@_msg_new:nnn { Unknown-key~for~NiceArray }
7178 {
7179     The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
7180     \{NiceArray\}. \\
7181     If~you~go~on,~it~will~be~ignored. \\
7182     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
7183 }
7184 {
7185     The~available~keys~are~(in~alphabetic~order):~
7186     b,~
7187     baseline,~
7188     c,~
7189     cell-space-bottom-limit,~
7190     cell-space-limits,~
7191     cell-space-top-limit,~
7192     code-after,~
7193     code-for-first-col,~
7194     code-for-first-row,~
7195     code-for-last-col,~
7196     code-for-last-row,~
7197     colortbl-like,~
7198     columns-width,~
7199     corners,~
7200     create-extra-nodes,~
7201     create-medium-nodes,~
7202     create-large-nodes,~
7203     delimiters/color,~
7204     extra-left-margin,~
7205     extra-right-margin,~
7206     first-col,~
7207     first-row,~
7208     hlines,~
7209     hvlines,~
7210     last-col,~
7211     last-row,~
7212     left-margin,~
7213     light-syntax,~
7214     name,~
7215     notes/bottomrule,~
7216     notes/para,~
7217     nullify-dots,~
7218     renew-dots,~
7219     right-margin,~
7220     rules~(with~the~subkeys~'color'~and~'width'),~
7221     small,~
7222     t,~
7223     tabularnote,~
7224     vlines,~
7225     xdots/color,~
7226     xdots/shorten~and~
7227     xdots/line-style.
7228 }

```

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).

```

7229 \@@_msg_new:nnn { Unknown-key~for~NiceMatrix }
7230 {
7231     The~key~'\l_keys_key_str'~is~unknown~for~the~
7232     \@@_full_name_env:. \\

```

```

7233 If~you~go~on,~it~will~be~ignored. \\
7234 For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
7235 }
7236 {
7237 The~available~keys~are~(in~alphabetic~order):~
7238 b,~
7239 baseline,~
7240 c,~
7241 cell-space-bottom-limit,~
7242 cell-space-limits,~
7243 cell-space-top-limit,~
7244 code-after,~
7245 code-for-first-col,~
7246 code-for-first-row,~
7247 code-for-last-col,~
7248 code-for-last-row,~
7249 colortbl-like,~
7250 columns-width,~
7251 corners,~
7252 create-extra-nodes,~
7253 create-medium-nodes,~
7254 create-large-nodes,~
7255 delimiters~(several~subkeys),~
7256 extra-left-margin,~
7257 extra-right-margin,~
7258 first-col,~
7259 first-row,~
7260 hlines,~
7261 hvlines,~
7262 l,~
7263 last-col,~
7264 last-row,~
7265 left-margin,~
7266 light-syntax,~
7267 name,~
7268 nullify-dots,~
7269 r,~
7270 renew-dots,~
7271 right-margin,~
7272 rules~(with~the~subkeys~'color'~and~'width'),~
7273 small,~
7274 t,~
7275 vlines,~
7276 xdots/color,~
7277 xdots/shorten~and~
7278 xdots/line-style.
7279 }
7280 \@_msg_new:nnn { Unknown~key~for~NiceTabular }
7281 {
7282 The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
7283 \{NiceTabular\}. \\
7284 If~you~go~on,~it~will~be~ignored. \\
7285 For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
7286 }
7287 {
7288 The~available~keys~are~(in~alphabetic~order):~
7289 b,~
7290 baseline,~
7291 c,~
7292 cell-space-bottom-limit,~
7293 cell-space-limits,~
7294 cell-space-top-limit,~
7295 code-after,~

```

```

7296 code-for-first-col,~
7297 code-for-first-row,~
7298 code-for-last-col,~
7299 code-for-last-row,~
7300 colortbl-like,~
7301 columns-width,~
7302 corners,~
7303 create-extra-nodes,~
7304 create-medium-nodes,~
7305 create-large-nodes,~
7306 extra-left-margin,~
7307 extra-right-margin,~
7308 first-col,~
7309 first-row,~
7310 hlines,~
7311 hvlines,~
7312 last-col,~
7313 last-row,~
7314 left-margin,~
7315 light-syntax,~
7316 name,~
7317 notes/bottomrule,~
7318 notes/para,~
7319 nullify-dots,~
7320 renew-dots,~
7321 right-margin,~
7322 rules~(with~the~subkeys~'color'~and~'width'),~
7323 t,~
7324 tabularnote,~
7325 vlines,~
7326 xdots/color,~
7327 xdots/shorten~and~
7328 xdots/line-style.
7329 }

7330 \@@_msg_new:nnn { Duplicate-name }
7331 {
7332   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
7333   the~same~environment~name~twice.~You~can~go~on,~but,~
7334   maybe,~you~will~have~incorrect~results~especially~
7335   if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
7336   message~again,~use~the~key~'allow-duplicate-names'~in~
7337   '\token_to_str:N \NiceMatrixOptions'.\\
7338   For~a~list~of~the~names~already~used,~type~H~<return>. \\
7339 }
7340 {
7341   The~names~already~defined~in~this~document~are:~
7342   \seq_use:Nnnn \g_@@_names_seq { ~and~ } { ,~ } { ~and~ }.
7343 }

7344 \@@_msg_new:nn { Option-auto-for-columns-width }
7345 {
7346   You~can't~give~the~value~'auto'~to~the~key~'columns-width'~here.~
7347   If~you~go~on,~the~key~will~be~ignored.
7348 }

```



## 19 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>67</sup>, Tikz externalization is now deactivated in the environments of the package `nicematrix`.<sup>68</sup>

---

<sup>67</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>68</sup>Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it's not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

## Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\begin{pmatrix} & C_j & \\ 0 & \vdots & 0 \\ & a \cdots & \\ 0 & & 0 \end{pmatrix} L_i$$

## Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See [www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)

## 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>69</sup>, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

## Changes between version 3.4 and 3.5

Correction on a bug on the two previous versions where the `code-after` was not executed.

## Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate-name` in the environments of `amsmath`.

## Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

---

<sup>69</sup>cf. [tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize](https://tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize)

## Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`. An error is raised when an obsolete environment is used.

## Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.  
New options `create-medium-nodes` and `create-large-nodes`.

## Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).  
New option `dotted-lines-margin` for fine tuning of the dotted lines.

## Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

## Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.  
Options `vlines`, `hlines` and `hvlines`.  
Option `baseline` pour `{NiceArray}` (not for the other environments).  
The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell  $i-j$ , the name is  $i-j$ -block and, if the creation of the “medium nodes” is required, a node  $i-j$ -block-medium is created.  
If the user 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](https://stackoverflow.com)).  
Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.

## Changes between versions 3.14 and 3.15

It's possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the code-after with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New 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 `hylvline` 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.

The version 5.15b is compatible with the version 3.0+ of `siunitx` (previous versions were not).

## Changes between versions 5.15 and 5.16

It's now possible to use the cells corresponding to the contents of the nodes (of the form `i-j`) in the `\CodeBefore` when the key `create-cell-nodes` of that `\CodeBefore` is used. The medium and the large nodes are also available if the corresponding keys are used.

## Changes between versions 5.16 and 5.17

The key `define-L-C-R` (only available at load-time) now raises a (non fatal) error.

Keys `L`, `C` and `R` for the command `\Block`.

Key `hvlines-except-borders`.

It's now possible to use a key `l`, `r` or `c` with the command `\pAutoNiceMatrix` (and the similar ones).

## Changes between versions 5.17 and 5.18

New command `\RowStyle`

## Changes between versions 5.18 and 5.19

New key `tikz` for the command `\Block`.

## Changes between versions 5.19 and 6.0

Columns X and environment {NiceTabularX}.

Command \rowlistcolors available in the \CodeBefore.

In columns with fixed width, the blocks are composed as paragraphs (wrapping of the lines).

The key define-L-C-R has been deleted.

## Changes between versions 6.0 and 6.1

Better computation of the widths of the X columns.

Key \color for the command \RowStyle.

# 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: .....	1210, 5389
\@@_Block_i .....	5394, 5395, 5399
\@@_Block_ii:nnnnn .....	5399, 5400
\@@_Block_iv:nnnnn .....	5437, 5441
\@@_Block_iv:nnnnnn .....	5653, 5655
\@@_Block_v:nnnnn .....	5438, 5560
\@@_Block_v:nnnnnn .....	5682, 5685
\@@_Cdots .....	1133, 1201, 3933
\g_@@_Cdots_lines_tl .....	1231, 3075
\@@_CodeAfter: .....	1214, 6236
\@@_CodeAfter_i:n .....	855, 2753, 2798, 6236, 6237, 6247
\@@_CodeAfter_ii:n .....	6240, 6242
\@@_CodeAfter_keys: .....	3015, 3036
\@@_CodeBefore:w .....	1412, 1414
\@@_CodeBefore_keys: .....	1392, 1409
\@@_Ddots .....	1135, 1203, 3965
\g_@@_Ddots_lines_tl .....	1234, 3073
\g_@@_HVdotsfor_lines_tl .....	1236, 3071, 4049, 4126, 6785
\@@_Hdotsfor: .....	1138, 1208, 4025
\@@_Hdotsfor:nnnn .....	4051, 4064
\@@_Hdotsfor_i .....	4034, 4040, 4047
\@@_Hline: .....	1206, 4883
\@@_Hline_i:n .....	4883, 4884, 4890
\@@_Hline_ii:nn .....	4887, 4890
\@@_Hline_iii:n .....	4888, 4891
\@@_Hspace: .....	1207, 4019
\@@_Iddots .....	1136, 1204, 3989
\g_@@_Iddots_lines_tl .....	1235, 3074
\@@_Ldots .....	1132, 1137, 1200, 3917
\g_@@_Ldots_lines_tl .....	1232, 3076
\l_@@_Matrix_bool .....	245, 1630, 1654, 2359, 2697, 2701, 2709, 2866
\l_@@_NiceArray_bool .....	242, 376, 1276, 1498, 1570, 1692, 1699, 1711, 2359, 2685, 2697, 2701, 2709, 2842, 4733, 4737, 4869, 4873, 5090, 5097, 6154
\g_@@_NiceMatrixBlock_int .....	229, 5149, 5154, 5157, 5168
\l_@@_NiceTabular_bool .....	168, 243, 860, 1059, 1391, 1394, 1465, 1603, 1607, 1700, 1712, 2686, 2897, 2907, 2916, 3014, 3017, 5470, 5569, 6162
\@@_NotEmpty: .....	1216, 2889
\@@_OnlyMainNiceMatrix:n .....	1212, 4588
\@@_OnlyMainNiceMatrix_i:n .....	4591, 4598, 4601
\@@_RowStyle:n .....	1217, 1335
\@@_S: .....	193, 1740
\@@_SubMatrix .....	2996, 6386
\@@_SubMatrix_in_code_before .....	1384, 6366
\@@_Vdots .....	1134, 1202, 3949
\g_@@_Vdots_lines_tl .....	1233, 3072
\@@_Vdotsfor: .....	1209, 4124
\@@_Vdotsfor:nnnn .....	4128, 4139
\@@_W: .....	1657, 1739, 2210
\l_@@_X_column_bool .....	247, 2111, 5435
\l_@@_X_columns_aux_bool .....	274, 1523, 2104
\l_@@_X_columns_dim .....	275, 1524, 1533, 1539, 2105
\@@_actually_color: .....	1393, 4265
\@@_actually_diagbox:nnnnnn .....	5448, 5754, 6174, 6190
\@@_actually_draw_Cdots: .....	3489, 3493
\@@_actually_draw_Ddots: .....	3639, 3643
\@@_actually_draw_Iddots: .....	3691, 3695
\@@_actually_draw_Ldots: .....	3450, 3454, 4115
\@@_actually_draw_Vdots: .....	3571, 3575, 4190
\@@_adapt_S_column: .....	201, 206, 219, 1464
\@@_add_to_colors_seq:nn .....	4251, 4263, 4264, 4293, 4302, 4311, 4320, 4427
\@@_adjust_pos_of_blocks_seq: .....	2983, 3038
\@@_adjust_pos_of_blocks_seq_i:nnnn .....	3041, 3043
\@@_adjust_size_box: .....	933, 960, 1956, 2271, 2777, 2822
\@@_adjust_to_submatrix:nn .....	3334, 3437, 3476, 3557, 3632, 3684
\@@_adjust_to_submatrix:nnnnnn .....	3341, 3343
\@@_after_array: .....	1639, 2920
\g_@@_after_col_zero_bool .....	276, 1100, 2754, 4031
\@@_analyze_end:Nn .....	2507, 2552



<code>\l_@@_argspec_tl</code>	3915, 3916, 3917, 3933, 3949, 3965, 3989, 4045, 4046, 4047, 4122, 4123, 4124, 4200, 4201, 4202, 6384, 6385, 6386	<code>\l_@@_code_before_bool</code>	..... 280, 615, 642, 1074, 1267, 1298, 1495, 2566, 2583, 2601, 2632, 2658, 2692, 2729, 3029, 3031
<code>\@@_array:</code>	..... 1054, 2508, 2535	<code>\g_@@_code_before_tl</code>	. 1484, 1493, 1496, 3026
<code>\@@_arraycolor</code>	..... 1381, 4357	<code>\l_@@_code_before_tl</code>	..... ..... 279, 614, 1297, 1392, 1496
<code>\c_@@_arydshln_loaded_bool</code>	... 24, 31, 1765	<code>\l_@@_code_for_first_col_tl</code>	..... 549, 2767
<code>\l_@@_auto_columns_width_bool</code>	..... ..... 479, 625, 2619, 2623, 5144	<code>\l_@@_code_for_first_row_tl</code>	. 553, 870, 5841
<code>\g_@@_aux_found_bool</code>	..... ..... 1248, 1253, 1416, 1486, 1489	<code>\l_@@_code_for_last_col_tl</code>	..... 551, 2811
<code>\g_@@_aux_tl</code>	..... 248, 1492, 1521, 1646, 2929, 2943, 2951, 3024, 4966	<code>\l_@@_code_for_last_row_tl</code>	. 555, 877, 5844
<code>\l_@@_bar_at_end_of_pream_bool</code>	..... ..... 1693, 1825, 2689	<code>\l_@@_code_tl</code>	..... 268, 6361, 6583
<code>\l_@@_baseline_tl</code>	470, 471, 618, 619, 620, 621, 1067, 1572, 2312, 2324, 2329, 2331, 2336, 2341, 2431, 2432, 2436, 2441, 2443, 2448	<code>\l_@@_col_max_int</code>	..... ..... 303, 3201, 3212, 3280, 3339, 3356
<code>\@@_begin_of_NiceMatrix:nn</code>	... 2864, 2885	<code>\l_@@_col_min_int</code>	..... ..... 302, 3206, 3269, 3274, 3337, 3354
<code>\@@_begin_of_row:</code>	... 858, 882, 1160, 2755	<code>\g_@@_col_total_int</code>	234, 977, 1227, 1347, 1358, 1429, 1556, 2191, 2192, 2651, 2652, 2732, 2736, 2741, 2742, 2801, 2924, 2926, 2938, 3100, 3524, 3542, 3602, 4185, 4578, 5195, 5205, 5239, 5326, 5665, 5879, 6408, 6457
<code>\l_@@_block_auto_columns_width_bool</code>	.. ..... 1478, 2624, 5137, 5142, 5152, 5162	<code>\l_@@_col_width_dim</code>	..... ..... 231, 232, 1896, 1945, 5474, 5477
<code>\g_@@_block_box_int</code>	..... 320, 1458, 5443, 5457, 5459, 5513, 5525, 5537, 5546, 5556	<code>\@@_color_index:n</code>	..... 4417, 4438, 4441
<code>\@@_block_tikz:nnnn</code>	..... 5742, 6064	<code>\l_@@_color_int</code>	..... ..... 4381, 4382, 4399, 4400, 4420, 4431
<code>\g_@@_blocks_dp_dim</code>	..... ..... 239, 941, 944, 945, 5540, 5543	<code>\l_@@_color_tl</code>	..... ..... 311, 4414, 4415, 4425, 4428, 5386, 5461, 5463
<code>\g_@@_blocks_ht_dim</code>	..... ..... 238, 947, 950, 951, 5531, 5534	<code>\g_@@_colors_seq</code>	1388, 4254, 4258, 4259, 4269
<code>\g_@@_blocks_seq</code>	..... ..... 291, 1480, 2369, 5550, 5562, 5653	<code>\l_@@_colors_seq</code>	4376, 4377, 4421, 4440, 4442
<code>\g_@@_blocks_wd_dim</code>	..... ..... 237, 935, 938, 939, 5519, 5522	<code>\@@_colortbl_like:</code>	..... 1140, 1218
<code>\c_@@_booktabs_loaded_bool</code>	25, 34, 1149, 2402	<code>\l_@@_colortbl_like_bool</code>	456, 641, 1218, 1681
<code>\l_@@_borders_clist</code>	..... 309, 5623, 5715, 6006, 6022, 6024, 6026, 6028, 6047, 6059	<code>\c_@@_colortbl_loaded_bool</code>	. 100, 104, 1168
<code>\@@_cartesian_color:nn</code>	4283, 4295, 4304, 4429	<code>\l_@@_cols_tl</code>	..... ..... 4286, 4334, 4368, 4378, 4379, 4429, 4476, 4479
<code>\@@_cartesian_path:</code>	..... 4287, 4528	<code>\g_@@_cols_vlism_seq</code>	. 256, 1676, 1756, 6485
<code>\@@_cartesian_path:n</code>	..... 4335, 4470, 4528	<code>\@@_columncolor</code>	..... 1382, 4298
<code>\@@_cell_begin:w</code>	..... 852, 1779, 1897, 1947, 1980, 1996, 2110, 2220, 2241, 2262, 2875	<code>\@@_columncolor_preamble</code>	..... 1144, 4576
<code>\l_@@_cell_box</code>	..... 859, 907, 909, 915, 921, 924, 928, 937, 938, 943, 944, 949, 950, 961, 962, 963, 964, 966, 969, 973, 975, 994, 1009, 1011, 1018, 1019, 1032, 1151, 1261, 1263, 1924, 1928, 1932, 1935, 1946, 1957, 2113, 2261, 2272, 2756, 2780, 2783, 2785, 2802, 2825, 2829, 5762, 5872, 5909, 6169	<code>\c_@@_columncolor_regex</code>	..... 59, 1684
<code>\@@_cell_end:</code>	..... 954, 1781, 1915, 1952, 1985, 1998, 2119, 2222, 2251, 2267, 2875	<code>\l_@@_columns_width_dim</code>	..... ..... 230, 626, 768, 2620, 2626, 5150, 5156
<code>\l_@@_cell_space_bottom_limit_dim</code>	.. ..... 459, 532, 964, 1317	<code>\g_@@_com_or_env_str</code>	..... 260, 263
<code>\l_@@_cell_space_top_limit_dim</code>	..... ..... 458, 530, 962, 1308	<code>\@@_computations_for_large_nodes:</code>	... ..... 5266, 5279, 5284
<code>\@@_cellcolor</code>	..... 1375, 4337, 4349, 4350	<code>\@@_computations_for_medium_nodes:</code>	.. ..... 5186, 5255, 5265, 5276
<code>\@@_cellcolor_tabular</code>	..... 1142, 4553	<code>\@@_compute_a_corner:nnnnn</code>	..... ..... 4954, 4956, 4958, 4960, 4973
<code>\g_@@_cells_seq</code>	..... 2546, 2547, 2548, 2550	<code>\@@_compute_corners:</code>	..... 2982, 4946
<code>\@@_center_cell_box:</code>	..... 1884, 1919	<code>\@@_construct_preamble:</code>	..... 1273, 1651
<code>\@@_chessboardcolors</code>	..... 1383, 4342	<code>\l_@@_corners_cells_seq</code>	..... ..... 295, 2988, 4473, 4513, 4664, 4670, 4677, 4802, 4808, 4815, 4948, 4964, 4968, 4969, 5029
<code>\@@_cline</code>	..... 147, 1199	<code>\l_@@_corners_clist</code>	..... ..... 475, 603, 608, 4631, 4769, 4949
<code>\@@_cline_i:nn</code>	..... 148, 149, 161, 164	<code>\@@_create_col_nodes:</code>	... 2511, 2539, 2558
<code>\@@_cline_i:w</code>	..... 149, 150	<code>\@@_create_diag_nodes:</code>	... 1364, 2959, 3142
		<code>\@@_create_extra_nodes:</code>	..... ..... 1447, 2368, 3134, 5176
		<code>\@@_create_large_nodes:</code>	..... 5184, 5260
		<code>\@@_create_medium_and_large_nodes:</code>	.. ..... 5181, 5271
		<code>\@@_create_medium_nodes:</code>	..... 5182, 5250
		<code>\@@_create_nodes:</code>	5257, 5268, 5278, 5281, 5322

<code>\@@_create_row_node:</code>	.....	1070, 1103, 1150	<code>\c_@@_enumitem_loaded_bool</code>	.....	26, 37, 349, 669, 674, 685, 690
<code>\@@_cut_on_hyphen:w</code>	.....	4278, 4327, 4332, 4395, 4483, 4484, 4506, 4507, 4537, 4538, 5942, 5951, 5982, 5985, 6011, 6014, 6370, 6375, 6400, 6403	<code>\@@_env:</code>	.....	224, 228, 892, 898, 995, 1001, 1015, 1023, 1079, 1085, 1091, 1163, 1348, 1349, 1354, 1355, 1360, 1361, 1372, 1426, 1427, 1432, 1435, 1438, 1441, 2567, 2570, 2572, 2588, 2594, 2597, 2606, 2612, 2615, 2637, 2643, 2646, 2663, 2669, 2675, 2699, 2707, 2721, 2732, 2736, 2742, 3005, 3102, 3106, 3112, 3119, 3125, 3159, 3161, 3168, 3170, 3171, 3176, 3179, 3241, 3309, 3373, 3384, 3397, 3400, 3419, 3422, 3527, 3530, 3545, 3548, 4078, 4096, 4153, 4171, 4222, 4224, 4243, 4246, 4984, 5003, 5021, 5208, 5210, 5218, 5329, 5338, 5356, 5777, 5786, 5790, 5804, 5809, 5821, 5827, 5828, 5831, 5849, 5886, 6265, 6268, 6424, 6426, 6431, 6433, 6460, 6462, 6467, 6469, 6567, 6579
<code>\g_@@_ddots_int</code>	.....	2962, 3663, 3664	<code>\g_@@_env_int</code>	.....	223, 224, 226, 1477, 1487, 1490, 1645, 5365
<code>\@@_def_env:nnn</code>	.....	2848, 2859, 2860, 2861, 2862, 2863	<code>\@@_error:n</code>	.....	12, 352, 377, 502, 512, 565, 695, 751, 761, 767, 776, 784, 802, 809, 817, 818, 819, 825, 830, 831, 832, 846, 848, 849, 850, 1333, 1407, 1517, 1551, 1561, 1633, 2346, 2407, 2453, 4356, 4371, 5613, 5648, 6004, 6234, 6346, 6357, 6364, 6409, 6708, 6713, 6740, 6750
<code>\@@_define_com:nnn</code>	.....	6138, 6146, 6147, 6148, 6149, 6150	<code>\@@_error:nn</code>	.....	13, 633, 2029, 2075, 2081, 3920, 3923, 3936, 3939, 3952, 3955, 3969, 3970, 3975, 3976, 3993, 3994, 3999, 4000, 4962, 6009, 6351, 6439, 6442
<code>\@@_delimiter:nnn</code>	.....	2026, 2047, 2055, 2068, 2074, 2080, 6250	<code>\@@_error:nnn</code>	.....	14, 4227, 6514, 6549
<code>\l_@@_delimiters_color_tl</code>	.....	492, 701, 1405, 1595, 1596, 1613, 1614, 6230, 6288, 6289, 6316, 6635, 6636, 6659, 6660	<code>\@@_error_too_much_cols:</code>	.....	1721, 6770
<code>\l_@@_delimiters_max_width_bool</code>	.....	493, 699, 1618	<code>\@@_everycr:</code>	.....	1096, 1173, 1176
<code>\g_@@_delta_x_one_dim</code>	....	2964, 3666, 3676	<code>\@@_everycr_i:</code>	.....	1096, 1097
<code>\g_@@_delta_x_two_dim</code>	....	2966, 3714, 3724	<code>\l_@@_except_borders_bool</code>	.....	484, 577, 4733, 4737, 4869, 4873
<code>\g_@@_delta_y_one_dim</code>	....	2965, 3668, 3676	<code>\@@_exec_code_before:</code>	.....	1267, 1386
<code>\g_@@_delta_y_two_dim</code>	....	2967, 3716, 3724	<code>\@@_expand_clist:NN</code>	.....	4476, 4477, 4529
<code>\@@_diagbox:nn</code>	.....	1215, 6170	<code>\l_@@_exterior_arraycolsep_bool</code>	.....	472, 764, 1702, 1714, 2688
<code>\@@_dotfill:</code>	.....	6159	<code>\l_@@_extra_left_margin_dim</code>	.....	487, 593, 1289, 2788
<code>\@@_dotfill_i:</code>	.....	6164, 6166	<code>\l_@@_extra_right_margin_dim</code>	.....	488, 594, 1512, 2833, 3605
<code>\@@_dotfill_ii:</code>	.....	6163, 6166, 6167	<code>\@@_extract_brackets</code>	.....	5727, 5914
<code>\@@_dotfill_iii:</code>	.....	6167, 6168	<code>\@@_extract_coords_values:</code>	....	5347, 5354
<code>\@@_double_int_eval:n</code>	....	4196, 4210, 4211	<code>\@@_fatal:n</code>	....	15, 252, 1468, 2007, 2036, 2516, 2520, 2522, 2555, 4036, 6775, 6778, 6781
<code>\g_@@_dp_ante_last_row_dim</code>	....	886, 1184	<code>\@@_fatal:nn</code>	.....	16, 1770, 2214
<code>\g_@@_dp_last_row_dim</code>	....	886, 887, 1187, 1188, 1262, 1263, 1589	<code>\l_@@_fill_tl</code>	.....	306, 5615, 5725, 5727
<code>\g_@@_dp_row_zero_dim</code>	....	906, 907, 1178, 1179, 1582, 2425, 2464	<code>\l_@@_final_i_int</code>	.....	2971, 3188, 3193, 3196, 3221, 3229, 3233, 3242, 3250, 3330, 3385, 3466, 3539, 3545, 3548, 4069, 4097, 4165, 4175, 4177
<code>\@@_draw_Cdots:nnn</code>	.....	3474	<code>\l_@@_final_j_int</code>	.....	2972, 3189, 3194, 3201, 3206, 3212, 3222, 3230, 3234, 3243, 3251, 3331, 3386, 3419, 3422, 3430, 3656, 4090, 4100, 4102, 4144, 4173
<code>\@@_draw_Ddots:nnn</code>	.....	3630	<code>\l_@@_final_open_bool</code>	.....	2974, 3195, 3199, 3202, 3209, 3215, 3219, 3235, 3463, 3498, 3503, 3514, 3578, 3588, 3593, 3614,
<code>\@@_draw_Iddots:nnn</code>	.....	3682			
<code>\@@_draw_Ldots:nnn</code>	.....	3435			
<code>\@@_draw_Vdots:nnn</code>	.....	3555			
<code>\@@_draw_blocks:</code>	.....	2369, 5650			
<code>\@@_draw_dotted_lines:</code>	.....	2981, 3060			
<code>\@@_draw_dotted_lines_i:</code>	.....	3063, 3067			
<code>\l_@@_draw_first_bool</code>	....	318, 3980, 4004, 4015			
<code>\@@_draw_hlines:</code>	.....	2994, 4865			
<code>\@@_draw_line:</code>	.....	3472, 3517, 3628, 3680, 3728, 3730, 4249, 5105, 5135			
<code>\@@_draw_line_ii:nn</code>	.....	4229, 4233			
<code>\@@_draw_line_iii:nn</code>	.....	4236, 4240			
<code>\@@_draw_non_standard_dotted_line:</code>	....	3736, 3738			
<code>\@@_draw_non_standard_dotted_line:n</code>	....	3741, 3744			
<code>\@@_draw_non_standard_dotted_line:nnn</code>	.....	3746, 3751, 3765			
<code>\@@_draw_standard_dotted_line:</code>	....	3735, 3766			
<code>\@@_draw_standard_dotted_line_i:</code>	....	3829, 3833			
<code>\l_@@_draw_tl</code>	.....	307, 5617, 5621, 5703, 5924, 5930, 5932, 5934, 5971, 5972			
<code>\@@_draw_vlines:</code>	.....	2995, 4729			
<code>\g_@@_empty_cell_bool</code>	....	288, 968, 978, 2790, 2837, 3931, 3947, 3963, 3987, 4010, 4021			
<code>\l_@@_end_of_row_tl</code>	.....	489, 490, 543, 2531, 2532, 6889			
<code>\c_@@_endpgfortikzpicture_tl</code>	.....	43, 47, 3064, 4237, 5076			

```

3653, 3703, 3837, 3852, 3883, 3884, 4067,
4091, 4103, 4142, 4166, 4178, 4219, 5073, 5110
\@@_find_extremities_of_line:nnnn ...
..... 3183, 3440, 3479, 3560, 3635, 3687
\l_@@_first_col_int .....
..... 135, 148, 330, 331, 545, 823, 858,
1358, 1429, 1565, 1694, 2561, 2581, 2936,
3100, 3524, 3542, 4029, 4497, 4590, 5195,
5205, 5239, 5287, 5326, 6119, 6125, 6131, 6457
\l_@@_first_i_tl 6396, 6401, 6420, 6451,
6460, 6462, 6517, 6524, 6526, 6608, 6619, 6623
\l_@@_first_j_tl ..... 6397, 6402, 6424,
6426, 6487, 6500, 6507, 6509, 6609, 6620, 6624
\l_@@_first_row_int .....
..... 328, 329, 546, 827, 1225,
1352, 1424, 1580, 2343, 2422, 2450, 2461,
2933, 3098, 3394, 3416, 5188, 5202, 5229,
5286, 5324, 5783, 5801, 6117, 6262, 6421, 6900
\c_@@_footnote_bool .....
1453, 1649, 6678, 6711, 6734, 6737, 6747, 6753
\c_@@_footnotehyper_bool . 6677, 6712, 6744
\@@_full_name_env: .....
261, 6759, 6791, 6798, 6806, 6814, 6818,
6914, 6932, 6935, 6948, 6953, 6958, 6960,
6990, 7009, 7014, 7019, 7026, 7052, 7061, 7232
\@@_hdottedline: ..... 1205, 5058
\@@_hdottedline:n ..... 5066, 5070
\@@_hdottedline_i: ..... 5061, 5063
\@@_hdottedline_i:n ..... 5075, 5079
\@@_hline:nnnn ..... 4747, 4880, 4899, 5996
\@@_hline_i:nn ..... 2992
\@@_hline_i:nnnn ..... 4750, 4753
\@@_hline_i_complete:nn ..... 2992, 4863
\@@_hline_ii:nnnn ... 4778, 4789, 4822, 4864
\l_@@_hlines_clist ..... 324, 557, 571,
576, 605, 1104, 1106, 1110, 2994, 4878, 4879
\l_@@_hpos_block_str ... 313, 314, 5370,
5372, 5374, 5376, 5378, 5380, 5415, 5416,
5418, 5469, 5478, 5491, 5502, 5553, 5576,
5580, 5592, 5597, 5629, 5631, 5633, 5635,
5638, 5641, 5853, 5865, 5876, 5880, 5890, 5902
\l_@@_hpos_cell_str ..... 235, 236,
1779, 1872, 1874, 1948, 2220, 2263, 5414, 5416
\l_@@_hpos_col_str ... 1828, 1830, 1832,
1834, 1859, 1871, 1875, 1877, 1885, 1886, 2097
\l_@@_hpos_of_block_cap_bool .....
.... 315, 5636, 5639, 5642, 5780, 5850, 5887
\g_@@_ht_last_row_dim .....
..... 888, 1185, 1186, 1260, 1261, 1588
\g_@@_ht_row_one_dim .. 914, 915, 1182, 1183
\g_@@_ht_row_zero_dim .....
.... 908, 909, 1180, 1181, 1583, 2424, 2463
\@@_hvlines_block:nnn ..... 5693, 5978
\l_@@_hvlines_block_bool .. 319, 5625, 5689
\@@_i: ..... 5188, 5190,
5191, 5192, 5193, 5202, 5208, 5210, 5211,
5212, 5213, 5218, 5219, 5220, 5221, 5229,
5232, 5234, 5235, 5236, 5288, 5290, 5293,
5294, 5298, 5299, 5324, 5329, 5331, 5333,
5337, 5338, 5349, 5356, 5358, 5360, 5364, 5365
\g_@@_iddots_int ..... 2963, 3711, 3712
\l_@@_in_env_bool .... 241, 376, 1468, 1469
\c_@@_in_preamble_bool . 21, 22, 23, 665, 681
\l_@@_initial_i_int ..... 2969,
3186, 3261, 3264, 3289, 3297, 3301, 3310,
3318, 3328, 3374, 3459, 3505, 3507, 3521,
3527, 3530, 4068, 4069, 4079, 4147, 4157, 4159
\l_@@_initial_j_int .....
..... 2970, 3187, 3262, 3269,
3274, 3280, 3290, 3298, 3302, 3311, 3319,
3329, 3375, 3397, 3400, 3408, 3595, 3597,
3602, 3648, 4072, 4082, 4084, 4143, 4144, 4155
\l_@@_initial_open_bool .....
.... 2973, 3263, 3267, 3270, 3277, 3283,
3287, 3303, 3456, 3495, 3502, 3512, 3578,
3585, 3591, 3645, 3697, 3835, 3882, 4066,
4073, 4085, 4141, 4148, 4160, 4218, 5072, 5109
\@@_insert_tabularnotes: ..... 2374, 2377
\@@_instruction_of_type:nnn .....
..... 1035, 3925, 3941, 3957, 3980, 4004
\c_@@_integers_alist_tl ..... 6593, 6604
\l_@@_inter_dots_dim .....
. 460, 461, 2978, 3840, 3847, 3858, 3866,
3873, 3878, 3890, 3898, 5101, 5103, 5131, 5133
\g_@@_internal_code_after_tl 269, 1815,
2025, 2046, 2054, 2067, 2073, 2079, 2130,
3010, 3011, 4898, 5065, 5446, 5752, 6172, 6378
\@@_intersect_our_row:nnnn ..... 4459
\@@_intersect_our_row_p:nnnn ..... 4409
\@@_j: ..... 5195, 5197,
5198, 5199, 5200, 5205, 5208, 5210, 5213,
5215, 5216, 5218, 5221, 5223, 5224, 5239,
5242, 5244, 5245, 5246, 5301, 5303, 5306,
5308, 5312, 5313, 5326, 5329, 5330, 5332,
5337, 5338, 5350, 5356, 5357, 5359, 5364, 5365
\l_@@_l_dim .....
.... 3813, 3814, 3827, 3828, 3840, 3846,
3857, 3865, 3873, 3878, 3890, 3891, 3898, 3899
\l_@@_large_nodes_bool 483, 584, 5180, 5184
\g_@@_last_col_found_bool ..... 338,
1230, 1557, 1625, 2650, 2724, 2799, 2923, 5877
\l_@@_last_col_int .....
.... 336, 337, 752, 795, 797, 810, 826,
847, 1251, 1254, 1560, 1706, 2871, 2873,
2924, 2926, 3565, 3600, 3922, 3938, 3976,
4000, 5658, 5663, 5664, 5665, 5668, 5696,
5710, 5722, 5734, 5746, 5758, 5773, 5804,
5809, 5817, 5833, 6121, 6127, 6133, 6774, 6792
\l_@@_last_col_without_value_bool ...
..... 335, 794, 2925, 6777
\l_@@_last_empty_column_int .....
..... 4994, 4995, 5008, 5014, 5027
\l_@@_last_empty_row_int .....
..... 4976, 4977, 4990, 5011
\l_@@_last_i_tl ..... 6398,
6404, 6407, 6420, 6454, 6467, 6469, 6517, 6524
\l_@@_last_j_tl .....
6399, 6405, 6408, 6431, 6433, 6490, 6500, 6507
\l_@@_last_row_int .....
.... 332, 333, 547, 875, 922, 1116, 1245,
1249, 1256, 1545, 1549, 1552, 1564, 1586,
2533, 2534, 2763, 2764, 2808, 2809, 2928,
3445, 3484, 3954, 3970, 3994, 4596, 4604,
5657, 5660, 5661, 5680, 5696, 5710, 5722,
5734, 5745, 5757, 5771, 5832, 5843, 6129, 7051

```

<code>\l_@@_last_row_without_value_bool</code> . . .	<code>\@@_node_position:</code> . . . . .
. . . . . 334, 1247, 1547, 2927	1354, 1356, 1360, 1362, 1426, 1428, 1436, 1442
<code>\l_@@_left_delim_dim</code> . . . . .	<code>\@@_node_position_i:</code> . . . . . 1439, 1443
. . . . . 1274, 1278, 1283, 2499, 2786	<code>\@@_node_right:nnnn</code> . . . . . 6576, 6578, 6651
<code>\g_@@_left_delim_tl</code> . . . . .	<code>\g_@@_not_empty_cell_bool</code> 278, 972, 979, 2890
1282, 1455, 1597, 1621, 1690, 2010, 2012, 5100	<code>\@@_not_in_exterior:nnnn</code> . . . . . 4451
<code>\l_@@_left_margin_dim</code> . . . . .	<code>\@@_not_in_exterior_p:nnnn</code> . . . . . 4387
. . . . . 485, 587, 1288, 2787, 5091, 5317	<code>\l_@@_notes_above_space_dim</code> . . . . . 476, 477
<code>\l_@@_letter_for_dotted_lines_str</code> . . .	<code>\l_@@_notes_bottomrule_bool</code> . . . . .
. . . . . 775, 786, 787, 1751	. . . . . 653, 813, 841, 2400
<code>\l_@@_letter_vlism_tl</code> . . . . . 255, 564, 1754	<code>\l_@@_notes_code_after_tl</code> . . . . . 651, 2409
<code>\l_@@_light_syntax_bool</code> 469, 541, 1291, 1507	<code>\l_@@_notes_code_before_tl</code> . . . . . 649, 2381
<code>\@@_light_syntax_i</code> . . . . . 2524, 2527	<code>\@@_notes_label_in_list:n</code> 345, 364, 372, 661
<code>\@@_line</code> . . . . . 3009, 4202	<code>\@@_notes_label_in_tabular:n</code> . 344, 385, 658
<code>\@@_line_i:nn</code> . . . . . 4209, 4216	<code>\l_@@_notes_para_bool</code> . . 647, 811, 839, 2385
<code>\l_@@_line_width_dim</code> . . . . .	<code>\@@_notes_style:n</code> . . . . .
. . . . . 312, 5627, 5925, 5963, 5974, 5980,	. . . . . 343, 346, 364, 372, 388, 393, 655
6001, 6021, 6036, 6038, 6048, 6049, 6051, 6062	<code>\l_@@_nullify_dots_bool</code> . . . . .
<code>\@@_line_with_light_syntax:n</code> . . . 2538, 2542	. . . . . 478, 582, 3929, 3945, 3961, 3985, 4008
<code>\@@_line_with_light_syntax_i:n</code> . . . . .	<code>\l_@@_number_of_notes_int</code> 342, 379, 389, 399
. . . . . 2537, 2543, 2544	<code>\@@_old_CT@arc@</code> . . . . . 1470, 3034
<code>\@@_math_toggle_token:</code> . . . . .	<code>\@@_old_cdots</code> . . . . . 1193, 3946
167, 956, 2757, 2774, 2803, 2819, 6214, 6218	<code>\@@_old_ddots</code> . . . . . 1195, 3986
<code>\g_@@_max_cell_width_dim</code> . . . . .	<code>\@@_old_dotfill</code> . . . . . 6158, 6161, 6169
. . . . . 965, 966, 1479, 2625, 5143, 5169	<code>\@@_old_dotfill:</code> . . . . . 1213
<code>\c_@@_max_l_dim</code> . . . . . 3827, 3832	<code>\l_@@_old_iRow_int</code> . . . . . 270, 1241, 3080
<code>\l_@@_medium_nodes_bool</code> 482, 583, 5178, 5824	<code>\@@_old_ialign:</code> . . . . . 1069, 1189, 5652
<code>\@@_message_hdotsfor:</code> 6783, 6791, 6798, 6806	<code>\@@_old_iddots</code> . . . . . 1196, 4009
<code>\@@_msg_new:nn</code> . . . . . 17,	<code>\l_@@_old_jCol_int</code> . . . . . 271, 1243, 3081
6692, 6716, 6725, 6755, 6788, 6795, 6803,	<code>\@@_old_ldots</code> . . . . . 1192, 3930
6811, 6816, 6822, 6827, 6832, 6837, 6842,	<code>\@@_old_multicolumn</code> . . . . . 1222, 4024
6847, 6852, 6857, 6863, 6869, 6875, 6880,	<code>\@@_old_pgfpaintanchor</code> . . . . . 175, 6585, 6589
6886, 6892, 6897, 6904, 6911, 6917, 6926,	<code>\@@_old_pgful@check@rerun</code> . . . . . 93, 97
6931, 6933, 6939, 6956, 6963, 6968, 6975,	<code>\@@_old_vdots</code> . . . . . 1194, 3962
6982, 6988, 6995, 7002, 7007, 7017, 7023,	<code>\@@_open_x_final_dim:</code> . . . . .
7030, 7037, 7043, 7049, 7057, 7059, 7065, 7344	. . . . . 3413, 3465, 3499, 3657, 3706
<code>\@@_msg_new:nnn</code> . . . 18, 6679, 6945, 7070,	<code>\@@_open_x_initial_dim:</code> . . . . .
7080, 7095, 7116, 7134, 7177, 7229, 7280, 7330	. . . . . 3391, 3458, 3496, 3650, 3700
<code>\@@_msg_redirect_name:nn</code> . . . . .	<code>\@@_open_y_final_dim:</code> 3537, 3589, 3655, 3705
. . . . . 19, 770, 1637, 5671, 5674	<code>\@@_open_y_initial_dim:</code> . . . . .
<code>\@@_multicolumn:nnn</code> . . . . . 1220, 2160	. . . . . 3519, 3586, 3647, 3699
<code>\g_@@_multicolumn_cells_seq</code> . . . . .	<code>\l_@@_parallelize_diags_bool</code> . . . . .
. . . . . 298, 1223, 1269, 2175,	. . . . . 473, 474, 579, 2960, 3661, 3709
2949, 2953, 2954, 5213, 5221, 5343, 5788, 5806	<code>\@@_patch_m_preamble:n</code> . . . . .
<code>\g_@@_multicolumn_sizes_seq</code> . . . . .	. . . . . 2169, 2195, 2229, 2234, 2300
. . . . . 299, 1224, 1270, 2177, 2955, 2956, 5344	<code>\@@_patch_m_preamble_i:n</code> . . . . .
<code>\g_@@_name_env_str</code> . . . . . 259, 264,	. . . . . 2199, 2200, 2201, 2216
265, 1462, 1463, 2554, 2843, 2844, 2852,	<code>\@@_patch_m_preamble_ii:nn</code> . . . . .
2853, 2882, 2895, 2903, 2913, 3032, 6142, 6772	. . . . . 2202, 2203, 2204, 2226
<code>\l_@@_name_str</code> . . . . . 481,	<code>\@@_patch_m_preamble_iii:n</code> . . . . . 2205, 2231
635, 894, 897, 997, 1000, 1087, 1090, 2571,	<code>\@@_patch_m_preamble_iv:nnn</code> . . . . .
2572, 2596, 2597, 2614, 2615, 2645, 2646,	. . . . . 2206, 2207, 2208, 2236
2671, 2674, 2717, 2720, 2738, 2741, 3160,	<code>\@@_patch_m_preamble_ix:n</code> . . . . . 2285, 2303
3161, 3172, 3175, 3178, 5334, 5337, 5361, 5364	<code>\@@_patch_m_preamble_v:nnnn</code> 2209, 2210, 2256
<code>\g_@@_names_seq</code> . . . . . 240, 632, 634, 7342	<code>\@@_patch_m_preamble_vi:n</code> . . . . . 2211, 2277
<code>\l_@@_nb_cols_int</code> . . . . .	<code>\@@_patch_m_preamble_x:n</code> . . . . .
. . . . . 6095, 6106, 6114, 6120, 6126, 6132	. . . . . 2224, 2254, 2275, 2280, 2282, 2306
<code>\l_@@_nb_rows_int</code> . . . . . 6094, 6105, 6123	<code>\@@_patch_node_for_cell:</code> . . . . . 1007, 1397
<code>\@@_newcolumnmtype</code> . . . . . 1123, 1656, 1657	<code>\@@_patch_node_for_cell:n</code> 1005, 1031, 1034
<code>\@@_node_for_cell:</code> 974, 981, 1397, 2784, 2834	<code>\@@_patch_preamble:n</code> . . . . .
<code>\@@_node_for_multicolumn:nn</code> . . . . 5345, 5352	. . . . . 1678, 1724, 1760, 1768, 1789, 1822,
<code>\@@_node_left:nn</code> . . . . . 6566, 6567, 6627	2014, 2030, 2032, 2048, 2056, 2082, 2132, 2152
	<code>\@@_patch_preamble_i:n</code> 1728, 1729, 1730, 1775

```

\@@_patch_preamble_ii:nn .....
..... 1731, 1732, 1733, 1786
\@@_patch_preamble_iii:n . 1734, 1791, 1799
\@@_patch_preamble_iii_i:n .... 1794, 1796
\@@_patch_preamble_iv:n .....
..... 1735, 1736, 1737, 1844
\@@_patch_preamble_iv_i:n ..... 1847, 1849
\@@_patch_preamble_iv_ii:w 1852, 1853, 1855
\@@_patch_preamble_iv_iii:nn ... 1856, 1857
\@@_patch_preamble_iv_iv:n 1861, 1863, 2105
\@@_patch_preamble_iv_v:nnnnnn . 1867, 1891
\@@_patch_preamble_ix:n ..... 1747, 2085
\@@_patch_preamble_ix_i:w 2088, 2089, 2091
\@@_patch_preamble_ix_ii:n .... 2092, 2095
\@@_patch_preamble_v:nnnn 1738, 1739, 1940
\@@_patch_preamble_vi:n ..... 1740, 1963
\@@_patch_preamble_vi_i:w 1966, 1967, 1969
\@@_patch_preamble_vi_ii:n 1970, 1975, 1992
\@@_patch_preamble_vii:nn .....
..... 1741, 1742, 1743, 2005
\@@_patch_preamble_vii_i:nn 2018, 2021, 2023
\@@_patch_preamble_viii:nn .....
..... 1744, 1745, 1746, 2034
\@@_patch_preamble_viii_i:nnn .. 2038, 2060
\@@_patch_preamble_x:n .....
..... 1784, 1889, 1961, 1988, 2001, 2123, 2134, 2158
\@@_patch_preamble_xi:n ..... 1752, 2126
\@@_patch_preamble_xii:n ..... 2137, 2155
\@@_pgf_rect_node:nnn ..... 432, 1440, 5826
\@@_pgf_rect_node:nnnnn .....
..... 407, 5328, 5355, 5776, 5820, 6553
\c_@@_pgfortikzpicture_tl .....
..... 42, 46, 3062, 4235, 5074
\@@_pgfpntanchor:n ..... 6581, 6586
\@@_pgfpntanchor_i:nn ..... 6589, 6591
\@@_pgfpntanchor_ii:w ..... 6592, 6600
\@@_pgfpntanchor_iii:w ..... 6613, 6615
\@@_picture_position: .....
..... 1349, 1356, 1362, 1428, 1442, 1443
\g_@@_pos_of_blocks_seq .....
..... 292, 1268, 1481, 2178, 2941, 2945, 2946,
..... 2986, 3040, 4385, 4625, 4763, 5041, 5700, 6182
\g_@@_pos_of_stroken_blocks_seq .....
..... 294, 1482, 4629, 4767, 5712
\g_@@_pos_of_xdots_seq .....
..... 293, 1483, 2987, 3326, 4627, 4765
\g_@@_post_action_cell_tl .....
..... 854, 958, 1307, 1316, 1921, 2112
\@@_pre_array: ..... 1239, 1299, 1504
\@@_pre_array_i:w ..... 1295, 1504
\@@_pre_array_ii: ..... 1146, 1271
\@@_pre_code_before: ..... 1342, 1418
\c_@@_preamble_first_col_tl .... 1695, 2749
\c_@@_preamble_last_col_tl .... 1707, 2794
\g_@@_preamble_tl .....
..... 1457, 1658, 1662, 1666, 1672, 1686, 1695,
..... 1704, 1707, 1716, 1720, 1758, 1767, 1777,
..... 1788, 1801, 1893, 1942, 1977, 1994, 2017,
..... 2045, 2053, 2066, 2072, 2107, 2128, 2141,
..... 2148, 2157, 2168, 2170, 2218, 2228, 2233,
..... 2238, 2258, 2279, 2289, 2296, 2305, 2508, 2535
\@@_pred:n ..... 136,
..... 166, 2873, 4665, 4678, 4803, 4816, 5991, 5996

\@@_provide_pgfsyspdfmark: ... 60, 69, 1452
\@@_put_box_in_flow: ..... 1623, 2308, 2501
\@@_put_box_in_flow_bis:nn .... 1620, 2468
\@@_put_box_in_flow_i: ..... 2314, 2316
\@@_qpoint:n . 227, 2319, 2321, 2333, 2349,
..... 2416, 2418, 2434, 2445, 2456, 3148, 3150,
..... 3152, 3154, 3164, 3166, 3408, 3430, 3459,
..... 3466, 3505, 3507, 3521, 3539, 3595, 3597,
..... 3648, 3656, 4243, 4246, 4496, 4500, 4516,
..... 4518, 4688, 4690, 4692, 4826, 4828, 4830,
..... 5082, 5086, 5093, 5127, 5130, 5132, 5234,
..... 5244, 5767, 5769, 5771, 5773, 5797, 5817,
..... 5846, 5947, 5949, 5956, 5958, 6035, 6037,
..... 6039, 6046, 6050, 6052, 6195, 6197, 6200,
..... 6202, 6255, 6257, 6451, 6454, 6492, 6509, 6526
\l_@@_radius_dim ..... 464, 465, 2129,
..... 2977, 3470, 3471, 3907, 5060, 5084, 5128, 5129
\l_@@_real_left_delim_dim 2470, 2485, 2500
\l_@@_real_right_delim_dim 2471, 2497, 2503
\@@_recreate_cell_nodes: ..... 1365, 1422
\g_@@_recreate_cell_nodes_bool .....
..... 480, 1158, 1365, 1389, 1396, 1401
\@@_rectanglecolor .. 1376, 4307, 4340, 4360
\@@_rectanglecolor:nnn ... 4313, 4322, 4325
\@@_renew_NC@rewrite@S: .... 186, 188, 1229
\@@_renew_dots: ..... 1130, 1219
\l_@@_renew_dots_bool .....
..... 580, 760, 1219, 6700, 6707
\@@_renew_matrix: 755, 759, 6074, 6702, 6706
\l_@@_respect_blocks_bool 4366, 4383, 4406
\@@_restore_iRow_jCol: ..... 3033, 3078
\@@_revtex_array: ..... 1046, 1057
\c_@@_revtex_bool ..... 50, 52, 55, 57, 1056
\l_@@_right_delim_dim .....
..... 1275, 1279, 1285, 2502, 2831
\g_@@_right_delim_tl .. 1284, 1456, 1615,
..... 1621, 1691, 2013, 2042, 2043, 2064, 2069, 5102
\l_@@_right_margin_dim .....
..... 486, 589, 1511, 2832, 3604, 5098, 5320
\@@_rotate: ..... 1211, 4195
\g_@@_rotate_bool .....
..... 246, 931, 959, 1955, 2270, 2776,
..... 2821, 4195, 5469, 5510, 5515, 5575, 5591, 5763
\@@_rotate_cell_box: .....
..... 919, 959, 1955, 2270, 2776, 2821, 5763
\l_@@_rounded_corners_dim .....
..... 310, 5619, 5735, 5939, 5940, 5975, 6003, 6060
\@@_roundedrectanglecolor ..... 1377, 4316
\l_@@_row_max_int .... 301, 3196, 3338, 3355
\l_@@_row_min_int .... 300, 3264, 3336, 3353
\g_@@_row_of_col_done_bool .....
..... 277, 1101, 1461, 2580
\g_@@_row_style_tl ..... 281, 865,
..... 884, 1305, 1314, 1329, 1337, 1485, 1905, 5464
\g_@@_row_total_int .... 233, 1226, 1346,
..... 1352, 1424, 1563, 2344, 2451, 2928, 2935,
..... 3098, 3394, 3416, 4110, 5188, 5202, 5229,
..... 5324, 5680, 5783, 5801, 6262, 6407, 6421, 6901
\@@_rowcolor ..... 1378, 4289
\@@_rowcolor_tabular ..... 1143, 4564
\@@_rowcolors ..... 1379, 4444
\@@_rowcolors_i:nnnn ..... 4410, 4446
\l_@@_rowcolors_restart_bool ... 4369, 4398

```



<code>\@@_rowlistcolors</code>	1380, 4373, 4445	<code>\c_@@_table_collect_begin_tl</code>	214, 216, 1996
<code>\g_@@_rows_seq</code>	2530, 2532, 2534, 2536, 2538	<code>\c_@@_table_print_tl</code>	217, 218, 1998
<code>\l_@@_rows_tl</code>	4285, 4333, 4412, 4429, 4477, 4502	<code>\l_@@_tabular_width_dim</code>	244, 1062, 1064, 1718, 2914
<code>\l_@@_rules_color_tl</code>	272, 516, 1502, 1503, 6482, 6483	<code>\l_@@_tabularnote_tl</code>	341, 815, 843, 2373, 2382
<code>\@@_set_CT@arc@:</code>	169, 1503, 6483	<code>\g_@@_tabularnotes_seq</code>	340, 380, 2388, 2394, 2410
<code>\@@_set_CT@arc@_i:</code>	170, 171	<code>\@@_test_hline_in_block:nnnn</code>	4764, 4766, 4902
<code>\@@_set_CT@arc@_ii:</code>	170, 173	<code>\@@_test_hline_in_stroken_block:nnnn</code>	4768, 4924
<code>\@@_set_final_coords:</code>	3364, 3389	<code>\@@_test_if_cell_in_a_block:nn</code>	4980, 4998, 5016, 5036
<code>\@@_set_final_coords_from_anchor:n</code>	3380, 3469, 3500, 3581, 3590, 3660, 3708	<code>\@@_test_if_cell_in_block:nnnnnnn</code>	5042, 5044
<code>\@@_set_initial_coords:</code>	3359, 3378	<code>\@@_test_if_math_mode:</code>	249, 1467, 2854
<code>\@@_set_initial_coords_from_anchor:n</code>	3369, 3462, 3497, 3580, 3587, 3652, 3702	<code>\@@_test_in_corner_h:</code>	4769, 4797
<code>\@@_set_size:n</code>	6092, 6107	<code>\@@_test_in_corner_v:</code>	4631, 4659
<code>\c_@@_siunitx_loaded_bool</code>	176, 180, 185, 203	<code>\@@_test_vline_in_block:nnnn</code>	4626, 4628, 4913
<code>\c_@@_size_seq</code>	1249, 1254, 1344, 1345, 1346, 1347, 2931	<code>\@@_test_vline_in_stroken_block:nnnn</code>	4630, 4935
<code>\l_@@_small_bool</code>	753, 800, 806, 828, 863, 1153, 2007, 2036, 2758, 2804, 2975	<code>\l_@@_the_array_box</code>	1272, 1287, 1529, 1537, 2361, 2362, 2364, 2367
<code>\@@_standard_cline</code>	132, 1198	<code>\c_@@_tikz_loaded_bool</code>	27, 41, 1367, 2997, 5111, 5611
<code>\@@_standard_cline:w</code>	132, 133	<code>\l_@@_tikz_seq</code>	308, 5612, 5738, 5747
<code>\l_@@_standard_cline_bool</code>	457, 528, 1197	<code>\g_@@_total_X_weight_int</code>	273, 1148, 1516, 1519, 1538, 2103
<code>\c_@@_standard_tl</code>	467, 468, 3734, 5104, 5134	<code>\@@_true_c:</code>	2211
<code>\g_@@_static_num_of_col_int</code>	305, 1632, 1679, 5668, 6807, 6819, 7010	<code>\l_@@_type_of_col_tl</code>	798, 799, 2883, 2885, 6099, 6100, 6101, 6109, 6114
<code>\l_@@_stop_loop_bool</code>	3190, 3191, 3223, 3236, 3245, 3258, 3259, 3291, 3304, 3313	<code>\c_@@_types_of_matrix_seq</code>	6762, 6763, 6768, 6772
<code>\@@_store_in_tmpb_tl</code>	5917, 5919	<code>\@@_update_for_first_and_last_row:</code>	902, 967, 1258, 2778, 2823
<code>\@@_stroke_block:nnn</code>	5707, 5921	<code>\@@_use_arraybox_with_notes:</code>	1577, 2429
<code>\@@_stroke_borders_block:nnn</code>	5719, 5999	<code>\@@_use_arraybox_with_notes_b:</code>	1574, 2413
<code>\@@_stroke_horizontal:n</code>	6027, 6029, 6044	<code>\@@_use_arraybox_with_notes_c:</code>	1575, 1606, 2357, 2427, 2466
<code>\@@_stroke_vertical:n</code>	6023, 6025, 6033	<code>\@@_vdottedline:n</code>	2131, 5107
<code>\@@_sub_matrix:nnnnnnn</code>	6389, 6393	<code>\@@_vdottedline_i:n</code>	5114, 5119, 5123
<code>\@@_sub_matrix_i:nnnn</code>	6443, 6449	<code>\@@_vline:nnnn</code>	1817, 4606, 4744, 5991
<code>\l_@@_submatrix_extra_height_dim</code>	321, 6308, 6477	<code>\@@_vline_i:nn</code>	2991
<code>\l_@@_submatrix_hlines_clist</code>	326, 6320, 6338, 6516, 6518	<code>\@@_vline_i:nnnn</code>	4611, 4615
<code>\l_@@_submatrix_left_xshift_dim</code>	322, 6310, 6529, 6562	<code>\@@_vline_i_complete:nn</code>	2991, 4727
<code>\l_@@_submatrix_name_str</code>	6353, 6411, 6551, 6553, 6565, 6567, 6575, 6579	<code>\@@_vline_ii:nnnn</code>	4640, 4651, 4684, 4728
<code>\g_@@_submatrix_names_seq</code>	296, 3013, 6350, 6354, 6954	<code>\l_@@_vlines_clist</code>	325, 558, 570, 575, 604, 1664, 1670, 1701, 1713, 2139, 2146, 2287, 2294, 2687, 2995, 4742, 4743
<code>\l_@@_submatrix_right_xshift_dim</code>	323, 6312, 6538, 6572	<code>\l_@@_vpos_col_str</code>	1836, 1839, 1841, 1846, 1868, 1884, 2098
<code>\g_@@_submatrix_seq</code>	304, 1266, 3340, 6376	<code>\l_@@_vpos_of_block_tl</code>	316, 317, 5382, 5384, 5490, 5501, 5579, 5596, 5644, 5646
<code>\l_@@_submatrix_slim_bool</code>	6318, 6419, 6922	<code>\@@_w:</code>	1656, 1738, 2209
<code>\l_@@_submatrix_vlines_clist</code>	327, 6322, 6340, 6499, 6501	<code>\l_@@_weight_int</code>	2094, 2099, 2100, 2103, 2105
<code>\@@_succ:n</code>	161, 165, 1079, 1085, 1090, 1091, 1111, 1818, 2026, 2146, 2176, 2294, 2321, 2663, 2669, 2674, 2675, 2699, 2707, 2720, 2721, 2732, 2736, 2741, 2742, 3430, 3507, 3597, 3656, 4455, 4500, 4516, 4661, 4692, 4738, 4799, 4830, 4874, 4899, 5046, 5048, 5050, 5052, 5093, 5132, 5294, 5298, 5308, 5312, 5771, 5773, 5817, 5956, 5958, 6070, 6200, 6202, 6257	<code>\l_@@_width_dim</code>	836, 1529, 1537, 2893, 2894, 2904, 2905
<code>\l_@@_suffix_tl</code>	5256, 5267, 5277, 5280, 5329, 5337, 5338, 5356, 5364, 5365	<code>\g_@@_width_first_col_dim</code>	290, 1460, 1568, 2575, 2779, 2780
		<code>\g_@@_width_last_col_dim</code>	289, 1459, 1627, 2728, 2824, 2825
		<code>\l_@@_width_used_bool</code>	297, 837, 1514

<code>\@@_write_aux_for_cell_nodes:</code>	.. 3031, 3093	<code>\arraycolsep</code>	..... .. 588, 590, 592, 1061, 1156, 1278, 1279, 1605, 1609, 2362, 2698, 2702, 2712, 5090, 5097
<code>\l_@@_x_final_dim</code>	..... 284, 3366, 3415, 3424, 3425, 3428, 3431, 3432, 3583, 3599, 3607, 3611, 3615, 3617, 3622, 3624, 3658, 3667, 3675, 3715, 3723, 3762, 3777, 3786, 3820, 3872, 3888, 4247, 5094, 5103, 5129, 6418, 6434, 6435, 6441, 6538, 6555, 6572	<code>\arrayrulecolor</code>	..... 107
<code>\l_@@_x_initial_dim</code>	..... 282, 3361, 3393, 3402, 3403, 3406, 3409, 3410, 3583, 3598, 3599, 3606, 3611, 3615, 3617, 3619, 3622, 3624, 3649, 3667, 3675, 3715, 3723, 3759, 3776, 3786, 3820, 3872, 3886, 3888, 3906, 3908, 4244, 5087, 5101, 5128, 6417, 6427, 6428, 6438, 6529, 6554, 6562	<code>\arrayrulewidth</code>	140, 145, 157, 518, 893, 1078, 1080, 1086, 1117, 1667, 1673, 1759, 1809, 2142, 2149, 2290, 2297, 2421, 2460, 2587, 2589, 2595, 2605, 2607, 2613, 2636, 2638, 2644, 2662, 2664, 2670, 2696, 2700, 2712, 2715, 4498, 4499, 4501, 4517, 4519, 4703, 4704, 4706, 4717, 4723, 4842, 4853, 4859, 4895, 5169, 5925, 5980, 6001, 6280, 6477, 6481
<code>\l_@@_xdots_color_tl</code>	491, 505, 3449, 3488, 3569, 3570, 3638, 3690, 3742, 4114, 4189, 4206	<code>\arraystretch</code>	..... 1155, 3523, 3541, 5466, 5572, 5588, 6453, 6456
<code>\l_@@_xdots_down_tl</code>	... 509, 3749, 3770, 3805	<code>\AtBeginDocument</code>	23, 28, 85, 101, 177, 183, 199, 347, 461, 463, 465, 477, 667, 683, 1971, 3058, 3913, 4043, 4120, 4198, 4231, 5068, 6382
<code>\l_@@_xdots_line_style_tl</code>	..... ..... 466, 468, 501, 3734, 3742, 5104, 5134	<code>\AtBeginEnvironment</code>	..... 131, 1221
<code>\l_@@_xdots_shorten_dim</code>	..... 462, 463, 507, 2979, 3756, 3757, 3846, 3857, 3865	<code>\AutoNiceMatrix</code>	..... 6151
<code>\l_@@_xdots_up_tl</code>	... 510, 3748, 3769, 3795	<code>\AutoNiceMatrixWithDelims</code>	... 6103, 6143, 6155
<code>\l_@@_y_final_dim</code>	..... ..... 285, 3367, 3467, 3471, 3509, 3513, 3515, 3540, 3550, 3551, 3669, 3672, 3717, 3720, 3762, 3777, 3785, 3822, 3877, 3896, 4248, 5085, 5133, 6258, 6280, 6295, 6455, 6470, 6471, 6476, 6494, 6511, 6555, 6563, 6573	<b>B</b>	
<code>\l_@@_y_initial_dim</code>	..... ..... 283, 3362, 3460, 3470, 3508, 3509, 3513, 3515, 3522, 3532, 3533, 3669, 3674, 3717, 3722, 3759, 3776, 3785, 3822, 3877, 3894, 3896, 3906, 3909, 4245, 5083, 5084, 5085, 5131, 6256, 6280, 6295, 6452, 6463, 6464, 6476, 6493, 6510, 6554, 6563, 6573		
<code>\</code>	..... 2521, 2543, 6121, 6127, 6133, 6681, 6682, 6722, 6731, 6759, 6819, 6824, 6829, 6834, 6839, 6883, 6888, 6894, 6901, 6907, 6908, 6923, 6928, 6936, 6942, 6948, 6949, 6960, 6972, 6979, 6991, 6998, 7005, 7014, 7020, 7027, 7034, 7040, 7046, 7058, 7062, 7067, 7073, 7082, 7083, 7097, 7098, 7114, 7118, 7119, 7137, 7138, 7180, 7181, 7232, 7233, 7283, 7284, 7337, 7338	<code>\baselineskip</code>	..... 110, 116, 1933
<code>\{</code>	..... 265, 1743, 2027, 2052, 2861, 6150, 6534, 6941, 7027, 7180, 7283	<code>\begingroup</code>	..... 2163
<code>\}</code>	..... 265, 1746, 2027, 2037, 2861, 6150, 6543, 6941, 7027, 7180, 7283	<code>\bgroup</code>	..... 1454
<code>\ </code>	..... 2863, 6149	<code>\bigskip</code>	..... 3138
<code>\_</code>	.. 6758, 6786, 6791, 6798, 6806, 6807, 6818, 6819, 6882, 6900, 6901, 6914, 6920, 6924, 6932, 6935, 6947, 6953, 6965, 7009, 7010, 7011, 7019, 7025, 7032, 7045, 7052, 7053, 7061	<code>\Block</code>	..... 1210, 6978, 6985, 7032, 7073
<b>A</b>		<code>\BNiceMatrix</code>	..... 6089
<code>\A</code>	..... 6348	<code>\bNiceMatrix</code>	..... 6086
<code>\aboverulesep</code>	..... 2404	<code>\Body</code>	..... 1295
<code>\addtocounter</code>	..... 397	bool commands:	
<code>\alph</code>	..... 343	<code>\bool_do_until:Nn</code>	..... 3191, 3259
<code>\anchor</code>	..... 3090, 3091	<code>\bool_gset_false:N</code>	..... 931, 978, 979, 1100, 1230, 1389, 1461, 1486, 1663, 2790, 2837, 4911, 4922, 4933, 4944, 5515
<code>\arraybackslash</code>	..... 1906, 2114, 2244	<code>\bool_gset_true:N</code>	..... 1489, 1821, 2580, 2754, 2799, 2890, 3931, 3947, 3963, 3987, 4010, 4021, 4195, 4624, 4762
<code>\arraycolor</code>	..... 1381, 6758	<code>\bool_if:NTF</code>	..... .. 168, 669, 674, 685, 690, 860, 863, 959, 1074, 1101, 1149, 1153, 1158, 1218, 1219, 1248, 1253, 1267, 1365, 1367, 1391, 1394, 1396, 1416, 1453, 1468, 1478, 1514, 1547, 1625, 1630, 1649, 1654, 1681, 1693, 1955, 2007, 2036, 2270, 2400, 2566, 2583, 2601, 2619, 2632, 2658, 2692, 2724, 2729, 2758, 2776, 2804, 2821, 2923, 2925, 2927, 2960, 2975, 2997, 3014, 3017, 3031, 3512, 3514, 3661, 3709, 3929, 3945, 3961, 3985, 4008, 4383, 4406, 4989, 5007, 5025, 5152, 5162, 5184, 5469, 5510, 5575, 5591, 5763, 5780, 5824, 5850, 5887, 6162, 6734, 6744, 6777, 6922
		<code>\bool_if:nTF</code>	..... ..... 185, 349, 376, 1037, 1557, 3345, 4220, 4461, 4733, 4737, 4869, 4873, 5178, 5427, 5877, 6259, 6269, 6271, 6284, 6290, 6299
		<code>\bool_lazy_all:nTF</code>	..... 1697, 1709, 2683, 2984, 4694, 4832, 4904, 4915, 4926, 4937
		<code>\bool_lazy_and:nnTF</code>	..... 202, 1763, 2359, 2622, 2697, 2701, 2709, 2759, 3501, 3768, 4027, 4472, 5472, 5943, 6503, 6520
		<code>\bool_lazy_or:nnTF</code>	..... ..... 498, 971, 1029, 1689, 2342, 2371,

2449, 2807, 3578, 3826, 4453, 4485, 4489, 4539, 4543, 4981, 4999, 5017, 5402, 5407, 6406	\clist_set:Nn 570, 571, 575, 576, 603, 604, 605
\bool_lazy_or_p:n 2762	\clist_set_eq:NN 4531
\bool_not_p:n	\l_tmpa_clist 4531, 4533
... 1700, 1702, 1712, 1714, 2624, 2686, 2688	\CodeAfter 855, 1214, 2524, 2527, 2753, 2798, 3012, 7085
\bool_set:Nn 3582	\CodeBefore 1450, 6758
\c_false_bool	\color 111, 117, 172,
2047, 2055, 2068, 2074, 2080, 3925, 3941, 3957	174, 1328, 1330, 1596, 1614, 3443, 3446,
\g_tmpa_bool	3449, 3482, 3485, 3488, 3563, 3566, 3570,
... 4624, 4632, 4666, 4674, 4679, 4762,	3638, 3690, 4108, 4111, 4114, 4183, 4186,
4770, 4804, 4812, 4817, 4911, 4922, 4933, 4944	4189, 4206, 4271, 5463, 5621, 6289, 6636, 6660
\g_tmpb_bool 1663, 1693, 1821	\colorlet 257, 258, 871, 878, 2768, 2812
\l_tmpb_bool 4986, 5000, 5018, 5040, 5053	\columncolor 1144, 1382, 3023, 4582
\c_true_bool 2026	\cr 144, 162, 2747
box commands:	\crrc 2560
\box_clear_new:N 1151, 1272	cs commands:
\box_dp:N	\cs_generate_variant:Nn
... 887, 907, 944, 964, 1019, 1179, 1188,	... 58, 164, 3765, 4263, 4264, 5174, 5175
1263, 2249, 2311, 2479, 2492, 3541, 5545, 6456	\cs_gset:Npn 111, 117, 5167
\box_gclear_new:N 5456	\cs_gset_eq:NN 69, 219, 1172, 1470, 3034
\box_grotate:Nn 5512	\cs_if_exist:Ntf 57, 131, 1241,
\box_ht:N 888, 909, 915, 927, 950, 962, 1011,	1243, 1431, 1471, 1474, 1973, 3080, 3081,
1181, 1183, 1186, 1261, 1901, 1924, 1926,	3102, 3226, 3239, 3294, 3307, 3396, 3418,
1932, 2245, 2310, 2479, 2492, 3523, 5536, 6453	3526, 3544, 4076, 4094, 4151, 4169, 5154,
\box_move_down:nn 1019, 1930	5207, 5785, 5803, 6264, 6423, 6430, 6459, 6466
\box_move_up:nn	\cs_if_exist_p:N
... 76, 78, 80, 1011, 2354, 2427, 2466	... 204, 499, 4697, 4835, 4983, 5002, 5020
\box_rotate:Nn 921	\cs_if_free:Ntf
\box_set_dp:Nn 943, 963, 2311	... 65, 3438, 3477, 3558, 3633, 3685
\box_set_ht:Nn 949, 961, 2310	\cs_if_free_p:N 4222, 4224
\box_set_wd:Nn 937, 2361	\cs_new_protected:Npx 3060, 4233, 5070
\box_use:N 400, 928, 1018, 1935	\cs_set:Nn 655, 658, 661
\box_use_drop:N 969, 975, 994, 1957, 2272,	\cs_set:Npn 107, 108, 113, 114,
2313, 2354, 2355, 2367, 2785, 5555, 5872, 5909	119, 132, 133, 147, 149, 150, 172, 174, 346,
\box_wd:N 401, 938, 966, 973,	1125, 2164, 2186, 3185, 3247, 3315, 4118,
1032, 1283, 1285, 1529, 1537, 2362, 2364,	4193, 4883, 4884, 4890, 4891, 5466, 5572, 5588
2486, 2498, 2780, 2783, 2825, 2829, 5524, 6169	\cs_set_nopar:Npn
\l_tmpa_box	... 1051, 1063, 1155, 1166, 5349, 5350
... 383, 400, 401, 1282, 1283, 1284, 1285,	\cs_set_nopar:Npx 1064
1592, 2310, 2311, 2313, 2354, 2355, 2479, 2492	\cs_set_protected:Npn 6140
\l_tmpb_box 2472, 2486, 2487, 2498	\cs_set_protected_nopar:Npn 5444, 5750
<b>C</b>	
\c 59, 1685	<b>D</b>
\Cdots 1201, 3936, 3939	\Ddots 1203, 3969, 3970, 3975, 3976
\cdots 1133, 1193	\ddots 1135, 1195
\cellcolor 1142, 1375, 4559	\diagbox 1215, 5444, 5750
\centering 1879, 5480	dim commands:
char commands:	\dim_add:Nn 5318
\char_set_catcode_space:n 1642	\dim_compare:nNtf
\chessboardcolors 1383	110, 116, 935, 941, 947, 1718, 2620, 2829,
\cline 160, 1198, 1199	3406, 3428, 3617, 5231, 5241, 5795, 5815, 6169
clist commands:	\dim_compare_p:n 5474
\clist_clear:N 4532	\dim_gzero:N 939, 945, 951
\clist_if_empty:Ntf 4631, 4769, 5715	\dim_max:nn 5220, 5224
\clist_if_empty_p:N 2687	\dim_min:nn 5212, 5216
\clist_if_in:NnTF 1109, 1670, 2146,	\dim_ratio:nn 3676, 3724,
2294, 4743, 4879, 6022, 6024, 6026, 6028, 6047	3840, 3845, 3856, 3864, 3873, 3878, 3889, 3897
\clist_if_in:nnTF 6008	\dim_set:Nn 5193, 5200, 5211, 5215,
\clist_map_inline:Nn	5219, 5223, 5235, 5236, 5245, 5246, 5290, 5303
... 4479, 4502, 4533, 4949, 6006, 6501, 6518	\dim_set_eq:NN 5191, 5198, 5298, 5312
\clist_map_inline:nn 2878, 4339, 4391, 6067	\dim_sub:Nn 5315
\clist_new:N 309, 324, 325, 326, 327, 475	\dim_use:N
\clist_put_right:Nn 4550	... 5232, 5242, 5293, 5294, 5306, 5307,
	5330, 5331, 5332, 5333, 5357, 5358, 5359, 5360



<code>\dim_zero_new:N</code> . . . . .	5190, 5192, 5197, 5199	<code>\fontdimen</code> . . . . .	2350
<code>\c_max_dim</code> . . . . .	3393, 3406, 3415, 3428, 5191, 5193, 5198, 5200, 5232, 5242, 5782, 5795, 5800, 5815, 6260, 6261, 6417, 6418, 6438, 6441	fp commands:	
<code>\dotfill</code> . . . . .	1213, 6158	<code>\fp_eval:n</code> . . . . .	3781
<code>\dots</code> . . . . .	1137	<code>\fp_to_dim:n</code> . . . . .	3816
<code>\doublerulesep</code> . . . . .	1810, 4706, 4718, 4842, 4854, 4896	<code>\futurelet</code> . . . . .	126
<code>\doublerulesepcolor</code> . . . . .	113		
<code>\draw</code> . . . . .	3753		
		<b>G</b>	
		<code>\globaldefs</code> . . . . .	1636, 5673
		group commands:	
		<code>\group_insert_after:N</code> . . . . .	6163, 6164, 6166, 6167
<b>E</b>		<b>H</b>	
<code>\egroup</code> . . . . .	1640	<code>\halign</code> . . . . .	1190
<code>\else</code> . . . . .	2164	<code>\hbox</code> . . . . .	1072, 1601, 2427, 2466, 2585, 2603, 2630, 2634, 2660, 2694, 3111, 3124
else commands:		hbox commands:	
<code>\else:</code> . . . . .	251	<code>\hbox:n</code> . . . . .	76, 78, 81, 2365
<code>\endarray</code> . . . . .	2512, 2540	<code>\hbox_gset:Nn</code> . . . . .	5458
<code>\endBNiceMatrix</code> . . . . .	6090	<code>\hbox_overlap_left:n</code> . . . . .	1012, 1020, 2564, 2781
<code>\endbNiceMatrix</code> . . . . .	6087	<code>\hbox_overlap_right:n</code> . . . . .	400, 2726, 2827
<code>\endgroup</code> . . . . .	2172	<code>\hbox_set:Nn</code> . . . . .	383, 1009, 1282, 1284, 1592, 1928, 2113, 2472, 2487, 5762
<code>\endNiceArray</code> . . . . .	2900, 2910, 2919	<code>\hbox_set:Nw</code> . . . . .	859, 1287, 1946, 2261, 2756, 2802
<code>\endNiceArrayWithDelims</code> . . . . .	2847, 2857	<code>\hbox_set_end:</code> . . . . .	957, 1513, 1954, 2269, 2775, 2820
<code>\endpgfscope</code> . . . . .	3116, 3129, 3810, 6215	<code>\hbox_to_wd:nn</code> . . . . .	425, 450
<code>\endpNiceMatrix</code> . . . . .	6078	<code>\Hdotsfor</code> . . . . .	1208, 6786, 6965
<code>\endsavenotes</code> . . . . .	1649	<code>\hdotsfor</code> . . . . .	1138
<code>\endtabularnotes</code> . . . . .	2395	<code>\hdottedline</code> . . . . .	1205
<code>\endVNiceMatrix</code> . . . . .	6084	<code>\heavyrulewidth</code> . . . . .	2405
<code>\endvNiceMatrix</code> . . . . .	6081	<code>\hfil</code> . . . . .	1739, 2210
<code>\enskip</code> . . . . .	2017, 2045, 2053, 2066, 2072	<code>\hfill</code> . . . . .	140, 157
<code>\ensuremath</code> . . . . .	3930, 3946, 3962, 3986, 4009	<code>\Hline</code> . . . . .	1206, 4886
<code>\everycr</code> . . . . .	143, 162, 1176	<code>\hline</code> . . . . .	119
<code>\everypar</code> . . . . .	1899, 1902	<code>\hrule</code> . . . . .	123, 140, 157, 1117, 2405, 6294, 6641, 6665
exp commands:		<code>\hskip</code> . . . . .	122
<code>\exp_after:wN</code> . . . . .	192, 1503, 1597, 1615, 1678, 2169, 6483	<code>\Hspace</code> . . . . .	1207
<code>\exp_args:NNc</code> . . . . .	5156	<code>\hspace</code> . . . . .	4022
<code>\exp_args:Nne</code> . . . . .	2885	<code>\hss</code> . . . . .	1739, 2210
<code>\exp_args:NNV</code> . . . . .	2532, 3917, 3933, 3949, 3965, 3989, 4047, 4124, 4202, 6386		
<code>\exp_args:NNx</code> . . . . .	1108, 2145, 2293		
<code>\exp_args:No</code> . . . . .	3741		
<code>\exp_args:NV</code> . . . . .	1658, 2170, 2508, 2535, 2537, 5934		
<code>\exp_args:Nxx</code> . . . . .	5437, 5438		
<code>\exp_last_unbraced:NV</code> . . . . .	1392, 3015, 5727		
<code>\exp_not:N</code> . . . . .	42, 43, 46, 47, 1759, 1803, 1872, 1874, 1879, 1880, 1881, 2931, 2945, 2953, 2955, 3026, 4570, 4582, 4968, 5072, 5073, 5490, 5501, 5579, 5596, 5730, 6113, 6589		
<code>\exp_not:n</code> . . . . .	1043, 1646, 3027, 4057, 4058, 4134, 4559, 4570, 4572, 4583, 5453, 5553, 5565, 5566, 5694, 5708, 5720, 5732, 5759, 6115, 6179, 6180		
<code>\ExplSyntaxOff</code> . . . . .	67, 1648, 5171		
<code>\ExplSyntaxOn</code> . . . . .	64, 1641, 5164		
<code>\extrarowheight</code> . . . . .	3523, 5467, 5573, 5589, 6453		
<b>F</b>		<b>I</b>	
<code>\fi</code> . . . . .	121, 1660, 2164, 2167, 4883, 4900	<code>\ialign</code> . . . . .	1069, 1166, 1189, 5652
fi commands:		<code>\iddots</code> . . . . .	1204, 3993, 3994, 3999, 4000
<code>\fi:</code> . . . . .	253	<code>\iddots</code> . . . . .	71, 1136, 1196
<code>\five</code> . . . . .	3085, 3090	if commands:	
flag commands:		<code>\if_mode_math:</code> . . . . .	251
<code>\flag_clear_new:n</code> . . . . .	6582	<code>\IfBooleanTF</code> . . . . .	1504
<code>\flag_height:n</code> . . . . .	6607	<code>\ifnum</code> . . . . .	121, 4883, 4900
<code>\flag_raise:n</code> . . . . .	6606	<code>\ifstandalone</code> . . . . .	1474
		<code>\ignorespaces</code> . . . . .	1340, 2193
		int commands:	
		<code>\int_case:nnTF</code> . . . . .	3967, 3973, 3991, 3997
		<code>\int_compare:nNnTF</code> . . . . .	135, 136, 152, 857, 858, 868, 875, 912, 922, 1114, 1116, 1245, 1251, 1256, 1516, 1519, 1545, 1549, 1560, 1564, 1565, 1632, 1923, 2173, 2191, 2383, 2422, 2461, 2533, 2561, 2805, 3201, 3208, 3212, 3214, 3269, 3276, 3280, 3282, 3445, 3484, 3565, 3600, 3602, 4110, 4185, 4448, 4493, 4511, 4547, 4578, 4595, 4596, 4603, 4604, 4608, 5046, 5048, 5050, 5052, 5462, 5464, 5517, 5529, 5843, 5875, 5879, 5952, 5954, 6117, 6119, 6121, 6125, 6127, 6129, 6131, 6133, 6487, 6489

\int_compare_p:n	3347, 3348, 3349, 3350, 4463, 4465, 5944, 5945
\int_do_until:nNnn	4403
\int_gadd:Nn	2103, 2190
\int_gincr:N	856, 885, 1477, 1783, 1888, 1960, 1987, 2000, 2122, 2656, 2681, 2800, 3663, 3711, 5149, 5443
\int_if_even:nTF	4348, 6607
\int_incr:N	379, 1793, 4431
\int_min:nn	3148, 3150, 3152, 3154, 3164, 3166, 3355, 3356
\int_mod:nn	4419
\int_step_inline:nn	3146, 4344, 4346, 6500, 6517
\int_step_inline:nnnn	4978, 4996, 5011, 5014
\c_zero_int	2008, 4256, 4454
iow commands:	
\iow_now:Nn ...	62, 88, 1641, 1642, 1643, 1648
\iow_shipout:Nn	5164, 5165, 5171
\item	2388, 2394
K	
\kern	81
keys commands:	
\keys_define:nn	494, 514, 521, 599, 645, 697, 704, 748, 790, 804, 821, 834, 1301, 1399, 1826, 2093, 4013, 4355, 4364, 5138, 5368, 5608, 5969, 6057, 6097, 6221, 6226, 6306, 6327, 6336, 6698
\l_keys_key_str	2094, 6681, 6860, 6867, 6872, 6958, 7067, 7072, 7082, 7097, 7118, 7136, 7179, 7231, 7282
\keys_set:nn .	524, 526, 540, 779, 782, 789, 1338, 1403, 1411, 1499, 1500, 1860, 1981, 2102, 2884, 2896, 2906, 2915, 3037, 3448, 3487, 3568, 3637, 3689, 4113, 4188, 4205, 4359, 4380, 5151, 5688, 6228, 6232, 6359, 6412
\keys_set_known:nn	3979, 4003, 5419, 5926, 5981, 6002
\keys_set_known:nnN	2101, 6110
\l_keys_value_tl	6906, 6993, 7000, 7332
L	
\Ldots	1200, 3920, 3923
\ldots	1132, 1192
\leaders	140, 157
\leavevmode	1328, 1330
\left	1597, 2475, 2490, 6290, 6637, 6661
legacy commands:	
\legacy_if:nTF	629
\line	3009, 6941, 7093
\linewidth	2894
M	
\makebox	1957, 2272
\mathinner	73
mode commands:	
\mode_leave_vertical:	1466, 2243
msg commands:	
\msg_error:nn	12
\msg_error:nnn	13
\msg_error:nnnn	14, 5670, 5677, 5681
\msg_fatal:nn	15
\msg_fatal:nnn	16
\msg_new:nnn	18
\msg_redirect_name:nnn	20
\multicolumn	1220, 1222, 4024, 4033, 4039, 4061
\multispan	136, 137, 153, 154, 2162
\myfiledate	6
\myfileversion	7
N	
\newcounter	339
\NewDocumentCommand	351, 374, 788, 1335, 1409, 3036, 3917, 3933, 3949, 3965, 3989, 4047, 4124, 4202, 4289, 4298, 4307, 4316, 4337, 4342, 4357, 4373, 4444, 4553, 4564, 4576, 5914, 6103, 6151, 6366, 6386
\NewDocumentEnvironment	1449, 2505, 2514, 2840, 2850, 2880, 2891, 2901, 2911, 5147
\NewExpandableDocumentCommand	225, 5389
\newlist	355, 366
\NiceArray	2898, 2908, 2917
\NiceArrayWithDelims	2845, 2855
nicematrix commands:	
\g_nicematrix_code_after_tl ..	267, 638, 2529, 3015, 3018, 5691, 5705, 5717, 6239, 6246
\g_nicematrix_code_before_tl	1237, 3020, 3027, 4557, 4568, 4580, 5728, 5740
\NiceMatrixLastEnv	225
\NiceMatrixOptions	788, 7137, 7337
\NiceMatrixoptions	6997
\noalign	110, 116, 121, 145, 1096, 1172, 4883, 5060
\nobreak	369
\normalbaselines	1152
\NotEmpty	1216
\null	2189
\nulldelimiterspace	2486, 2498, 6275, 6479
\nullfont	6286, 6293, 6633, 6640, 6657, 6664
\numexpr	165, 166
O	
\omit	135, 2563, 2579, 2655, 2680, 6236
\OnlyMainNiceMatrix	1212, 4587
P	
\par	2382, 2390
\path	6069
peek commands:	
\peek_meaning:NNTF	170, 381, 1126
\peek_meaning_ignore_spaces:NNTF	2507, 4886
\peek_meaning_remove_ignore_spaces:NNTF	160
\peek_remove_spaces:n	4555, 4566, 5391, 6368, 6388
\pgfdeclareshape	3083
\pgfextracty	5846
\pgfgetlastxy	443
\pgffpathcircle	3905
\pgffpathlineto	4714, 4720, 4850, 4856, 6041, 6054, 6204, 6494, 6511, 6545
\pgffpathmoveto	4713, 4719, 4849, 4855, 6040, 6053, 6199, 6493, 6510, 6536
\pgffpathrectanglecorners	4521, 4707, 4843, 5960
\pgfpointhead	441
\pgfpointanchor	175, 228, 3106, 3119, 3371, 3382, 3399, 3421, 3529, 3547,

5210, 5218, 5790, 5808, 5828, 5830, 5847, 5884, 6267, 6426, 6433, 6462, 6469, 6581, 6585	\renewcommand . . . . . 190
\pgfpointdiff . . . . . 442, 1356, 1362, 1428, 1442, 1443	\RenewDocumentEnvironment . . . . . 6076, 6079, 6082, 6085, 6088
\pgfpointlineattime . . . . . 3775	\RequirePackage . . . . . 1, 3, 9, 10, 11, 131
\pgfpointorigin . . . . . 2570, 2737, 3091	\right . . . . . 1615, 2482, 2494, 6299, 6645, 6669
\pgfpointscale . . . . . 441	\rotate . . . . . 1211
\pgfpointshapeborder . . . . . 4243, 4246	\roundedrectanglecolor . . . . . 1377, 5730
\pgfrememberpicturepositiononpagetrue . . . . . 890, 985, 1084, 2569, 2593, 2611, 2642, 2668, 2706, 2735, 3069, 3096, 3145, 3732, 4242, 4686, 4824, 5081, 5126, 5253, 5263, 5274, 5765, 5928, 6018, 6194, 6253, 6414	\rowcolor . . . . . 1143, 1378
\pgfscope . . . . . 3104, 3117, 3772, 6211	\rowcolors . . . . . 1379
\pgfset 410, 435, 986, 5861, 5898, 6210, 6274, 6416	\rowlistcolors . . . . . 1380
\pgfsetbaseline . . . . . 984	\RowStyle . . . . . 1217
\pgfsetcornersarced . . . . . 4520, 5936	
\pgfsetlinewidth . . . . . 4723, 4859, 5963, 6021, 6481	S
\pgfsetrectcap . . . . . 4724, 4860	\savedanchor . . . . . 3085
\pgfsetroundcap . . . . . 6207	\savenotes . . . . . 1453
\pgfsetstrokecolor . . . . . 5934	scan commands:
\pgfsyspdfmark . . . . . 65, 66	\scan_stop: . . . . . 3016
\pgftransformrotate . . . . . 3779	\scriptstyle . . . . . 863, 2758, 2804, 3760, 3761, 3795, 3805
\pgftransformshift 416, 441, 3105, 3118, 3156, 3773, 5860, 5882, 6212, 6216, 6276, 6559, 6569	seq commands:
\pgfusepath . . . . . 3799, 3809, 4274, 4710, 5964	\seq_clear:N . . . . . 1676
\pgfusepathqfill . . . . . 3911, 4846	\seq_clear_new:N . . . . . 4376, 4948
\pgfusepathqstroke . . . . . 4725, 4861, 6042, 6055, 6208, 6495, 6512, 6546	\seq_count:N . . . . . 2534, 4259, 4421
\phantom . . . . . 3930, 3946, 3962, 3986, 4009	\seq_gclear:N . . . . . 1223, 1224, 1266, 1268, 1480, 1481, 1482, 1483, 2410, 3013
\pNiceMatrix . . . . . 6077	\seq_gclear_new:N 1269, 1270, 1388, 2530, 2546
prg commands:	\seq_gpop_left:Nn . . . . . 2536, 2548
\prg_do_nothing: . . . . . 69, 186, 201, 210, 219, 511, 1172, 3012	\seq_gput_left:Nn . . . . . 634, 2175, 2177, 5700
\prg_new_conditional:Nnn . . . . . 4451, 4459	\seq_gput_right:Nn . . . . . 380, 1756, 2178, 3326, 4258, 5550, 5562, 5712, 6182, 6354, 6376
\prg_replicate:nn . . . . . 389, 2651, 2652, 4061, 4715, 4851, 6120, 6123, 6126, 6132	\seq_gset_from_clist:Nn . . . . . 2931, 2945, 2953, 2955
\prg_return_false: . . . . . 4456, 4468	\seq_gset_map_x:Nnn . . . . . 3040
\prg_return_true: . . . . . 4457, 4467	\seq_gset_split:Nnn . . . . . 2532, 2547
\ProcessKeysOptions . . . . . 6715	\seq_if_empty:NTF 2369, 2941, 2949, 4964, 5738
\ProvideDocumentCommand . . . . . 71	\seq_if_empty_p:N . . . . . 2986, 2987, 2988, 4473
\ProvidesExplPackage . . . . . 4	\seq_if_in:NnTF . . . . . 632, 4513, 4663, 4669, 4676, 4801, 4807, 4814, 5213, 5221, 5788, 5806, 6350, 6772
	\seq_item:Nn . . . . . 1249, 1254, 1344, 1345, 1346, 1347, 4440, 4442
Q	\seq_map_function:NN . . . . . 2538
\quad . . . . . 370	\seq_map_indexed_inline:Nn . . . . . 4254, 4269
quark commands:	\seq_map_inline:Nn . . . . . 2388, 2394, 2550, 3340, 4410, 4625, 4627, 4629, 4763, 4765, 4767, 5041, 5653, 6485
\q_stop . . . . . 132, 133, 149, 150, 171, 173, 1392, 1414, 1503, 1678, 1748, 1821, 2040, 2062, 2169, 2212, 2524, 2527, 4196, 4210, 4211, 4278, 4327, 4332, 4395, 4483, 4484, 4506, 4507, 4537, 4538, 5347, 5354, 5394, 5395, 5399, 5727, 5919, 5942, 5951, 5982, 5985, 6011, 6014, 6092, 6107, 6370, 6375, 6400, 6403, 6483, 6592, 6600	\seq_mapthread_function:NNN . . . . . 5342
	\seq_new:N . . . . . 240, 256, 291, 292, 293, 294, 295, 296, 298, 299, 304, 308, 340, 6762
R	\seq_put_right:Nn . . . . . 5028, 5612
\raggedleft . . . . . 1881, 5481	\seq_set_eq:NN . . . . . 4385
\raggedright . . . . . 1880, 5482	\seq_set_filter:Nnn . . . . . 4386, 4408
\rectanglecolor . . . . . 1376, 3022, 4570	\seq_set_from_clist:Nn . . . . . 4968, 6763
\refstepcounter . . . . . 398	\seq_set_map_x:Nnn . . . . . 6768
regex commands:	\seq_set_split:Nnn . . . . . 4377
\regex_const:Nn . . . . . 59	\seq_use:Nn . . . . . 5747
\regex_match:nnTF . . . . . 6348	\seq_use:Nnnn . . . . . 2946, 2954, 2956, 4969, 6954, 7342
\regex_replace_all:NnN . . . . . 1683	\l_tmpa_seq . . . . . 4386, 4408
\relax . . . . . 165, 166	\l_tmpb_seq . . . . . 4385, 4386, 4408, 4410
	\setlist . . . . . 356, 367, 670, 675, 686, 691
	siunitx commands:
	\siunitx_cell_begin:w . 204, 1885, 1973, 1982
	\siunitx_cell_end: . . . . . 1886, 1985

skip commands:

<code>\skip_gadd:Nn</code> .....	2627
<code>\skip_gset:Nn</code> .....	2618
<code>\skip_gset_eq:NN</code> .....	2625, 2626
<code>\skip_horizontal:N</code> .....	141, 158, 1288, 1289, 1511, 1512, 1567, 1568, 1604, 1605, 1608, 1609, 1627, 1628, 1667, 1673, 1759, 2129, 2142, 2149, 2290, 2297, 2499, 2500, 2502, 2503, 2574, 2575, 2587, 2589, 2605, 2607, 2629, 2636, 2638, 2657, 2662, 2664, 2682, 2691, 2696, 2698, 2700, 2702, 2728, 2786, 2787, 2788, 2791, 2826, 2831, 2832, 2833
<code>\skip_horizontal:n</code> .....	401, 1032, 1805
<code>\skip_vertical:N</code> .....	145, 1078, 1080, 1600, 1611, 2379, 2404, 5060
<code>\skip_vertical:n</code> .....	927, 4893
<code>\g_tmpa_skip</code> .....	2618, 2625, 2626, 2627, 2629, 2657, 2682
<code>\c_zero_skip</code> .....	1177
<code>\smash</code> .....	6670, 6671
<code>\space</code> .....	264, 265
<code>\stepcounter</code> .....	387, 392
str commands:	
<code>\c_backslash_str</code> .....	264
<code>\c_colon_str</code> .....	787
<code>\str_case:nn</code> .....	1877, 5478, 5853, 5865, 5890, 5902, 6530, 6539
<code>\str_case:nnTF</code> .....	1572, 1726, 2197, 2336, 4951, 6604, 6617
<code>\str_clear_new:N</code> .....	6411
<code>\str_gclear:N</code> .....	3032
<code>\str_gset:Nn</code> .....	1463, 2844, 2853, 2882, 2895, 2903, 2913, 6142
<code>\str_if_empty:N</code> .....	894, 997, 1087, 1462, 2571, 2596, 2614, 2645, 2671, 2717, 2738, 2843, 2852, 3160, 3172, 5334, 5361, 5414, 6551, 6565, 6575
<code>\str_if_eq:nnTF</code> .....	96, 263, 1067, 1751, 1754, 1798, 1821, 1851, 1868, 1871, 1884, 1885, 1886, 1965, 2010, 2042, 2064, 2087, 2136, 2284, 2519, 2521, 2554, 4440, 5932, 6244
<code>\str_if_eq_p:nn</code> .....	500, 1690, 1691, 1764, 4487, 4491, 4541, 4545, 5404, 5409
<code>\str_if_in:NnTF</code> .....	2324, 2436
<code>\str_new:N</code> .....	235, 259, 313, 481, 786
<code>\str_range:Nnn</code> .....	2328, 2440
<code>\str_set:Nn</code> .....	236, 314, 631, 775, 1779, 1828, 1830, 1832, 1834, 1836, 1839, 1841, 1846, 1859, 1872, 1874, 1948, 2097, 2220, 2263, 5370, 5372, 5374, 5376, 5378, 5380, 5382, 5384, 5415, 5418, 5469, 5576, 5592, 5629, 5631, 5633, 5635, 5638, 5641, 5644, 5646, 5876, 5880, 6353
<code>\str_set_eq:NN</code> .....	635, 787, 5416
<code>\l_tmpa_str</code> .....	631, 632, 634, 635
<code>\strut</code> .....	2388, 2394
<code>\strutbox</code> .....	3523, 3541, 6453, 6456
<code>\SubMatrix</code> .....	1384, 2996, 6379, 6882, 6907, 6914, 6920, 6924, 6947, 7100

sys commands:

<code>\sys_if_engine_xetex_p:</code> .....	1029
<code>\sys_if_output_dvi_p:</code> .....	1029

T

<code>\tabcolsep</code> .....	1060, 1604, 1608
<code>\tabskip</code> .....	1177
<code>\tabularnote</code> .....	351, 374, 381, 7025, 7045
<code>\tabularnotes</code> .....	2393
T <sub>E</sub> X and L <sup>A</sup> T <sub>E</sub> X <sub>ε</sub> commands:	
<code>\@BTnormal</code> .....	1150
<code>\@acol</code> .....	1050
<code>\@acoll</code> .....	1048
<code>\@acolr</code> .....	1049
<code>\@addamp</code> .....	2164
<code>\@addtopreamble</code> .....	2171
<code>\@array@array</code> .....	1052
<code>\@arrayacol</code> .....	1048, 1049, 1050
<code>\@arstrut</code> .....	2187
<code>\@arstrutbox</code> .....	887, 888, 927, 1179, 1181, 1183, 1186, 1188, 1901, 1912, 1926, 1932, 2245, 2249
<code>\@currentenv</code> .....	6244
<code>\@depth</code> .....	6296, 6642, 6666
<code>\@empty</code> .....	2171
<code>\@finalstrut</code> .....	1912
<code>\@firststampfalse</code> .....	2164
<code>\@gobblethree</code> .....	66
<code>\@halignto</code> .....	1051, 1063, 1064
<code>\@height</code> .....	124, 140, 157, 6294, 6641, 6665
<code>\@ifclassloaded</code> .....	51, 54, 6736, 6746
<code>\@ifnextchar</code> .....	1228
<code>\@ifpackageloaded</code> .....	30, 33, 36, 39, 87, 103, 179, 6739, 6749
<code>\@mainaux</code> .....	62, 88, 1641, 1642, 1643, 1648, 5164, 5165, 5171
<code>\@mkpream</code> .....	2170
<code>\@preamble</code> .....	2188
<code>\@preamerr</code> .....	2164
<code>\@sharp</code> .....	2186
<code>\@tabarray</code> .....	1065
<code>\@tempswafalse</code> .....	1660, 2167
<code>\@tempswatrue</code> .....	1659, 2166
<code>\@temptokena</code> .....	192, 193, 209, 212, 1658, 1678, 2165, 2169
<code>\@whiles</code> .....	1660, 2167
<code>\@width</code> .....	124, 6297, 6643, 6667
<code>\@xhline</code> .....	127
<code>\bBigg@</code> .....	1282, 1284
<code>\c@MaxMatrixCols</code> .....	2872, 6800
<code>\c@tabularnote</code> .....	2372, 2383, 2411
<code>\col@sep</code> .....	1060, 1061, 1567, 1628, 2574, 2627, 2691, 2791, 2826, 3410, 3432
<code>\CT@arc</code> .....	107, 108
<code>\CT@arc@</code> .....	106, 111, 125, 139, 156, 172, 174, 1470, 2405, 3034, 4722, 4858, 5125, 5933, 6020, 6206, 6484
<code>\CT@drs</code> .....	113, 114
<code>\CT@drsc@</code> .....	117, 4697, 4698, 4702, 4835, 4836, 4840
<code>\CT@everycr</code> .....	1170
<code>\CT@row@color</code> .....	1172
<code>\if@firststamp</code> .....	2164
<code>\if@tempswa</code> .....	1660, 2167
<code>\NC@</code> .....	1125
<code>\NC@find</code> .....	194, 210
<code>\NC@list</code> .....	1660, 2167

<code>\NC@rewrite@S</code> .....	190, 211	<code>\tl_if_eq:NnTF</code> ..	1106, 1664, 2139, 2287, 2312, 2431, 4742, 4878, 5100, 5102, 6499, 6516
<code>\new@ifnextchar</code> .....	1228	<code>\tl_if_eq:nnTF</code> ...	624, 766, 2040, 2062, 4255
<code>\newcol@</code> .....	1127, 1128	<code>\tl_if_exist:NnTF</code> .....	1487
<code>\nicematrix@redefine@check@rerun</code> ..	88, 91	<code>\tl_if_in:NnTF</code> .....	4394, 4482, 4505, 4536
<code>\pgf@relevantforpicturesizefalse</code> ....		<code>\tl_if_in:nnTF</code> .....	2027, 2037, 2052
.....	1351, 3070, 3097, 3733, 3902, 4268, 4390, 4687, 4825, 5254, 5264, 5275, 5766, 5929, 6019, 6193, 6254, 6415	<code>\tl_if_single_token:nTF</code> .....	563, 774
<code>\pgfsys@getposition</code> .....		<code>\tl_if_single_token_p:n</code> .....	58
.....	1349, 1354, 1360, 1426, 1434, 1437	<code>\tl_item:Nn</code> .....	215, 216, 218
<code>\pgfsys@markposition</code> .....		<code>\tl_map_inline:nn</code> .....	2517
....	1014, 1022, 1079, 1162, 1348, 2567, 2588, 2606, 2637, 2663, 2699, 2731, 3112, 3125	<code>\tl_new:N</code> .....	214, 217, 248, 255, 267, 268, 269, 272, 279, 281, 306, 307, 311, 316, 341, 466, 470, 489, 491, 492
<code>\pgfutil@check@rerun</code> .....	93, 94	<code>\tl_put_left:Nn</code> .....	1150, 1397
<code>\reserved@a</code> .....	126	<code>\tl_put_right:Nn</code> ..	614, 1160, 1258, 1297, 1496
<code>\rvtx@ifformat@geq</code> .....	57	<code>\tl_range:nnn</code> .....	96
<code>\set@color</code> .....	5462, 5762	<code>\tl_set:Nn</code> ..	215, 4333, 4334, 4396, 4412, 4415, 4492, 4494, 4510, 4512, 4546, 4548, 4617, 5420, 5953, 5955, 5986, 5987, 6015, 6016
<code>\tikz@library@external@loaded</code> .....	1471	<code>\tl_set_eq:NN</code> .....	468, 4330, 4331, 4495, 4635, 4773, 5104, 5134, 5983, 5984, 6012, 6013, 6373, 6374, 6401, 6402, 6404, 6405
tex commands:		<code>\tl_set_rescan:Nnn</code> .....	
<code>\tex_mkern:D</code> .....	75, 77, 79, 82	.....	2531, 3916, 4046, 4123, 4201, 6385
<code>\tex_the:D</code> .....	193	<code>\tl_to_str:n</code> .....	6769
<code>\textfont</code> .....	2350	<code>\g_tmpa_tl</code> .....	212, 215, 218
<code>\textit</code> .....	343	tmpc commands:	
<code>\textsuperscript</code> .....	344, 345	<code>\l_tmpc_dim</code> .....	286, 3153, 3157, 4498, 4499, 4522, 4693, 4704, 4709, 4714, 4720, 4831, 4845, 4850, 4856, 5772, 5778, 5822, 5950, 5961, 6038, 6041, 6201, 6204, 6212
<code>\the</code> .....	165, 166, 1660, 1678, 2167, 2169	<code>\l_tmpc_int</code> .....	4401, 4402, 4403
<code>\thetabularnote</code> .....	346	<code>\l_tmpc_tl</code> .....	4328, 4330, 4333, 4495, 4514, 4618, 4634, 4635, 4638, 4643, 4645, 4649, 4654, 4656, 4756, 4772, 4773, 4776, 4781, 4783, 4787, 4792, 4794, 5983, 5991, 5993, 6012, 6029, 6035, 6371, 6373, 6377
<code>\tikzexternaldisable</code> .....	1473	tmpd commands:	
<code>\tikzset</code> .....	1369, 1475, 2999	<code>\l_tmpd_dim</code> .....	
tl commands:		.....	287, 3155, 3158, 4519, 4522, 4705, 4709, 4841, 4845, 5774, 5778, 5800, 5811, 5815, 5818, 5822, 5959, 5962, 6203, 6204, 6216
<code>\tl_clear:N</code> ....	4645, 4656, 4783, 4794, 5924	<code>\l_tmpd_tl</code> .....	4329, 4331, 4334, 5984, 5988, 5996, 6013, 6025, 6046, 6372, 6374, 6377
<code>\tl_clear_new:N</code> ..	4328, 4329, 4378, 4414, 4618, 4756, 6371, 6372, 6396, 6397, 6398, 6399	token commands:	
<code>\tl_const:Nn</code> .....		<code>\token_to_str:N</code> .....	6757, 6758, 6786, 6877, 6882, 6888, 6906, 6914, 6920, 6924, 6941, 6947, 6965, 6978, 6984, 7025, 7032, 7045, 7062, 7072, 7084, 7093, 7099, 7137, 7337
.....	42, 43, 46, 47, 467, 2749, 2794, 6593		
<code>\tl_count:n</code> .....	2331, 2443		
<code>\tl_gclear:N</code> .....	854, 884, 1485, 1492, 1662, 2168, 3011, 3018, 4273		
<code>\tl_gclear_new:N</code> .....			
.....	1231, 1232, 1233, 1234, 1235, 1236, 1237, 1484		
<code>\tl_gput_left:Nn</code> .....	1037, 1695, 4580		
<code>\tl_gput_right:Nn</code> ..	1037, 1521, 1707, 1758, 1801, 1815, 2025, 2046, 2054, 2067, 2073, 2079, 2130, 2929, 2943, 2951, 3024, 4049, 4126, 4261, 4557, 4568, 4898, 4966, 5065, 5446, 5691, 5705, 5717, 5728, 5740, 5752, 6172		
<code>\tl_gset:Nn</code> .....	212, 216, 218, 1337, 1455, 1456, 1457, 1645, 1666, 1672, 2012, 2013, 2043, 2069, 3026, 4259		
<code>\tl_if_blank:nTF</code> .....			
.....	4291, 4294, 4300, 4303, 4309, 4312, 4318, 4321, 4428, 4621, 4655, 4759, 4793, 5393		
<code>\tl_if_blank_p:n</code> .....			
.....	4486, 4490, 4540, 4544, 4698, 4836, 5403, 5408		
<code>\tl_if_empty:NnTF</code> ..	1104, 1493, 1502, 1595, 1613, 2382, 2994, 2995, 3020, 4425, 4508, 4509, 4634, 4638, 4649, 4772, 4776, 4787, 5461, 5703, 5725, 5930, 6288, 6482, 6635, 6659		
<code>\tl_if_empty:nTF</code> ..	612, 750, 792, 808, 824, 845, 2516, 2543, 3449, 3488, 3569, 3638, 3690, 4114, 4189, 4206, 6345, 6602, 6670, 6785		
<code>\tl_if_empty_p:N</code> ....	1701, 1713, 3769, 3770		
<code>\tl_if_empty_p:n</code> .....	2373, 5434		
<code>\tl_if_eq:NNTF</code> .....	3734		

## U

`\unskip` .....

use commands:

`\use:N` .....

.....

`\use:n` .....

.....

`\usepackage` .....

`\usepgfmodule` .....

## V

vbox commands:

`\vbox:n` .....

`\vbox_set_top:Nn` .....

<code>\vbox_to_ht:nn</code> . . . . .	421, 448, 2478, 2491	<code>\vNiceMatrix</code> . . . . .	6080
<code>\vbox_to_zero:n</code> . . . . .	926	<code>\vrule</code> . . . . .	123, 1901, 2245, 2249
<code>\vcenter</code> . . . . .	1598, 2476, 6291, 6638, 6662, 7062	<code>\vskip</code> . . . . .	122
<code>\Vdots</code> . . . . .	1202, 3952, 3955	<code>\vtop</code> . . . . .	1076
<code>\vdots</code> . . . . .	1134, 1194		
<code>\Vdotsfor</code> . . . . .	1209		<b>X</b>
<code>\vfill</code> . . . . .	424, 450	<code>\xglobal</code> . . . . .	871, 878, 2768, 2812
<code>\vline</code> . . . . .	125		<b>Z</b>
<code>\VNiceMatrix</code> . . . . .	6083	<code>\Z</code> . . . . .	6348

## Contents

<b>1</b>	<b>The environments of this package</b>	<b>2</b>
<b>2</b>	<b>The vertical space between the rows</b>	<b>2</b>
<b>3</b>	<b>The vertical position of the arrays</b>	<b>3</b>
<b>4</b>	<b>The blocks</b>	<b>4</b>
4.1	General case . . . . .	4
4.2	The mono-column blocks . . . . .	5
4.3	The mono-row blocks . . . . .	6
4.4	The mono-cell blocks . . . . .	6
4.5	Horizontal position of the content of the block . . . . .	7
<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 nicematrix for the rules . . . . .	9
5.3.1	The keys <code>hlines</code> and <code>vlines</code> . . . . .	9
5.3.2	The keys <code>hvlines</code> and <code>hvlines-except-borders</code> . . . . .	9
5.3.3	The (empty) corners . . . . .	10
5.4	The command <code>\diagbox</code> . . . . .	11
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 nicematrix in the <code>\CodeBefore</code> . . . . .	12
6.3	Color tools with the syntax of <code>colortbl</code> . . . . .	16
<b>7</b>	<b>The command <code>\RowStyle</code></b>	<b>16</b>
<b>8</b>	<b>The width of the columns</b>	<b>17</b>
8.1	Basic tools . . . . .	17
8.2	The columns <b>X</b> . . . . .	18
<b>9</b>	<b>The exterior rows and columns</b>	<b>18</b>

<b>10</b>	<b>The continuous dotted lines</b>	<b>20</b>
10.1	The option nullify-dots . . . . .	21
10.2	The commands <code>\Hdotsfor</code> and <code>\Vdotsfor</code> . . . . .	22
10.3	How to generate the continuous dotted lines transparently . . . . .	23
10.4	The labels of the dotted lines . . . . .	24
10.5	Customisation of the dotted lines . . . . .	24
10.6	The dotted lines and the rules . . . . .	25
<b>11</b>	<b>The <code>\CodeAfter</code></b>	<b>25</b>
11.1	The command <code>\line</code> in the <code>\CodeAfter</code> . . . . .	25
11.2	The command <code>\SubMatrix</code> in the <code>\CodeAfter</code> . . . . .	26
<b>12</b>	<b>The notes in the tabulars</b>	<b>28</b>
12.1	The footnotes . . . . .	28
12.2	The notes of tabular . . . . .	28
12.3	Customisation of the tabular notes . . . . .	30
12.4	Use of <code>{NiceTabular}</code> with <code>threeparttable</code> . . . . .	32
<b>13</b>	<b>Other features</b>	<b>32</b>
13.1	Use of the column type S of <code>siunitx</code> . . . . .	32
13.2	Alignment option in <code>{NiceMatrix}</code> . . . . .	32
13.3	The command <code>\rotate</code> . . . . .	33
13.4	The option <code>small</code> . . . . .	33
13.5	The counters <code>iRow</code> and <code>jCol</code> . . . . .	34
13.6	The option <code>light-syntax</code> . . . . .	34
13.7	Color of the delimiters . . . . .	35
13.8	The environment <code>{NiceArrayWithDelims}</code> . . . . .	35
<b>14</b>	<b>Use of Tikz with <code>nicematrix</code></b>	<b>35</b>
14.1	The nodes corresponding to the contents of the cells . . . . .	35
14.2	The “medium nodes” and the “large nodes” . . . . .	36
14.3	The nodes which indicate the position of the rules . . . . .	37
14.4	The nodes corresponding to the command <code>\SubMatrix</code> . . . . .	38
<b>15</b>	<b>API for the developpers</b>	<b>39</b>
<b>16</b>	<b>Technical remarks</b>	<b>40</b>
16.1	Definition of new column types . . . . .	40
16.2	Diagonal lines . . . . .	40
16.3	The “empty” cells . . . . .	41
16.4	The option <code>exterior-arraycolsep</code> . . . . .	41
16.5	Incompatibilities . . . . .	41
<b>17</b>	<b>Examples</b>	<b>42</b>
17.1	Utilisation of the key “tikz” of the command <code>\Block</code> . . . . .	42
17.2	Notes in the tabulars . . . . .	42
17.3	Dotted lines . . . . .	43
17.4	Dotted lines which are no longer dotted . . . . .	44
17.5	Stacks of matrices . . . . .	45
17.6	How to highlight cells of a matrix . . . . .	48
17.7	Utilisation of <code>\SubMatrix</code> in the <code>\CodeBefore</code> . . . . .	50
<b>18</b>	<b>Implementation</b>	<b>51</b>
<b>19</b>	<b>History</b>	<b>217</b>
	<b>Index</b>	<b>224</b>